

Faster, Higher, Easier: Toward a Systematic Study of Parameterized Vertex and Edge Selection Problems

Philipp Schepper

A dissertation submitted toward the degree
Doctor of Natural Sciences (Dr. rer. nat.)
of the Faculty of Mathematics and Computer Science
of Saarland University

Saarbrücken, 2025

Colloquium

Date	2025-09-29
Dean of the Faculty	Prof. Dr. Roland Speicher
Chair of the Committee	Prof. Dr. Danupon Nanongkai
Reporters	Prof. Dr. Dániel Marx Prof. Dr. Karl Bringmann Prof. Michael Lampis, PhD
Academic Assistant	Dr. Tim A. Hartmann

FASTER, HIGHER, EASIER: TOWARD A SYSTEMATIC STUDY OF
PARAMETERIZED VERTEX AND EDGE SELECTION PROBLEMS



Zusammenfassung

In dieser Arbeit betrachten wir die Entscheidungs- und Zählvarianten von Λ -FACTOR und (σ, ρ) -DOMINATING SET. Abhängig von den Mengen Λ , σ und ρ von nicht-negativen Zahlen, erfassen wir damit unter anderem MATCHING, INDEPENDENT SET oder DOMINATING SET.

Das Ziel von Λ -FACTOR ist, für einen gegebenen Graphen G eine *Kantenmenge* $S \subseteq E(G)$ zu finden, sodass für jeden Knoten v die Anzahl der angrenzenden Kanten in S in Λ liegt.

Das Ziel von (σ, ρ) -DOMINATING SET ist, für einen gegebenen Graphen G eine *Knotenmenge* $S \subseteq V(G)$ zu finden, sodass für jeden Knoten v in S die Zahl der Nachbarn in S in σ liegt und für jeden Knoten v , der nicht in S liegt, die Zahl der Nachbarn in S in ρ liegt.

Wenn die Mengen Λ , σ und ρ oder ihre Komplemente zu $\mathbb{N}$ endlich sind, geben wir durch eine Parametrisierung anhand der Baumweite (*treewidth*) effiziente Algorithmen für beide obigen Problemfamilien an. Mit bedingten unteren Schranken basierend auf einer Variante der STARKEN EXPONENTIALZEIT HYPOTHESE (*SETH*) beweisen wir, dass unsere Programme annähernd optimal sind.

Wenn extrem *wenige Grade verboten* sind, parametrisieren wir (zusätzlich zur Baumweite) anhand dieser Zahl und ermöglichen so einen deutlich besseren Algorithmus.

Wir erweitern die Beweise auf die Zählvarianten der betrachteten Probleme und zeigen, dass die Algorithmen in diese Fällen optimal (in Abhängigkeit von der Baumweite) sind.

Abstract

In this thesis we consider the decision and counting versions of the two problem families Λ -FACTOR and (σ, ρ) -DOMINATING SET which generalize MATCHING, DOMINATING SET, and INDEPENDENT SET for suitable choices of sets Λ , σ , and ρ of non-negative integers.

For Λ -FACTOR, for a given graph G , the goal is to find a set of *edges* $S \subseteq E(G)$ such that, for every vertex v , the number of incident edges from S is contained in Λ .

For (σ, ρ) -DOMINATING SET, for a given graph G , the goal is to find a set of *vertices* $S \subseteq V(G)$ such that, for every vertex in S , the number of adjacent vertices in S is contained in σ and, for every vertex not contained in S , the number of adjacent vertices in S is contained in ρ .

We restrict ourselves to the cases when Λ , σ , and ρ are finite or cofinite and prove efficient algorithms for the two problem families parameterized by treewidth. We complement these algorithmic results by conditional lower bounds, which are based on a variant of the STRONG EXPONENTIAL TIME HYPOTHESIS, to prove that the algorithms are optimal in most cases.

When the cofinite sets exclude only few degrees, we additionally parameterize by the *number of forbidden values* and provide improved algorithmic results.

We extend the general algorithms to the counting versions and provide matching lower bounds for this setting.

Acknowledgements

First and foremost I want to thank Dániel that he agreed more than five years ago to host me in his group for the time of my PhD and to be my supervisor.

Despite the difficulties of starting a PhD during a pandemic, Dániel, his group, and entire D1 of the MPI Informatics (at which Dániel's group still officially was at that time) provided one of the best working atmospheres I could have ever imagined. This welcoming and friendly environment worked as a catalyst for new projects and I hope this will not change in the future. Special thanks to all the MPI people for hosting us all this time until (or partially even after) our sudden move from D1 (on Campus) to D1 (in St. Ingbert). To be honest, I guess I mentally never left the MPI Informatics and still miss this unique working atmosphere.

I also want to thank Dániel for providing me and his entire group with this insanely high degree of freedom for choosing projects to work on and whom to collaborate with. Additionally, I also really appreciated having the possibility of attending all the conferences I have been to and the chances to meet the people there.

I also want to thank all my coauthors and (former) colleagues from Saarbrücken and St. Ingbert for the nice and friendly atmosphere and the multitude of projects I have been involved in.

Although mentioning some people explicitly, creates the hassle of having to leave out others, I still feel indebted to thank few of my colleagues explicitly. First, I want to thank Philip for the countless conversations (though sometimes controversial) about $\LaTeX$ dos and don'ts, good scientific writing and notation, trains, quality of food, and what not. It helped a lot to settle at D1. Second, I want to thank Jacob for the awesome collaborations on the projects we had. The mutual feedback on the work for (σ, ρ) -DOMSET helped me a lot in thinking about my own writing style. Working together with you was always extremely fun. Third, I am also very grateful for the collaborations I had together with Karol—too sad we did not have more. Additionally, I also want to thank Karol for organizing the joint visit of Michał Pilipczuk in Warsaw before HALG 2024. Special thanks also to Tim as my former office mate as he had to suffer a lot from my complaints about my own DP write-ups. Kudos also to Ahmed for proofreading and his feedback on a draft of this thesis.

Last but certainly not least, möchte ich mich auch noch bei meinen Eltern bedanken. Auch wenn ihr fachlich nicht helfen konntet, wart ihr doch in anderen Bereichen eine große Unterstützung, sodass mehr Zeit für diese Arbeit blieb. Danke.

Preface

This thesis is mainly based on two projects that resulted in three conference papers.

- The results about Λ -FACTOR are based on the ICALP paper [104] (see [103] for the full version) and the follow-up IPEC paper [105] (see [106] for the journal version in *Algorithmica*). For this project, the authors contributed equally to the research and the write-up.

The revised results about Λ -FACTOR constitute Part II of this thesis.

- The results about (σ, ρ) -DOMSET are based on the SODA paper [53] which was, due to its length, split into the algorithmic results (see [56] for the journal version in *TALG*) and the lower bound results (see [57] for the journal version in *TOCT*). For this project, the author of the current thesis foremost contributed to the lower bound results.

The work on [70] (see [69] for the full version) inspired a revision of the results about (σ, ρ) -DOMSET which resulted in an optimized presentation of the algorithms. We note however, that no formal results from [69, 70] are included in this thesis.

The revised results about (σ, ρ) -DOMSET constitute Part III of this thesis.

Additionally, we unify common techniques from the original presentation of the mentioned results. More precisely, we explicitly summarize and extend the common convolution algorithms and present the results about representative sets in a self-contained way. Moreover, we provide tools that allow us to prove stronger and simpler lower bound results under (the counting version of) pw-SETH. These unified techniques constitute Part I of this thesis.

For this thesis, we revised the results in Parts II and III several times and used the tools from Part I to simplify and unify their presentation. Moreover, we strengthen several of the lower bounds compared to their original version by basing them on pw-SETH.

Most of the figures in this thesis appeared in (slightly) different form in the original presentations of the results. However, all figures have been adjusted multiple times and those which were initially created by coauthors have been redesigned from scratch. Thus, all figures and illustrations in this thesis are the work of the author.

The results from [48] (see [46] for the conference version) are, despite a similarity in name with Section 3.1, not included in this thesis. However, as this project was conducted at the same time as [53], we note that the fix for the convolution algorithm by van Rooij (see Section 3.1.1) also appeared as an appendix in [48].

The other publications of the author in [54, 65, 66] (and their conference versions [63, 64]) that were written during the time of the dissertation phase, are *not* included in this thesis. Likewise, the publications [29] and [119] which are (partially) based on the Bachelor's Thesis [118] and Master's Thesis [120] of the author, respectively, are not included as well.

Contents

1	Introduction	1
1.1	Our Contribution	6
1.2	Technical Overview for the Upper Bounds	10
1.2.1	Efficient Convolution Algorithms	10
1.2.2	Improving the Bound on the Number of States	12
1.2.3	Representative Sets	12
1.3	Technical Overview for the Lower Bound	14
1.3.1	Classical Approach	15
1.3.2	Our Approach	19
1.4	Organization	26
I	Our Toolkit	29
2	Preliminaries	31
2.1	General Notation	31
2.2	Treewidth and Pathwidth	34
3	A Toolkit for Improved Algorithms	37
3.1	Convolution Algorithms	37
3.1.1	Cyclic Addition	39
3.1.2	Standard Addition with Large Numbers	44
3.1.3	Standard Addition	48
3.1.4	Truncated Addition	51
3.2	Representative Sets	60
3.2.1	Computing Representative Sets via Matrix Multiplication	63
3.2.2	Computing Representative Sets via a Single Matroid	68
3.2.3	Compatibility Based on Forbidden and Allowed Values	70
4	A Toolkit for Proving Lower Bounds	77
4.1	From Satisfiability to Constraint Satisfaction	77
4.1.1	Primal Pathwidth	80
4.1.2	Larger Alphabets	80
4.1.3	Multi-Assignments	87
4.2	Half-Induced Matchings	97
4.2.1	Relation to Arithmetic Progressions	98
4.2.2	Relation to Representative Sets	99

II	General Factor	103
5	Introduction To General Factor	105
6	Solving General Factor	113
6.1	Parameterizing by the Largest Degree	114
6.2	Parameterizing by the Number of Excluded Degrees	120
7	Lower Bounds for General Factor	127
7.1	Outline and Preliminaries	128
7.1.1	Lower Bound for the Intermediate Problem	131
7.1.2	Extension to Cofinite Sets	132
7.1.3	Reducing to Few Types of Relations	132
7.1.4	Realizing Relations	133
7.1.5	Combining the Results	134
7.2	Lower Bound for the Intermediate Problem	138
7.3	Decision Version: Extension to Cofinite Sets	148
7.4	Counting Version: Extension to Cofinite Sets	152
7.5	Reducing to Few Types of Relations	165
7.6	Decision Version: Realizing Relations	168
7.7	Counting Version: Realizing Relations	175
7.8	Special Case: Counting Version Parameterized by List Size	194
7.9	Special Case: Counting Edge Covers	204
III	General Domination	217
8	Introduction to General Domination	219
8.1	Preliminaries	222
9	Solving General Domination	225
9.1	Parameterizing by the Largest Degree	226
9.1.1	Structural Insights	229
9.1.1.1	A Single Realized Language	233
9.1.1.2	Combinations of Realized Languages	240
9.1.2	Fast Join Operations	242
9.1.3	Finalizing the Algorithm	252
9.2	Parameterizing by the Number of Excluded Degrees	256
10	Lower Bounds for General Domination	263
10.1	Outline and Preliminaries	264
10.1.1	Notation	266
10.1.2	Providers and Managers	268
10.1.3	Lower Bound for the Intermediate Problem	270
10.1.4	Counting Version: Extension to Cofinite Sets	275

10.1.5	Reducing to One Type of Relations	276
10.1.6	Decision Version: Realizing Relations	277
10.1.7	Counting Version: Realizing Relations	277
10.1.8	Special Case: σ is Cofinite and $\rho = \mathbb{N}$	280
10.1.9	Combining the Results	281
10.2	Providers	284
10.2.1	Simple General-Purpose Providers	285
10.2.2	A Provider for σ -States	286
10.2.3	Providers for Combinations of σ - and ρ -States	288
10.2.3.1	1-Structured Sets	295
10.2.3.2	2-Structured Sets	296
10.3	Managers	298
10.3.1	A Generic Manager	302
10.4	Lower Bound for the Intermediate Problem	307
10.5	Counting Version: Extension to Cofinite Sets	317
10.6	Reducing to One Type of Relations	325
10.7	Decision Version: Realizing Relations	331
10.8	Counting Version: Realizing Relations	336
10.8.1	Shifting the Set ρ	340
10.8.2	The Case When $\rho \neq \mathbb{N}$	344
10.8.2.1	Constructing Strong Candidates and Winners	347
10.8.2.2	Winners Force the Selection Status	352
10.8.2.3	A Single Unselected Vertex with No Selected Neighbors	354
10.8.3	The Case When σ Is Finite and $\rho = \mathbb{N}$	362
10.9	Special Case: σ Is Cofinite and $\rho = \mathbb{N}$	367
10.9.1	Adjusting to the New Definition of Relations	371
10.9.2	Reducing to One Type of Relations	378
10.9.3	Realizing Relations	379
11	Conclusion	393
IV	Appendix	399
	List of Figures	401
	Bibliography	403
	Notation	417

Introduction

*People think of these eureka moments
and my feeling is that they tend to be little things,
a little realisation and then a little realisation built on that.*

— Roger Penrose

Patterns are ubiquitous in daily life. They range from abstract and vague structures such as the daily routines via mind-boggling, aperiodic tilings of the plane to easy and well understood patterns such as the Fibonacci numbers.

While probably all of these patterns have a valid reason to be studied, some of them are easier to analyze while others are more complex. Frequently, the “structure” of the patterns makes it easier to analyze them. For example, the Fibonacci numbers are comparably well understood, although the number of applications is steadily growing and possible properties and relations of these numbers might not be known yet.

Besides patterns of numbers, a different concept that allows for a compact characterization of patterns is based on *graphs*. In this context, we can interpret the meaning of patterns quite literally. That is, we are given some (sub)structure which we want to find in the graph. A simple example would be to *decide* whether a graph contains a triangle or directly *count* the number of triangles in the particular given graph.

This problem of deciding whether a graph G contains an (induced) subgraph that satisfies some property Φ is usually referred to as $\text{INDSUB}(\Phi)$ and the counting version as $\#\text{INDSUB}(\Phi)$. These problems (and the related concept of finding and counting homomorphisms) have recently received rising interest and the complexity becomes more and more clear despite the many open problems [34, 39, 40, 52, 55, 77, 84, 116, 117]. A common theme of these results is that they frequently consider *small* (induced) subgraphs, that is, (induced) subgraphs where the size is bounded by some predefined value k .

In contrast to this, we can also study problems that consider large (induced) subgraphs which are not even given explicitly. Instead, the patterns we want to find are specified by *degree constraints*. The example with the triangles from above could be generalized such that we consider (induced) subgraphs that have degree exactly two, that is, we consider cycles.

As finding *some* cycle in a graph can be done efficiently, we can think of variations of this problem; pack as many of these patterns in the graph or every vertex should be part of exactly one pattern [11, 45, 75, 76, 113]. Then, the well-known problem HAMILTONIAN CYCLE would require that there is a single cycle visiting all vertices.

Another variation could require that we only consider *induced* cycles. Phrased differently, a vertex that is part of some cycle C , must not have any edge to a vertex that is not a direct neighbor on the cycle C . More formally, two selected vertices u and v are adjacent if and only if the edge uv is part of a selected cycle.

A last variation we mention here could impose more flexible degree constraints. That is, we might also allow vertices of degree one which corresponds to finding cycles and *paths*. Similarly to the previous generalization, we could also extend this variant to the setting of induced subgraphs.

Formalizing the Degree Constraints. When we want to study the problem variations from above, we need a more formal problem definition and, more precisely, a formal specification of the degree constraints. By the definition of graphs based on vertices and edges, we have two natural ways to build the subgraph—either by selecting the vertices or by selecting the edges.

Selecting Edges. We first take a more detailed look at the significantly simpler case of selecting *edges* of some graph G , that is, we consider *non-induced* subgraphs of G . In this case, we specify the degree of the vertices in this subgraph. Formally, we first fix some set $\Lambda \subseteq \mathbb{N}$ of non-negative integers. Then, a solution is defined by an edge set $S \subseteq E(G)$, which characterizes some subgraph, such that every vertex $v \in V(G)$ has a degree in S that is contained in Λ , that is, the number of selected edges incident to v is contained in Λ .

In this thesis, the set $\mathbb{N}$ contains 0.

Depending on the set Λ , this gives rise to multiple different problems. We list some of the most frequent ones.

PERFECT MATCHING: Find a set $S \subseteq E(G)$ of edges such that every vertex is incident to *exactly one* selected edge, that is, $\Lambda := \{1\}$.

MATCHING: Find a set $S \subseteq E(G)$ of k edges such that every vertex is incident to *at most one* selected edge, that is, $\Lambda := \{0, 1\}$.

EDGE COVER: Find a set $S \subseteq E(G)$ of k edges such that every vertex is incident to *at least one* selected edge, that is, $\Lambda = \mathbb{N}_{\geq 1}$.

BOUNDED-DEGREE SUBGRAPH: Find a set $S \subseteq E(G)$ of k edges such that every vertex is incident to *at most d* selected edges, that is, $\Lambda := \{0, 1, \dots, d\}$.

d -REGULAR SUBGRAPH: Find a set $S \subseteq E(G)$ of k edges such that every vertex is incident to *exactly d* selected edges, that is, $\Lambda := \{d\}$.

Petersen [112] was one of the first to study such problems. He related multivariate polynomials and graphs, and established the name of a *graph factor* as a subgraph of the original graph. While Petersen mainly considered factors of degree 1 or 2, Tutte [128] generalized this to arbitrary numbers; the problem is named f -FACTOR when degree f is required or f -ANTI-FACTOR if degree f is forbidden [121]. Lovász [98] finally introduced the most general concept such that every vertex could potentially receive a different set; we refer to this variant as GENFAC (see also [9] for a more detailed introduction to this field). When we only consider a single set, that is, each vertex is assigned the same *fixed* set Λ , then we refer to this variant as Λ -FACTOR.

The complexity of the classical matching variants is well understood and polynomial-time algorithms are long known [42, 107]. For the more general f -FACTOR problem or variants with intervals, the algorithms frequently reduce the complexity to the classical matching problem

Combining the factors yields the original graph; similar as for the factors of a polynomial.

[9, 30, 41, 42, 78, 89, 98, 107, 123]. Cornuéjols [30] finally showed that the general problem GENFAC can be solved in polynomial time when the sets only have gaps of size at most 1.

The size of a gap is determined by the number of consecutive integers that are missing in the set; see Definition 2.1.2 for a formal definition. For example, the set $\Lambda = \{3, 4, 7, 9\}$ has a maximum gap of size 2 as 2 numbers are missing between 4 and 7, and thus, the algorithm by Cornuéjols is not applicable.

Although these cases where the algorithm does not work are not explicitly proven to be NP-hard, Cornuéjols [30] covers one particular case and the results by Dalmau and Ford [38] and Feder [50] indicate that this hardness should also be true in general.

Hence, in order to achieve (faster) algorithms for the difficult cases, different approaches are needed. One such approach is the analysis from a parameterized perspective. Before we introduce this concept more formally, we first focus on the related problem where we select vertices instead of edges.

Selecting Vertices. When we select vertices, we essentially consider *induced* subgraphs. Similarly to before, we define a set $\sigma \subseteq \mathbb{N}$ of non-negative integers. Then, for every *vertex* selection $S \subseteq V(G)$, we require that, for every vertex $v \in S$, the number of neighbors of v in S is contained in σ . For example, when we choose $\sigma = \{0\}$ this resembles INDEPENDENT SET.

As we select vertices, we are usually also left with unselected vertices. Hence, we can impose additional degree constraints on these unselected vertices. Analogously to before, we define a set $\rho \subseteq \mathbb{N}$ and require that, for every vertex v that is *not* contained in the selection S , the number of selected neighbors of v in S is contained in ρ . In the following, we refer to a vertex set S which satisfies these σ -constraints and ρ -constraints as a (σ, ρ) -set.

Depending on the choice of σ and ρ , we can cover many classical problems. In the following, we list some of the most important problems that can be reformulated as finding a (σ, ρ) -set. We refer to [18, 126] for a more extensive list of examples.

INDEPENDENT SET: Find a set $S \subseteq V(G)$ of k vertices that are pairwise non-adjacent, that is, $\sigma = \{0\}$ and $\rho = \mathbb{N}$.

STRONG INDEPENDENT SET: Find a set $S \subseteq V(G)$ of k vertices that are pairwise at distance at least three, that is, $\sigma = \{0\}$ and $\rho = \{0, 1\}$.

DOMINATING SET: Find a set $S \subseteq V(G)$ of k vertices such that every remaining vertex has a neighbor in S , that is, $\sigma = \mathbb{N}$ and $\rho = \mathbb{N}_{\geq 1}$.

INDEPENDENT DOMINATING SET: Find an independent set $S \subseteq V(G)$ of k vertices such that every remaining vertex has at least one neighbor in S , that is, $\sigma = \{0\}$ and $\rho = \mathbb{N}_{\geq 1}$.

PERFECT CODE/EXACT INDEPENDENT DOMINATING SET: Find an independent set $S \subseteq V(G)$ of k vertices such that every remaining vertex has *exactly one* neighbor in S , that is, $\sigma = \{0\}$ and $\rho = \{1\}$.

TOTAL DOMINATING SET: Find a set $S \subseteq V(G)$ of k vertices such that every vertex in the graph has *at least one* neighbor in S , that is, $\sigma = \mathbb{N}_{\geq 1}$ and $\rho = \mathbb{N}_{\geq 1}$.

PERFECT DOMINATING SET: Find a set $S \subseteq V(G)$ of k vertices such that every remaining vertex has *exactly one* neighbor in S , that is, $\sigma = \mathbb{N}$ and $\rho = \{1\}$.

INDUCED BOUNDED-DEGREE SUBGRAPH: Find a set $S \subseteq V(G)$ of k vertices such that every such vertex has *at most* d neighbors in S , that is, $\sigma = \{0, 1, \dots, d\}$ and $\rho = \mathbb{N}$.

INDUCED d -REGULAR SUBGRAPH: Find a set $S \subseteq V(G)$ of k vertices such that every such vertex has *exactly* d neighbors in S , that is, $\sigma = \{d\}$ and $\rho = \mathbb{N}$.

Telle and Proskurowski [126, 127] initially introduced this problem and referred to it as (σ, ρ) -DOMSET as it generalizes many “domination”-like problems as indicated above. However, sometimes the problem is also referred to as *locally checkable vertex subset problem (LC-VS)* [18] as the validity of a solution can be checked by inspecting the neighborhood of each vertex.

Naturally, if we do not impose a size constraint on the solution, then the problem is polynomial-time solvable when $0 \in \rho$. Indeed, the algorithm is even trivial as the empty set is a valid solution. However, as the framework covers many classical NP-hard problems, (σ, ρ) -DOMSET is NP-hard in general as well. Hence, we need to characterize the problem from a different perspective.

Parameterized Complexity. Motivated by the NP-hardness of most of the problems listed earlier, it is clear that we need to change perspective to obtain faster and better algorithms. For this approach we use parameterized complexity, that is, instead of restricting the input (graph) to have some specific form (for example, a tree), we can consider some measure of the instance that possibly provides a good interpolation between the easy cases and the hard cases. Phrased differently, in the worst case we might still have a slow algorithm, but in the best case we possibly obtain a significantly faster algorithm.

Besides the default parameter *solution size*, one of the most famous and best understood such structural graph parameters is *treewidth*. Intuitively, the treewidth measures how similar a graph is to a tree. One of the main motivations for this parameter comes from the observation that many computationally difficult problems admit an efficient algorithm for trees. More precisely, for trees we can often design a simple dynamic program that directly finds a partial solution for the leaves, then extends these solutions at the internal non-branching vertices, and combines them at the branching vertices. The concept of treewidth allows for an extension of these techniques to general graphs (of low treewidth).

While the notion of treewidth was rediscovered several times (see [10, 72], for example), the most prominent definition is by Robertson and Seymour [114] which is usually phrased as follows (see Section 2.2 for a more detailed introduction).

A tree decomposition of a graph G is a tuple (T, β) where T is a tree and β is a function that maps every vertex of T , we refer to these vertices as *nodes*, to a subset of vertices of G , we refer to these sets as *bags*, such that

- for every vertex $v \in V(G)$, there is (at least) one bag $\beta(j)$ that contains v ,
- for every edge $uv \in E(G)$, there is (at least) one bag $\beta(j)$ that contains u and v , and

- for every vertex $v \in V(G)$, the nodes whose corresponding bags contain v induce a connected subgraph of T .

Then, the *width* of a tree decomposition is the size of the largest bag minus 1. The *treewidth* of a graph G , denoted by $\text{tw}(G)$, is the minimum width over all possible tree decompositions for G . For the related concept of pathwidth, we require that the graph T is a path and not a tree.

The -1 is needed to ensure that trees have treewidth 1. However, this frequently leads to “off-by-one” errors.

Although a tree decomposition of minimum size exists for every graph, it is NP-hard to decide, for a given value k , whether the treewidth is at most k [5]. However, several approximation algorithms [90, 91, 115] and exact algorithms [5, 12, 91] are known that compute tree decompositions. See also [91] for a more detailed overview.

As the algorithms that operate on the tree decompositions are independent of the algorithms that compute the tree decompositions, both problems are of independent interest. Therefore, we usually assume that a tree decomposition of width w is given with the input graph.

Fast Parameterized Algorithms. Now, when we want to solve a problem Π parameterized by treewidth, the input consists of a graph G and a tree decomposition of width w . Then, the goal is to find an algorithm that operates in time $f(w) \cdot |G|^{\mathcal{O}(1)}$ for some computable function f . We say that this algorithm and thus, the problem Π is *fixed-parameter tractable* (FPT). With such a running time, there are two potentially opposing goals to aim for.

We abstain from the $\mathcal{O}^*$ -notation to hide the polynomial term but mark it explicitly instead.

First, the polynomial dependence on the size of the input graph should be as low as possible, that is, in best case linear. Surprisingly, Courcelle [31] showed that every graph property ϕ that can be described by monadic second-order logic on graphs (MSO_2), can be decided in time $f(w) \cdot |G|$ for some computable function f parameterized by the treewidth w . However, the function f resulting from Courcelle’s Theorem [31] is a tower of powers depending on the quantifiers of the formula ϕ . Although the running time is FPT, this makes the algorithm highly impractical. Hence, the natural question would be to find the best function f .

This monadic second-order logic extends first-order logic with the quantification over vertex sets and edges sets.

A second goal, especially in the context of Courcelle’s Theorem, but also in general, is to find the smallest function f_Π such that problem Π can be solved in time $f_\Pi(w) \cdot |G|^{\mathcal{O}(1)}$. These algorithms usually have a larger dependence on the size of the input graph, but once we know an algorithm for some significantly better function f_Π , we can frequently improve the polynomial term as well by a careful design and analysis of the algorithm. As improving this polynomial term makes the algorithms usually more complex and does not provide too much insight into the problem, we mainly focus on the improvement of the function f_Π instead of the polynomial term in $|G|$.

In most cases, we actually want to find the smallest *constant* c_Π that depends on the problem Π such that Π can be solved in time $(c_\Pi)^w \cdot |G|^{\mathcal{O}(1)}$. That is, the running time depends only single-exponentially on the treewidth. As most of the problems for which we consider a parameterization by treewidth are NP-hard, it is very unlikely to obtain a linear dependence on treewidth unless $P = NP$. We observe that such a constant does not always exist even if the problem is FPT parameterized by treewidth. For some problems a double-exponential running time in treewidth (or even worse) is necessary, that is, an algorithm with running time $2^{2^{\mathcal{O}(w)}} \cdot |G|^{\mathcal{O}(1)}$ might be optimal [54, 62].

Finding the optimal value of c_Π often results in a sequence of improvements. In general, a first approach is to identify the “states” of the vertices in a (partial) solution. This number usually determines the value of c_Π and leads to an algorithm with running time $(c_\Pi)^{\mathcal{O}(w)} \cdot |G|^{\mathcal{O}(1)}$. Then, by non-trivial improvements of the various involved subprocedures, the running time can often be improved to $(c_\Pi)^w \cdot |G|^{\mathcal{O}(1)}$.

Lower Bounds. Once, we obtain said running times, further improvements seems to require novel ideas to bypass the limitations imposed by the bound on the states. As such improvements seem to be unlikely, one might ask whether the algorithms are optimal. The ultimate goal would be to prove that there is no algorithm that can solve Π in time $(c_\Pi - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$ for any $\varepsilon > 0$.

As such an *unconditional* lower bound would imply that P and NP are different, we focus on conditional lower bounds under “plausible” conjectures. While the assumptions such as $P \neq NP$ only give very coarse lower bounds, even the assumption “SAT requires exponential time” (formalized via the EXPONENTIAL TIME HYPOTHESIS [80]) would only rule out algorithms of the form $2^{o(w)} \cdot |G|^{\mathcal{O}(1)}$.

To prove tight bounds, that is, to prove bounds as stated above, Lokshtanov, Marx, and Saurabh [97] use the STRONG EXPONENTIAL TIME HYPOTHESIS (SETH). This hypothesis intuitively states that the larger k , the closer the running time of k -SAT is to 2^n for a formula with n variables. See Conjecture 4.1.1 for the formal statement. With SETH it is possible to prove that the known algorithms are tight and no further improvement of the constant c_Π is possible.

1.1 Our Contribution

After this rough outline of the general concepts when parameterizing by treewidth, we transition back to the two problem families which we consider in this thesis, that is, Λ -FACTOR and (σ, ρ) -DOMSET together with their respective counting versions.

General Factor. For Λ -FACTOR or rather the more general GENFAC only few results are known in the parameterized setting. Arulselvan, Cseh, Groß, Manlove, and Matuschke [6] consider the case where the sets contain 0 and an interval of integers. They provide an algorithm that solves the maximization version of the problem in time $(M + 1)^{3w} \cdot |G|^{\mathcal{O}(1)}$ where M denotes the maximum over all intervals. A later result by van Rooij, Bodlaender, and Rossmanith [132] proves an efficient algorithm for COUNTING PERFECT MATCHINGS. Curticapean and Marx [35] proved that this algorithm for COUNTING PERFECT MATCHINGS is optimal and the dependence on the treewidth w cannot be improved further.

We focus on Λ -FACTOR and consider all finite or cofinite sets Λ . Note that this covers all problems listed earlier that can be phrased as Λ -FACTOR problem.

To formally state the running time, we first define some notation related to the states of the vertices. In a (partial) solution, the state of a vertex v is determined by the number of selected edges the vertex v is incident to. When the set Λ is finite and v has strictly more than $\max \Lambda$

The decision version
is trivial if degree 0
is allowed.

selected incident edges, then there is no solution. That is, a valid state is determined by a number between 0 and $\max \Lambda$; there are $\max \Lambda + 1$ states.

When the set Λ is cofinite, there is no degree from which on all higher degrees are invalid. Instead, there is some degree from which on all higher degrees are *valid*. That is, the vertex v of this particular degree is indifferent whether it receives further selected edges or not. This state is determined by the largest forbidden degree plus 1, that is, the value $\max(\mathbb{Z} \setminus \Lambda) + 1$. Hence, there are $\max(\mathbb{Z} \setminus \Lambda) + 2$ states to consider.

We need $\mathbb{Z}$ instead of $\mathbb{N}$ to handle the case when $\Lambda = \mathbb{N}$.

We unify both cases based on the value top of a set.

Definition 2.1.3 (The Top Element). *Consider a set $\tau \subseteq \mathbb{N}$ that is either finite or cofinite. Based on the set τ , we define the number $\text{top } \tau \in \tau$ as*

$$\text{top } \tau := \begin{cases} \max \tau & \text{if } \tau \text{ is finite,} \\ 1 + \max(\mathbb{Z} \setminus \tau) & \text{if } \tau \text{ is cofinite.} \end{cases}$$

With this notation, we can prove the following algorithmic result which extends the known algorithms.

Main Theorem II.1. *Fix some set $\Lambda \subseteq \mathbb{N}$ that is finite or cofinite.*

Given a Λ -FACTOR instance G along with a tree decomposition of width w , we can count the solutions of size exactly s in time $(\text{top } \Lambda + 1)^w \cdot |G|^{\mathcal{O}(1)}$ for all s simultaneously.

We observe that this algorithm not only solves the decision version but also the counting version and the exact counting version.

Surprisingly, when the set is cofinite, we can improve the algorithm such that it only depends on the number of missing values. In this case, we provide a faster algorithm for the more general problem ANTI-FACTOR_x . This problem is a generalization of Λ -FACTOR where, for each vertex v , we allow different cofinite sets $\Lambda(v)$ of allowed degrees but require that, for each set $\Lambda(v)$, at most x degrees are forbidden, that is, we require that $|\mathbb{N} \setminus \Lambda(v)| \leq x$. See Definition 5.1.2 for a formal definition.

Main Theorem II.2. *Fix some integer $x \geq 1$. Given an ANTI-FACTOR_x instance G along with a tree decomposition of width w , we can decide if there is a solution of size exactly s in time $(x + 1)^{\mathcal{O}(w)} \cdot |G|^{\mathcal{O}(1)}$ for all s simultaneously.*

Note that this algorithm only solves the exact decision version but is not applicable for the counting version.

We complement these algorithmic results with several computational hardness results. As a first result we prove that the algorithm from Main Theorem II.1 is optimal for the decision version when the sets are finite. For the case of cofinite sets we prove a weaker lower bound based on the size of the *half-induced matching*; similar concepts are known as half-graphs or (semi-)ladders [44, 49] (see Section 4.2 for a formal introduction).

Main Theorem II.3. *Fix some non-empty set $\Lambda \subseteq \mathbb{N}$ that is finite or cofinite with $0 \notin \Lambda$ and max-gap $\Lambda > 1$. If Λ is finite, we define $h := \max \Lambda + 1$, and if Λ is cofinite, then we define h as the size of the largest half-induced matching in the compatibility graph of $\mathbb{N} \setminus \Lambda$.*

If there is an $\varepsilon > 0$ and an algorithm that solves Λ -FACTOR on a given graph G with a given path decomposition of width w in time $(h - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-SETH is false.

1 Introduction

We note that the lower bound is not based on SETH, but on the *primal pathwidth STRONG EXPONENTIAL TIME HYPOTHESIS* (pw-SETH) (see Conjecture 4.1.4 for the formal definition). Lampis [96] introduced this hypothesis as a more robust version of SETH which makes the resulting lower bounds more plausible. Moreover, observe that the lower bound is stated for the larger parameter pathwidth and thus, follows immediately for the smaller parameter treewidth.

When we consider the variant of GENFAC where we allow multiple cofinite sets but restrict the number of forbidden degrees for each set, that is, when we consider ANTIFACTOR_x , then we can obtain faster algorithms by Main Theorem II.2. We show that for this problem, we can strengthen the non-tight lower bound from Main Theorem II.3 and prove that the algorithm from Main Theorem II.2 is almost optimal, that is, optimal up to the small constant hidden by the $\mathcal{O}(w)$ in the running time of the algorithm.

Main Theorem II.4. *Let $x \geq 2$ denote a fixed integer.*

If there is an $\varepsilon > 0$ and an algorithm that solves ANTIFACTOR_x on a given graph G with a given path decomposition of width w in time $(x + 1 - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-SETH is false.

For the case of the counting version, we recall that the algorithm from Main Theorem II.2 is not applicable. Indeed, for the counting version we can strengthen the lower bound from Main Theorem II.3 to match the running time of the algorithm from Main Theorem II.1.

Main Theorem II.5 (Lower Bound for Counting Version). *Fix a non-empty set $\Lambda \subseteq \mathbb{N}$ that is finite with $\Lambda \neq \{0\}$ or cofinite with $\Lambda \neq \mathbb{N}$ and $\Lambda \neq \mathbb{N}_{\geq 1}$.*

If there is an $\varepsilon > 0$ and an algorithm that solves $\Lambda\text{-}\#\text{FACTOR}$ on a given graph G with a given path decomposition of width w in time $(\text{top } \Lambda + 1 - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-#SETH is false.

We additionally prove that $\#\text{ANTIFACTOR}_x$ is $\#\text{W}[2]$ -hard and does not admit an FPT-algorithm parameterized by treewidth (see Main Theorem II.6 for the formal result).

We refer to Chapter 5 for an overview of all results for GENFAC.

General Domination. In the context of (σ, ρ) -DOMSET a more general result (compared to the results for Λ -FACTOR) is known. A first result by van Rooij, Bodlaender, and Rossmanith [132] provides a fast algorithm for (σ, ρ) -DOMSET when the sets are finite or cofinite. This again covers all problems listed earlier that can be described as (σ, ρ) -DOMSET problem. Later this algorithm was improved further by van Rooij [130].

Despite the fact that this algorithm seems to be optimal, we show that further improvements are possible. As the precise definition of the base $c_{\sigma, \rho}$ requires some more notation, we refer to Definition 8.1.4 instead but remark that in all cases we have $c_{\sigma, \rho} \leq \text{top } \sigma + \text{top } \rho + 2$.

Main Theorem III.1. *Fix two finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$. Given a graph G with a tree decomposition of width w and an integer s , we can count the number of (σ, ρ) -sets of size exactly s in time $(c_{\sigma, \rho})^w \cdot |G|^{\mathcal{O}(1)}$.*

Again we emphasize that this algorithm solves the decision version but also the harder exact counting version in the same time.

Similar to before, we improve this algorithm when at least one set is cofinite. To unify the notation for the four possible combinations of finite and cofinite sets, we define a new cost measure which takes this improvement into account.

Definition 8.1.7 (Cost of a Set). *Consider a set $\tau \subseteq \mathbb{N}$ that is either finite or cofinite. Based on the set τ , we define the number $\text{cost } \tau$ as*

$$\text{cost } \tau := \begin{cases} \max \tau & \text{if } \tau \text{ is finite,} \\ |\mathbb{N} \setminus \tau| & \text{if } \tau \text{ is cofinite.} \end{cases}$$

We observe that we always have $\text{cost } \tau \leq \text{top } \tau$. With this notation, the improved algorithm for (σ, ρ) -DOMSET reads as follows where we denote by $\omega \leq 2.371552$ the matrix multiplication exponent [135].

Main Theorem III.2. *Fix two finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$. Based on these sets we define $t_{\text{cost}} := \max\{\text{cost } \sigma, \text{cost } \rho\}$.*

Given a graph G with a tree decomposition of width w , and an integer s , we can decide whether G has a (σ, ρ) -set of size exactly s in time

$$2^w \cdot (t_{\text{cost}} + 1)^{w \cdot (\omega + 1)} \cdot |G|^{\mathcal{O}(1)}.$$

We complement these results by conditional lower bounds that follow the conceptual idea for Λ -FACTOR. However, for (σ, ρ) -DOMSET the situation is more complicated. Thus, for the decision version, we only prove a lower bound when both sets are finite.

Main Theorem III.3. *Fix two finite sets $\sigma, \rho \subseteq \mathbb{N}$ with $0 \notin \rho$. If there is an $\varepsilon > 0$ and an algorithm that decides whether there is a (σ, ρ) -set in a given graph G with a given path decomposition of width w in time $(c_{\sigma, \rho} - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-SETH fails.*

Similar to the previous lower bounds, this results holds for the larger parameter pathwidth.

In contrast to the decision version, for the counting version we can prove that the algorithmic result from Main Theorem III.1 is tight, and no further improvements are possible even if σ and ρ might be cofinite.

Main Theorem III.4. *Fix two non-empty finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$ with $\rho \neq \{0\}$ and $(\sigma, \rho) \neq (\mathbb{N}, \mathbb{N})$.*

If there is an $\varepsilon > 0$ and an algorithm that counts the number of (σ, ρ) -sets in a given graph G with a given path decomposition of width w in time $(c_{\sigma, \rho} - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-#SETH fails.

We refer to Chapter 8 for a more detailed overview of the previously known results and the comparison to our results.

Outline. Although the proofs are filled with a plethora of problem-specific constructions, our results for the variants of GENFAC and (σ, ρ) -DOMSET follow from the same basic ideas and concepts. Thus, we first provide a high-level overview of our techniques that we summarize in a toolbox before we apply these concepts to GENFAC and then to (σ, ρ) -DOMSET to prove the formal results.

1.2 Technical Overview for the Upper Bounds

The general approach to obtain some algorithmic result for a problem Π parameterized by treewidth is fairly easy. The essential idea is a dynamic programming based approach that constructs partial solutions for the subgraphs associated to the bags of the given tree decomposition.

For a first intuition, we consider a bag β of a given tree decomposition. We denote by tw the width of this decomposition, that is, essentially the size of the largest bag—for ease of presentation we ignore the additional ± 1 here. For now, it suffices to think of a bag just as a separator of the graph. Then, a *partial solution* S is a relaxed form of a solution for one half of the graph (determined by the separator β) where the constraints imposed by the problem definition of Π are possibly violated at the vertices in β . That is, in case of Λ -FACTOR and (σ, ρ) -DOMSET, the degree of the vertices from β with respect to S might not be contained in Λ , σ , or ρ .

The goal of the algorithm for Π is to inductively (over the tree decomposition) build the partial solutions for the bags of the decomposition such that we eventually arrive at a valid solution. Instead of storing the partial solutions directly, we derive a “fingerprint” from the partial solutions. This fingerprint consists of the *states* of the vertices in the bag β , that is, how the vertices in β interact with the partial solution, and thus, characterizes the partial solutions.

With this basic idea, the definition of the parameterized algorithm for Π is straightforward.

1. Identify the states which are needed to encode the partial solutions, that is, the states of the vertices in a bag β . Say, for problem Π , each vertex can have c_Π different states.
2. Define a dynamic program based on the tree decomposition of the given graph and maintain a set of state-vectors that represent the feasible partial solutions.

As we can assume that the given tree decomposition is *nice* (see Section 2.2 for more details), the changes between two bags are limited. Hence, we also know the types of changes in advance and can define the program accordingly.

Combining these two observations, it seems plausible to obtain an algorithm that solves problem Π in time $(c_\Pi)^{\text{tw}} \cdot |G|^{\mathcal{O}(1)}$. However, there is a significant issue with this approach as we have to handle the join nodes of the given tree decompositions.

1.2.1 Efficient Convolution Algorithms

As one step of the algorithm, we need to combine state-vectors at the join nodes, that is, the branching vertices of the tree of the tree decomposition. For such a join node, we might be given up to $(c_\Pi)^{\text{tw}}$ state-vectors for each of the two child nodes. A naive combination of these vectors would take time $(c_\Pi)^{2\text{tw}}$ which is slower than the running time we want to achieve.

Hence, we need a different and significantly faster approach for this procedure. A first such example was the algorithm by van Rooij, Bodlaender, and Rossmanith [132]. They use fast convolution techniques to obtain a better running time for DOMINATING SET. Although their algorithm covers many problems, the formal procedures are tailored to the underlying problem and do not directly transfer to other problems. Phrased differently, even the slightest

modification to the underlying problem (or an extension) requires a repetition of the technical argument that handles the convolution at the join nodes.

Hence, to avoid such further repetitions, we state a general algorithm to compute the convolution especially in the context of algorithms based on tree decompositions. Our algorithm covers three operations that might be used to combine the states. We include cyclic addition (addition modulo some number), standard addition, and truncated addition (addition with a maximum element that is absorbing under addition; denoted by $\oplus$). Moreover, all three of these operations might appear simultaneously for different vertices. For the case of the classical addition we also provide two ways to combine the values; the first one, denoted by the domain $\mathbb{A}$ in the following, handles the addition in many dimensions with few varying maximum elements. In contrast, the addition denoted by the domain $\mathbb{L}$ covers the addition of large numbers in few dimensions.

Formally, as a first algorithmic result in our toolbox, we prove Main Theorem I.1.

Main Theorem I.1 (General Convolution). *Consider a domain $\mathbb{D}$ composed of the four subdomains $\mathbb{S}$, $\mathbb{A}$, $\mathbb{L}$, and $\mathbb{T}$ such that $\mathbb{D} := \mathbb{S} \times \mathbb{A} \times \mathbb{L} \times \mathbb{T}$*

- *with $\mathbb{S} = \mathbb{Z}_{z_1} \times \cdots \times \mathbb{Z}_{z_\sigma}$ for the first domain,*
- *with $\mathbb{A} = [0 \dots q_1) \times \cdots \times [0 \dots q_\alpha)$ for the second domain and define $A_+ := \sum_{i=1}^\alpha q_i$,*
- *with $\mathbb{L} = [0 \dots x_1) \times \cdots \times [0 \dots x_\lambda)$ for the third domain, and*
- *with $\mathbb{T} = [0 \dots y_1) \cup \{\tau\} \times \cdots \times [0 \dots y_\tau) \cup \{\tau\}$ for the fourth domain and define $T_+ := \sum_{i=1}^\tau y_i$.*

Denote by $f_1, f_2: \mathbb{D} \rightarrow \mathbb{N}$ two functions that are given as table representations.

We denote by $f: \mathbb{D} \rightarrow \mathbb{N}$ the convolution of f_1 and f_2 that satisfies, for all $(s, a, \ell, t) \in \mathbb{S} \times \mathbb{A} \times \mathbb{L} \times \mathbb{T} = \mathbb{D}$,

$$f(s, a, \ell, t) = \sum_{s_1+s_2 \equiv s} \sum_{a_1+a_2=a} \sum_{\ell_1+\ell_2=\ell} \sum_{t_1 \oplus t_2=t} f_1(s_1, a_1, \ell_1, t_1) \cdot f_2(s_2, a_2, \ell_2, t_2).$$

If U denotes a non-negative integer such that all images of f_1 , f_2 and f are contained in $[0 \dots U)$, then the function f can be computed in time

$$X \cdot \log X \cdot (\log U + \overline{M})^{\mathcal{O}(1)} + 2^\tau \cdot (\log U + \exp(\overline{M}))^{\mathcal{O}(1)}$$

with $X = |\mathbb{D}| \cdot |\mathbb{L}|^{\log(3/2)} \cdot (A_+ + T_+)^{\log(3)} \cdot 3^\lambda$ and $\overline{M} = \prod_{m \in M} m$ where $M = \{z_1, \dots, z_\sigma, q_1, \dots, q_\alpha, 3, y_1, \dots, y_\tau, 3\}$.

As a first application of this algorithm, we provide a faster algorithm for Λ -FACTOR. Together with the intuitive bound on the number of states, the fast convolution algorithm leads to a (seemingly) optimal algorithm for all covered cases. Formally, in Chapter 6 we prove Main Theorem II.1.

Main Theorem II.1. *Fix some set $\Lambda \subseteq \mathbb{N}$ that is finite or cofinite.*

Given a Λ -FACTOR instance G along with a tree decomposition of width w , we can count the solutions of size exactly s in time $(\text{top } \Lambda + 1)^w \cdot |G|^{\mathcal{O}(1)}$ for all s simultaneously.

While the bound on the number of states intuitively seems to be optimal, this is far from being a formal proof. More precisely, although a single vertex might have up to c_Π states (for Λ -FACTOR we have $c_\Lambda = \text{top } \Lambda + 1$), this does not imply that for a set of w vertices, we can actually achieve all $(c_\Pi)^w$ states. Indeed, we show that in certain cases we can do much better than this.

1.2.2 Improving the Bound on the Number of States

To show that the intuitive number of states might be far from optimal, we consider (σ, ρ) -DOMSET. In light of the first general algorithm for (σ, ρ) -DOMSET by van Rooij, Bodlaender, and Rossmanith [132], see also [130, 131], and the observations about the number of states, it seemed plausible that the known algorithm is optimal.

To understand why this bound should be optimal, we take a more detailed look and fix two finite sets σ and ρ . Then, the state of a vertex in a partial solution is determined (1) by the selection status, that is, either the vertex is selected or unselected, and (2) by the number of neighbors in the solution.

Clearly there is no solution, when a vertex is selected and has more than $\max \sigma$ selected neighbors. Hence, we use the states $\sigma_0, \sigma_1, \dots, \sigma_{\max \sigma}$ to describe the possible states for a *selected* vertex. Similarly, there is also no solution where a vertex is unselected and has more than $\max \rho$ neighbors. Hence, the possible states of an *unselected* vertex are $\rho_0, \dots, \rho_{\max \rho}$. Note that these states do not depend on the remaining values contained in σ and ρ that are not maximum.

In total, a vertex can have $(\max \sigma + 1) + (\max \rho + 1)$ states. This also coincides with the running time of the algorithm from [132] which solves (σ, ρ) -DOMSET in time $(\max \sigma + \max \rho + 2)^{\text{tw}} \cdot |G|^{\mathcal{O}(1)}$.

However, it is actually not true that all of these $(\max \sigma + \max \rho + 2)^{\text{tw}}$ states can be achieved. Indeed, we prove that in certain cases at most $(\max\{\max \sigma, \max \rho\} + 1)^{\text{tw}}$ state-vectors can appear.

While such a bound alone is not yet sufficient to prove a faster algorithm, we show that we can compress the state-vectors such that even the convolution at the join nodes can be handled efficiently. Formally, we prove Main Theorem III.1 in Chapter 9 where the constant $c_{\sigma, \rho} \leq \text{top } \sigma + \text{top } \rho + 2$ is defined in Definition 8.1.4.

Main Theorem III.1. *Fix two finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$. Given a graph G with a tree decomposition of width w and an integer s , we can count the number of (σ, ρ) -sets of size exactly s in time $(c_{\sigma, \rho})^w \cdot |G|^{\mathcal{O}(1)}$.*

1.2.3 Representative Sets

To improve the algorithms further, we need a different approach. As an illustrative example, we again consider the conceptually simpler Λ -FACTOR where Λ is a cofinite set where only a single value is missing. For example, assume that we set $\Lambda = \mathbb{N} \setminus \{404\}$. As our general algorithm is applicable in this case, we can solve the problem in time $406^{\text{tw}} \cdot |G|^{\mathcal{O}(1)}$.

However, this running time feels somewhat unsatisfying. The fact that only a single number is missing should not affect the running time to that extent. To get an intuitive understanding why this running time can be improved, we consider a bag with a single vertex v .

Our general algorithm would store up to 406 partial solutions, one for each possible state (that is, degree) of the vertex. We claim that storing *any* two distinct solutions is already enough.

Denote by S_1 and S_2 two partial solutions for the subgraph corresponding to the bag containing only v . We denote by d_1 the degree of v in S_1 and by d_2 the degree of v in S_2 , and we assume that $d_1 \neq d_2$. Otherwise, there is no point in considering these two solutions. Now pretend that we are given some extension $\bar{S}$ to the remaining graph, that is, we are given a partial solution $\bar{S}$ for the remaining graph and denote by $\bar{d}$ the degree of v in $\bar{S}$. Now, if $d_1 + \bar{d} \neq 404$, then the set $S_1 \dot{\cup} \bar{S}$ is a solution for the entire graph. Otherwise, when $d_1 + \bar{d} = 404$, then we know that $d_2 + \bar{d} \neq 404$ as S_1 and S_2 are different and more precisely $d_1 \neq d_2$. Hence, in this case $S_2 \dot{\cup} \bar{S}$ is a valid solution. As we do not assume anything about the partial solution $\bar{S}$, this proves the claim.

This concept clearly generalizes to sets Λ where more numbers are forbidden as long as we store one partial solution more than we have forbidden values. Say, the set Λ has x forbidden numbers, that is, we have $|\mathbb{N} \setminus \Lambda| = x$. Then, it suffices to store $x + 1$ partial solutions.

Additionally, we can also extend this concept to more vertices. In this case we have to store $(x + 1)^w$ partial solutions assuming that we consider a bag with w vertices. For our example where $\Lambda = \mathbb{N} \setminus \{404\}$, this could potentially improve the algorithm from a running time of $406^{tw} \cdot |G|^{\mathcal{O}(1)}$ to $2^{tw} \cdot |G|^{\mathcal{O}(1)}$. We additionally emphasize that the latter bound is independent of the precise definition of Λ but just depends on the number of missing values.

Phrased differently, instead of storing all possible partial solutions, it suffices to store a (potentially) much smaller subset thereof that represents the original set. We formalize this idea based on the concept of *representative sets*. This concept was first introduced by Monien [108] to obtain an FPT-algorithm for k -PATH. Afterward, this concept was used multiple times to obtain faster algorithm for problems parameterized by treewidth [1, 13, 14, 43, 60, 61, 93, 102, 122].

We use this approach based on representative sets to obtain faster algorithms for Λ -FACTOR and (σ, ρ) -DOMSET when the underlying sets are cofinite. To state these two algorithms formally, we provide a second auxiliary result for improved upper bounds. While the general concept of representative sets is similar for all applications, there are usually some minor differences that require new algorithms. In Section 3.2 we present two algorithms that compute representative sets when vertices have to satisfy particular degree constraints or rather certain degrees should be avoided. We state the algorithm for a more general variant of representative sets which allows for applications in different contexts.

For an intuitive understanding of k - r -representation, denoted by $\subseteq_{r\text{-rep}}^k$, we consider k vertices and each vertex is “robust” against r forbidden values (see Definition 3.2.3 for the formal definition).

Main Theorem 1.2. *Consider k integer universes $U_1, \dots, U_k$ of size n each. Given a set $S \subseteq U_1 \times \dots \times U_k$, we can compute a set $S' \subseteq_{r\text{-rep}}^k S$ of size $|S'| \leq (r+1)^{\mathcal{O}(k)}$ in time $|S| \cdot (r+1)^{\mathcal{O}(k)} \cdot k$.*

We observe that the computation of the representative sets is slower than the running time we want to achieve. This raises the natural question regarding faster algorithms. However, as

already indicated above, we also have to deal with join nodes in this context. Although we could technically apply the convolution techniques from Section 3.1, they do not compute a representative set. Instead, currently the only way to deal with join nodes in the context of representative sets is the naive combination of the states and then, the computation of a representative set thereof. Therefore, besides a faster algorithm for the general case, it remains open as well to find a better algorithm to handle the join nodes.

Together with the computation of the representative sets, we can easily define the improved algorithms for Λ -FACTOR when the set Λ is cofinite. The idea is similar to the general algorithm except that we only store a representative set of the partial solutions.

Main Theorem II.2. *Fix some integer $x \geq 1$. Given an ANTIFACTOR_x instance G along with a tree decomposition of width w , we can decide if there is a solution of size exactly s in time $(x+1)^{\mathcal{O}(w)} \cdot |G|^{\mathcal{O}(1)}$ for all s simultaneously.*

Similarly, we also consider (σ, ρ) -DOMSET and provide a potentially faster algorithm when one set is cofinite. Recall that we use the notion of the cost of a set, to obtain a unified running time of the algorithm.

Definition 8.1.7 (Cost of a Set). *Consider a set $\tau \subseteq \mathbb{N}$ that is either finite or cofinite. Based on the set τ , we define the number $\text{cost } \tau$ as*

$$\text{cost } \tau := \begin{cases} \max \tau & \text{if } \tau \text{ is finite,} \\ |\mathbb{N} \setminus \tau| & \text{if } \tau \text{ is cofinite.} \end{cases}$$

With this notation, we can formally state the improved algorithm for (σ, ρ) -DOMSET as follows.

Main Theorem III.2. *Fix two finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$. Based on these sets we define $t_{\text{cost}} := \max\{\text{cost } \sigma, \text{cost } \rho\}$.*

Given a graph G with a tree decomposition of width w , and an integer s , we can decide whether G has a (σ, ρ) -set of size exactly s in time

$$2^w \cdot (t_{\text{cost}} + 1)^{w \cdot (\omega+1)} \cdot |G|^{\mathcal{O}(1)}.$$

We crucially observe that both algorithms from Main Theorems II.2 and III.2 do *not* solve the counting versions of the respective problem. This is actually not a shortcoming of our particular algorithm but inherent to the concept of representative sets. As we only store a reduced amount of information, we cannot solve the counting version anymore as this variant requires all information to be accessible; each individual solution might contribute to the output. Furthermore, we prove lower bounds that indicate that other methods do not yield improvements as well. Thus, the application of representative sets is not decisive for the limitations of the algorithms.

1.3 Technical Overview for the Lower Bound

Much more than the algorithmic results, the (conditional) lower bounds parameterized by treewidth are highly problem-specific tasks. Indeed, there are many steps where problem-specific tools are needed. However, most of these (tight) lower bounds are based on the

STRONG EXPONENTIAL TIME HYPOTHESIS (SETH). Hence, they follow a natural theme that was introduced by Lokshtanov, Marx, and Saurabh [97] for the conditional lower bounds under SETH for DOMINATING SET and INDEPENDENT SET parameterized by treewidth. See [16, 35, 37, 83] for more examples of similar hardness results.

Usually the algorithmic results can be stated for the smaller parameter treewidth but still achieve the same running time as for the larger parameter pathwidth. Hence, the approach to show matching lower bounds, follows the exactly opposite approach. That is, the lower bound is stated for the larger parameter pathwidth and thus, follows for the smaller parameter treewidth. For most problems this approach can be used to show that the complexity does not change between these two parameters. In general however, this is *not* true and the complexity of the problem might change between pathwidth and treewidth [7, 95]. Phrased differently, although the problem might be easy (that is, FPT) parameterized by pathwidth, the problem could be much harder (for example, $W[1]$ -hard) parameterized by treewidth.

As this behavior seems to occur only in few cases—which makes it actually more interesting to identify the affected problems and the reasons for this unexpected behavior—and since this change does not apply to the problems we consider, we prove the lower bounds for the parameter pathwidth.

1.3.1 Classical Approach

Before we outline our proofs of the conditional lower bounds for Λ -FACTOR and (σ, ρ) -DOM-SET, it is instructive to understand the classical approach initiated by Lokshtanov, Marx, and Saurabh [97] to obtain tight lower bounds.

For this we fix some problem Π for which some notion of pathwidth is defined. Assume for simplicity that Π is a graph problem. Suppose we want to show that given a graph G (as an instance of Π) along with a path decomposition of width w , no algorithm can solve Π in time $(c_\Pi - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$ for any $\varepsilon > 0$, unless SETH fails.

The main ingredient of the proof is a reduction from k -SAT to Π . Assume for this that we are given a k -SAT formula ϕ with variables $V(\phi)$. Then, we want to construct an equivalent instance I_ϕ of Π that has pathwidth approximately $|V(\phi)|/\log c_\Pi + \mathcal{O}(1)$ and size $|\phi|^{\mathcal{O}(1)}$. We construct this instance with the following four steps.

1. Group the variables of ϕ into sets of size $\log c_\Pi$. That is, we can encode the $2^{\log c_\Pi}$ assignments for one group by the c_Π possible states of a single vertex.
2. Construct a graph of pathwidth $|V(\phi)|/\log c_\Pi$ such that we can encode the assignments to the variables by the states of the vertices. Moreover, the construction needs to ensure that we can transfer this encoding through the entire graph.
3. Encode the clauses of the SAT-formula ϕ by gadgets which check that the encoded assignment satisfies the clauses and thus, the entire formula.
4. Patch the graph such that the encoded assignment is transmitted consistently (in some sufficiently large part of the graph).

We ignore rounding issues for a moment. The formal proofs use a smaller number of larger groups of variables where each group corresponds to multiple vertices.

Together with its generality, this approach comes with some weaknesses, that make the proofs more complicated, less elegant, and prohibit an extension to the counting version. For each of the previous steps, we identify one such obstacle.

Constraint Satisfaction Problems. First, although it seems comparably easy to group the variables when $c_\Pi > 2$, rounding issues (and sometimes also parity issues) make the procedure much more complicated. Indeed, as the grouping affects the entire constructions, multiple further steps get obfuscated because of this.

Although this technique of grouping the variables appears in multiple lower bounds (see [16, 37, 83, 97], for example) and is phrased there implicitly, Lampis [94] seems to be the first to make this construction explicit. The conceptual idea is pretty simple. Instead of reducing from k -SAT, the reduction starts from a constraint satisfaction problem (CSP) where every variable can take up to B values and the constraints involve at most q variables; we refer to this problem as q -CSP- B in the following. We note that k -SAT is a special case of k -CSP-2.

When we reduce from q -CSP- c_Π , we can directly establish a one-to-one correspondence between the assignments for the variable and the states of the vertices, that is, no grouping or similar techniques are needed, and the construction is simplified. As the construction was only implicitly presented in earlier constructions, Lampis also provides an explicit conditional lower bound for q -CSP- B under SETH which intuitively states that there is no algorithm that solves the problem in time $(B - \varepsilon)^n \cdot |\phi|^{\mathcal{O}(1)}$ for an instance ϕ with n variables (see [94, Theorem 3.1] and Conjecture 4.1.7).

Pathwidth-SETH. To prove tight lower bounds, we usually base the results on SETH which intuitively implies that there is no algorithm that can solve k -SAT in time $(2 - \varepsilon)^n \cdot n^{\mathcal{O}(1)}$ for a formula with n variables [80, 81].

Because of its strong implication, this conjecture is not unanimously agreed on. Indeed, we follow an argument introduced in previous work [96] to show that a different hypothesis is more appropriate to prove tight lower bounds. To understand this argument for choosing a different starting point, we consider an example. Assume we are given the formula

$$\begin{aligned} \phi = & (x_1 \vee \neg x_2 \vee x_3) \wedge (x_3 \vee x_4 \vee x_5) \wedge (x_5 \vee x_6 \vee \neg x_7) \\ & \wedge (x_1 \vee x_2 \vee x_4) \wedge (\neg x_3 \vee \neg x_4 \vee \neg x_6) \wedge (\neg x_5 \vee \neg x_6 \vee x_8). \end{aligned} \quad (1.3.1)$$

Whenever we fix the assignment for the two variables x_{2i-1} and x_{2i} , the simplified formula ϕ decomposes into two parts that do not share any variables, that is, we can handle each of these parts individually.

We formalize this intuition based on the *primal graph* of ϕ . The primal graph consists of one vertex for each variable and an edge uv whenever the variables u and v appear together in some clause of ϕ . Then, we can use the large variety of connectivity measures for graphs and apply them to the primal graph such that they transfer to the formula. We refer to Figure 1.1 for an illustration of the primal graph of the formula from Equation (1.3.1).

Iwata and Yoshida [82] combined these known concepts in the context of parameterized complexity and established lower bounds based on the treewidth of the primal graph. Lampis [96] focused on the *pathwidth* of the primal graph instead, which allowed for a significant

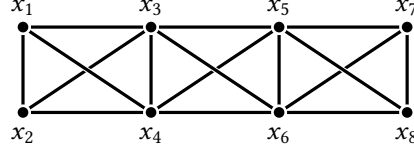


Figure 1.1: The primal graph of the formula ϕ in Equation (1.3.1), that is, we have a vertex for each variable and the vertices of two variables are connected if they appear in the same clause.

extension of the previous results by Iwata and Yoshida [82]. More precisely, Lampis defines the *PRIMAL PATHWIDTH STRONG EXPONENTIAL TIME HYPOTHESIS* (pw-SETH) which intuitively states that there is no algorithm that solves k -SAT for a formula ϕ which is given with a path decomposition of the primal graph of ϕ of width w in time $(2 - \varepsilon)^w \cdot |\phi|^{O(1)}$ (see [96, Conjecture 1.1] and Conjecture 4.1.4 for the formal statement). In the context of our previous example, it is straightforward to show that the pathwidth of the graph is at most 3 which agrees with our intuition of constant-sized separators.

Clearly, the classical SETH implies pw-SETH as the pathwidth is always bounded by the number of variables, and we can find a trivial path decomposition of this width. Thus, proving the lower bound under the weaker pw-SETH makes the lower bound results stronger, that is, more plausible.

Moreover, Lampis proves that this conjecture does not only *imply* certain lower bounds parameterized by pathwidth (for example, for DOMINATING SET), but the new hypothesis is actually *equivalent* to multiple of these lower bounds (see [96, Lemmas 5.7 and 5.8] for a formal statement). That is, pw-SETH and the conditional lower bounds form one equivalence class. Phrased differently, we already know that a faster algorithm for DOMINATING SET parameterized by pathwidth implies a faster algorithm for SAT parameterized by pathwidth. Additionally, Lampis [96] proves that a faster algorithm for SAT parameterized by pathwidth implies a faster algorithm for DOMINATING SET with the same parameterization. Thus, a faster algorithm for a single problem of this equivalence class would automatically result in faster algorithm for the other problems as well; the lower bounds are more reasonable to hold true. This behavior is different from the classical reductions which focus on one direction only.

With this new hypothesis, we revisit the CSP from above. It is natural to ask for a lower bound for q -CSP- B under pw-SETH. Lampis [96] answered this question already and provided precisely this conditional lower bound for q -CSP- B under pw-SETH (see [96, Theorem 3.2] and Conjecture 4.1.8) such that we can easily construct the lower bounds under pw-SETH.

Relations. While the previous two steps make it comparably easy to encode the assignments to the variables, it remains open to encode the clauses or rather the constraints in the context of the CSP. At a first glance one might think that clauses are easier to achieve than constraints, and the usage of the q -CSP- B problem might not be that much of a help. And indeed, the clauses are more structured as they are formed from few operators only (that is, “negation” and “logical or”).

However, recall that we usually need to group variables to achieve tight lower bounds. Therefore, the implementation of the clauses is again more complex as the gadgets have to

observe a group of vertices and then infer the assignment of a single variable. In the case of constraints, we have direct access to the assignment via the state of a single vertex, but checking whether the constraint is satisfied is more complex.

We conclude that it is actually indifferent whether we encode the simple clauses based on groups of variables or whether we encode the complex constraints based on single variables. In both cases, we need to design gadgets which might behave in a complex way.

To realize these complex gadgets, Curticapean and Marx [35] introduced a systematic approach to implement the constraints. In a first step they use arbitrary “relations” which essentially model the constraints in the graph setting. Then, in a second step, they replace these relations by simpler structures that are much easier to design. Moreover, these substructures are independent of the precise instance at hand and are essentially identical for *all* problems.

Phrased differently, Curticapean and Marx [35] show that an arbitrary complex relation can be broken down to two simple relations. The first one is “Hamming weight exactly 1” and requires that exactly one of the observed objects is selected. The second structure is “Equality” which requires that all or none of the observed objects are selected. We abbreviate the former by $\text{HW}_{=1}$ and the latter by EQ.

For example, when we select edges, then $\text{HW}_{=1}$ requires that exactly one of the incident edges is selected while EQ ensures that either all or none of the incident edges are selected.

With the observation that we can replace arbitrary complex constraints by $\text{HW}_{=1}$ and EQ, the implementation of the constraints of the CSP instance becomes much more approachable, that is, more or less it remains to construct gadgets only for these two cases.

Monotonicity. For the last step of our analysis of the existing constructions, we focus again on the encoding of the variable assignment. In most constructions it can actually happen that the variable assignment is *not* encoded consistently, that is, the assignment might be different for different clauses/constraints. In worst case we could have a different assignment for each variable in every constraint such that all constraints are satisfied.

To understand what we mean by *consistency* and why these inconsistencies appear, we consider the lower bound for INDEPENDENT SET by Lokshtanov, Marx, and Saurabh [97]. For each variable there is a path of length $2m$ assuming that m denotes the number of clauses. Then, selecting the vertices at odd positions encodes the value false, and selecting the vertices from the even positions encodes the value true. Clearly we can pick one of the independent sets where we either select all vertices at the even positions or all vertices at the odd positions. However, there is also another possibility to select vertices, that is, we might select the first i vertices from odd positions and then select $m - i$ vertices from the remaining even positions. Phrased differently, the encoded assignment of the variable might change at some point from false to true, that is, the assignment is not encoded consistently anymore.

A first idea to resolve this issue might be to connect the end of the path to the start. While this would indeed solve the consistency problem, the pathwidth of the final construction increases significantly. More precisely, the pathwidth would double which makes this approach insufficient to prove a tight lower bound.

As this problem appears for most of the considered problems, and the naive approach does not solve the problem, we need a general idea instead.

To establish the consistency of the variables, we observe that the changes happen to be monotone, that is, the encoded value of a variable is either non-decreasing or non-increasing. Although this behavior is non-trivial to see, it holds for the known constructions. As each variable can take at most B values, assuming we reduce from q -CSP- B , and there are n variables, we observe that there can be at most $n \cdot B$ such changes in a single instance.

Hence, we can repeat the entire construction $n \cdot B + 1$ times, that is, we introduce $n \cdot B + 1$ copies of each constraint, and concatenate the copies of the construction to form *one* new instance. By pigeonhole-principle there must be one repetition where no change happens. As we arrange the copies such that this repetition contains all constraints, we know that if there is a solution, then all constraints are satisfied by a consistent assignment and thus, the formula is satisfiable.

While this procedure works for the decision version, it potentially introduces many new solutions that do not correspond to assignments of the original CSP instance. Therefore, the construction does *not* work for the counting version of the problem considered.

Instead of repeating the entire construction, there are other possibilities to ensure that the assignment is consistent in the entire instance. For example, the construction in [104] ensures that all encodings of the assignments have a specific weight which is known upfront. Then, by the use of additional gadgets we can check that this weight is always preserved and by the monotonicity of the states, no change is allowed.

Note that this thesis is based on [104], but we use an improved construction instead.

A similar approach was taken by Lampis [96] based on monotone *multi-assignments*. To understand this variant, we consider the pathwidth variant of q -CSP- B , that is, we are given a path decomposition of the primal graph of the CSP-instance. Then, each variable might receive a different assignment in each bag. As this would clearly make the instance satisfiable, Lampis requires that the satisfying multi-assignments are monotone, that is, the values of each variable might only increase along the path decomposition. To establish the equivalence to the original q -CSP- B instance (where all variables must have a consistent assignment) Lampis additionally allows few variables (that is, a logarithmic number) that must receive a consistent assignment. Then, these consistent variables and additional constraints ensure that the assignment of the monotone variables is consistent too.

While it might seem unintuitive, to distinguish between the consistent variables and the monotone variables which are consistent in every satisfying assignment, the catch is that the construction only has to ensure that the monotone variables receive a monotone assignment. Then, the consistency follows implicitly from the satisfied constraints.

As the number of consistent variables is much smaller than the total number of variables, we can use the naive approach to ensure the consistency for these variables, that is, we can connect the start and the end of the corresponding paths.

With this analysis of the existing techniques and constructions, and the ideas to simplify and modularize the reductions, we move to our approach to provide tight lower bounds.

1.3.2 Our Approach

We split the proofs of the lower bound results into four major steps. Assume we want to prove a lower bound for the problem Π parameterized by pathwidth assuming pw-SETH. Instead of

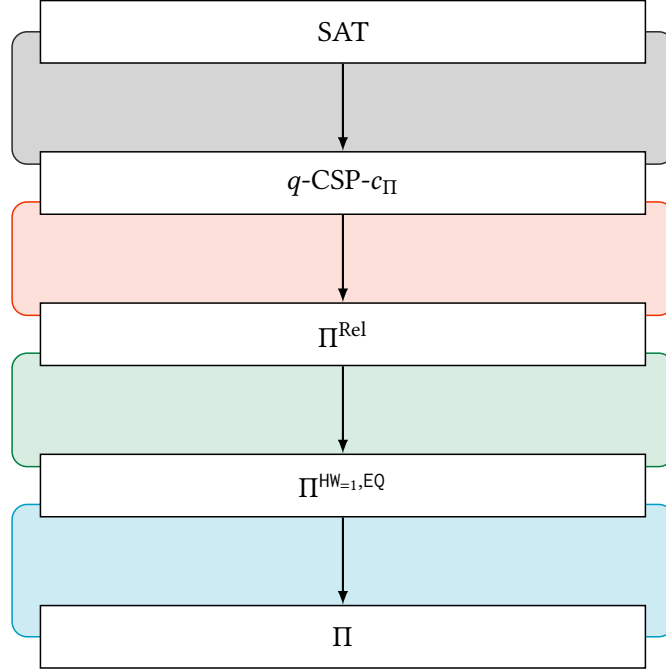


Figure 1.2: General outline of the reductions to prove a lower bound for the problem Π under pw-SETH.

proving a direct reduction, we propose the following general approach. We refer to Figure 1.2 for a symbolic overview of this outline.

Step 0: We introduce a conjecture regarding a specific variant of q -CSP- B and provide evidence for this lower bound. We define this variant to simplify the later constructions and hide unwanted technicalities, that is, we can pick an appropriate base of the running time, the conjecture is implied by pw-SETH, the problem is robust against monotone assignments, and we state a corresponding result for the counting variant.

We refer to this conjecture as **MONO-pw-CSPH** (and **MONO-pw-#CSPH** for the counting variant).

Step 1: With **MONO-pw-CSPH**, we need to prove a reduction from q -CSP- B to Π .

For this reduction, we introduce the intermediate problem Π^{Rel} which is a generalization of Π where we additionally allow relations. That is, in the context of graph problems the relations are special types of vertices that are connected to the remaining graph and impose restrictions on the selection status of the adjacent vertices or incident edges.

We provide a lower bound for Π^{Rel} under **MONO-pw-CSPH**. We use the states to encode the assignment of the variables. As we have a one-to-one correspondence between the assignments and the states, we can directly use the relations to implement the constraints of the CSP-instance.

Step 2: We replace the arbitrary relations of the resulting Π^{Rel} -instance by gadgets that only involve $\text{HW}_{=1}$ and EQ following a generic construction.

Step 3: As a last step, we replace these two types of relations by appropriate gadgets or simulate them with suitable Turing-reductions in the case of the counting version.

We define each of these reductions such that the pathwidth increase at most by a constant that depends on Π only. Hence, the final lower bound for Π parameterized by pathwidth follows directly from the combination of the previous steps.

In the following, we outline these four steps in more detail and explain the main ideas of the constructions.

Step 0: Appropriate Starting Point. As a first independent result, we extend the results by Lampis about q -CSP- B and pw-SETH from [94, 96] to the counting version. That is, we consider the counting variant of q -CSP- B with multi-assignments and prove a lower bound under the counting version of the pw-SETH. As we provide a parsimonious reduction, the result follows for the decision version as well.

Parsimonious reductions are many-one reductions that do not change the number of solutions.

Formally, we define the MONO-pw-CSPH as follows and use it as a starting point for our lower bound reductions. The counting variant of MONO-pw-CSPH is defined analogously (see Conjecture 4.1.12 for the formal statement).

Conjecture 4.1.11 (MONO-pw-CSPH). *For all $B \geq 2$ and $\varepsilon > 0$, there exists a q such that no algorithm can, for a q -CSP- B instance ϕ that is given with*

- *a partition $V_1 \dot{\cup} V_2$ of the variables,*
- *a nice path decomposition of the primal graph of ϕ of width w*
- *where every bag contains at most $\mathcal{O}(B \cdot \log|V_1|)$ variables from V_2 ,*
- *an injective mapping c from the constraints of ϕ to the bags such that the bag contains the variables of the respective constraint,*
- *the promise that every satisfying multi-assignment that is consistent on V_2 is also consistent on V_1 ,*

decide if there is a monotone satisfying multi-assignment that is consistent on V_2 in time

$$(B - \varepsilon)^w \cdot |\phi|^{\mathcal{O}(1)}.$$

Additionally, we prove that pw-SETH implies MONO-pw-CSPH which extends the lower bounds to the better known (and easier to understand) pw-SETH.

Main Theorem I.3. *If pw-SETH holds, then MONO-pw-CSPH holds.*

Moreover, if pw-#SETH holds, then MONO-pw-#CSPH holds.

Both results hold even if all constraints have a fixed parity.

With these preliminaries, we can move toward the lower bound reduction for Π .

Step 1: The Intermediate Lower Bound. For the lower bound of Π^{Rel} we focus on the encoding of the assignments and how this information is propagated through the entire graph. Once we established a correspondence between the assignments and the states of the vertices, we can implement the constraints almost directly by the use of additional relations.

Assume we are given a $q\text{-CSP-}c_\Pi$ instance ϕ together with a path decomposition of the primal graph of width w_ϕ . Moreover, suppose that the instance ϕ satisfies the additional assumptions from MONO-pw-CSPH as stated in Conjecture 4.1.11. We refer to Figure 1.3 for an illustration of the following construction.

For each bag $\beta(j)$ of the path decomposition and each variable $v \in \beta(j)$, we create a vertex t_v^j , to which we refer as *information vertex*.

Additionally, for each bag $\beta(j)$, we create a so-called *manager* M^j that observes the vertices t_v^j for all $v \in \beta(j)$. For each observed vertex t_v^j , the manager M^j has two blocks B_v^j and $\bar{B}_v^j$. We design these blocks such that vertex t_v^j can receive c_Π states from the left block B_v^j . That is, the block B_v^j can provide t_v^j with an (almost) arbitrary number of selected neighbors or edges.

Then, the state of vertex t_v^j with respect to B_v^j determines the assignment of variable v in bag $\beta(j)$. To make this formal, we define a bijection between the c_Π different values of a variable and the c_Π possible states of a vertex in a (partial) solution.

As we eventually want to construct a solution, vertex t_v^j must have a valid state. This is precisely what we use the right block $\bar{B}_v^j$ for. Intuitively, this block complements the state from B_v^j such that both states together provide a valid state to vertex t_v^j .

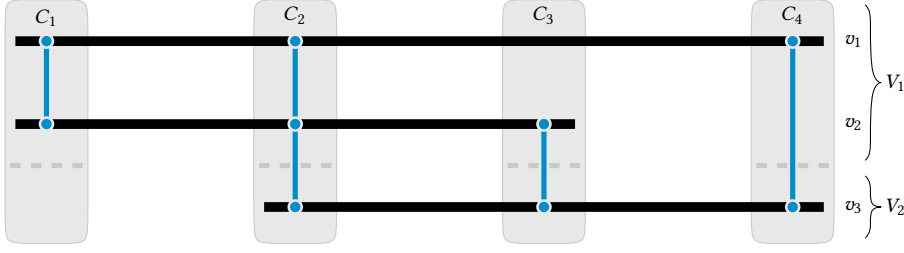
Because of parity issues these blocks are not possible in general. Thus, we allow few edges (that is, the number only depends on Π) between the blocks. These connections must be such that left blocks are only connected to left block and right blocks are only connected to right blocks except for a single pair of left and right blocks that have edges from the left to the right.

So far, this construction can be used to encode the multi-assignment to the variables. As a next step, we ensure that the assignment is consistent. For this we focus on a variable v that appears in the bags $\beta(j)$ and $\beta(j+1)$. We create a relation $\bar{R}$ that ensures that the state from the left block B_v^{j+1} is the complement of the state of the right block $\bar{B}_v^j$. When we define the mappings between states and assignments appropriately, this relation $\bar{R}$ together with the behavior of the information vertices ensures that the assignment for v in bag $\beta(j+1)$ is not smaller than the assignment for v in bag $\beta(j)$, that is, the assignment increases monotonically.

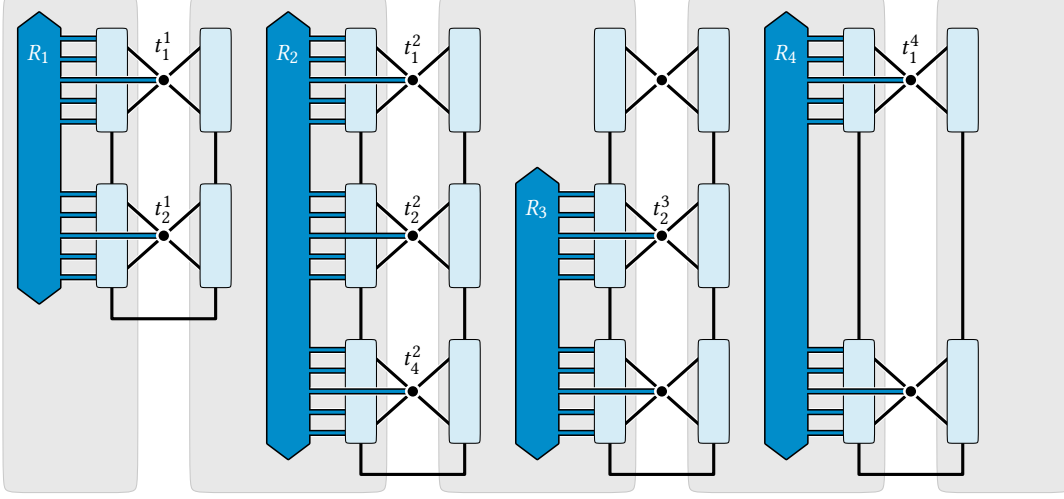
To ensure that the variables from V_2 receive a consistent assignment, we consider a variable $v \in V_2$ that is introduced at node $\text{intro}(v)$ and forgotten at node $\text{forget}(v)$. We add a relation $\bar{R}$ between the right block $\bar{B}_v^{\text{forget}(v)-1}$ and the left block $B_v^{\text{intro}(v)}$. By the monotonicity of the assignment, it follows that the assignment must be identical for all bags. Note that, for each variable from V_2 in the current bag, this increases the pathwidth by a constant.

It remains to encode the constraints. For each constraint C_j of the CSP-instance ϕ , we create a relation R_j . For each variable v in the scope of C_j , this relation observes the block B_v^j and the vertex t_v^j to infer the state of the information vertex with respect to the block B_v^j . As this state encodes the assignment of the variable, the relation R_j has access to the assignment and can check whether the constraint is satisfied.

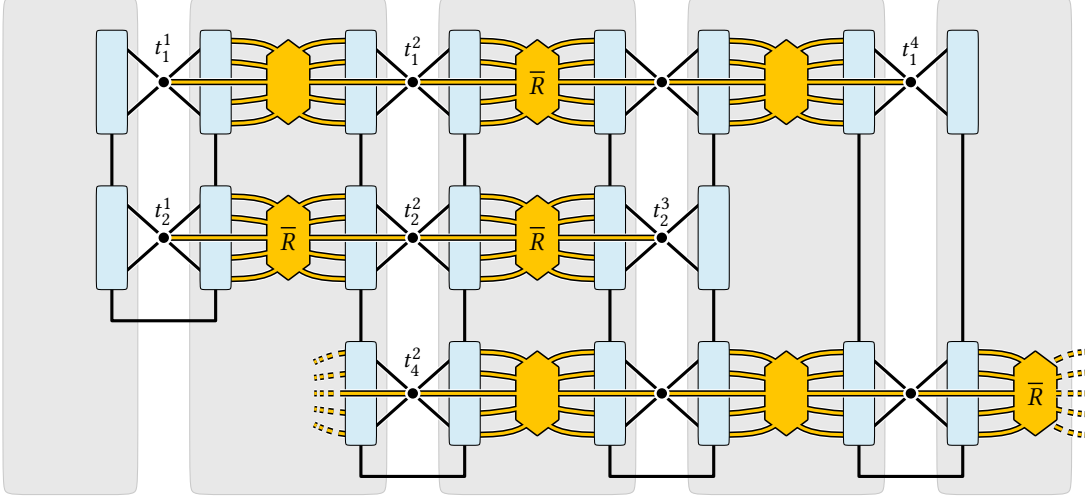
To conclude the lower bound for the intermediate problem Π^{Rel} , we observe that each relation observes only a constant number of objects. Indeed, the number of states is bounded



(a) A path decomposition of the primal graph of a given CSP-instance. The horizontal lines represent the three variables and the blue vertical lines the constraints.



(b) The instance of Π^{Rel} contains relations R_j that correspond to the constraints C_j of ϕ . The small light blue boxes represent the blocks of the managers that provide a state to the information vertices t_v^j .



(c) Additionally, the instance of Π^{Rel} contains relations that invert the state of the blocks. We depict these relations $\bar{R}$ as yellow hexagons.

Figure 1.3: Idea of the reduction from $q\text{-CSP-}\Pi$ to the intermediate problem Π^{Rel} . Based on the path decomposition for the CSP-instance ϕ shown on the top, we construct the instance of Π^{Rel} shown below.

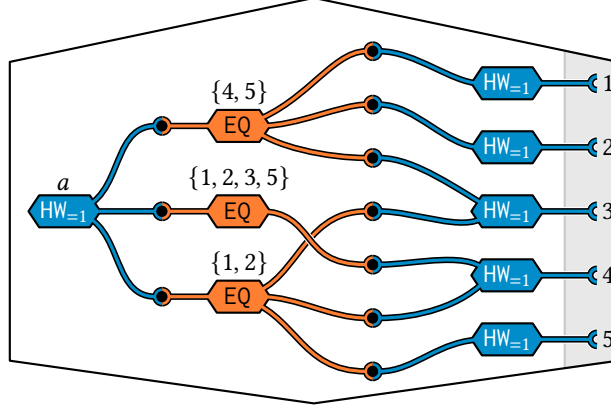


Figure 1.4: Conceptual approach to realize an arbitrary relation. Here shown for the relation $R = \{\{1, 2\}, \{1, 2, 3, 5\}, \{4, 5\}\}$ where we label the portal objects from top to bottom.

by c_Π and the arity of the constraints is bounded by q . Hence, the pathwidth of the new instance is determined by the pathwidth of the primal graph plus the number of vertices from V_2 in each bag. As the latter number is logarithmic in the total number of vertices, this does not increase the pathwidth significantly.

This concludes the idea of the conditional lower bound for the intermediate problem Π^{Rel} . It remains to remove the relations we just introduced.

Step 2: Reducing the Number of Different Relations. In the remaining steps of the lower bound, we remove the arbitrary relations that we introduced in the previous step. First, we replace arbitrary relations by gadgets that use $\text{HW}_{=1}$ and EQ only. We denote the resulting variant of Π^{Rel} by $\Pi^{\text{HW}_{=1}, \text{EQ}}$ to indicate that we use only these two types of relations. Formally, in this step we provide a reduction from Π^{Rel} to $\Pi^{\text{HW}_{=1}, \text{EQ}}$.

The reduction is based on a construction by Curticapean and Marx [35] with the following idea. We refer to Figure 1.4 for an illustration thereof.

We create one $\text{HW}_{=1}$ -vertex a which is connected to multiple EQ -vertices, each corresponding to one accepted input of the relation. Intuitively, this single $\text{HW}_{=1}$ -vertex a “activates” one EQ -vertex, namely the one corresponding to the current input.

Moreover, for each observed object (either a vertex or an edge) there is a $\text{HW}_{=1}$ -vertex. This $\text{HW}_{=1}$ -vertex is connected to those EQ -vertices where the observed object is not selected (that is, the complement of the accepted inputs). Hence, the EQ -vertices together with the $\text{HW}_{=1}$ -vertices ensure that precisely those portal objects are selected that encode a valid input.

Step 3: Realizing the Remaining Relations. With the result from the previous step, it remains to realize $\text{HW}_{=1}$ and EQ . That is, we need to reduce from $\Pi^{\text{HW}_{=1}, \text{EQ}}$ to the problem Π for which we want to establish the lower bound.

In contrast to the previous step which follows a general approach, the constructions that realize $\text{HW}_{=1}$ and EQ need to exploit the hardness of the problem Π . While the relations of higher arity can be constructed inductively, the smaller ones need to be constructed by hand.

We observe that the setting is even more complex for the counting variant. Even though it might be easy to find *some* solution for a problem, this does not mean that it is easy to count the number of all possible solutions. Additionally, for the counting version we also have to deal with those cases that are trivially solvable (for example, when the empty set is a solution). In these cases it is impossible to design an appropriate gadget that simulates a relation.

Instead, we exploit that the counting variant is more expressive and hence, we can also combine the results of multiple instances. That is, we frequently use Turing-reductions that create slightly modified instances and then combine the results to recover the answer for the original instance. Though we could also do this for the decision version, we would only deal with true and false as values which does not provide much insight.

A main technique that uses Turing-reductions is polynomial interpolation (or even multivariate interpolation). Valiant [129] introduced this technique in the context of reductions between #P-complete problems. The idea is comparably easy and reads as follows.

Consider some counting problem Π . Denote by I an instance of a variant of Π where the edges of some set F must be selected in any solution. Pretend that, for an arbitrary parameter ℓ , we have access to a gadget H_ℓ that could replace a single edge and has A_ℓ extensions if the edge is selected and B_ℓ extensions if the edge is not selected. Based on the instance I we construct a new instance I_ℓ of Π where we replace each edge in F by this gadget H_ℓ .

As a next step, we relate the number of solutions for the instance I and the number of solutions for instance I_ℓ . To this extent, denote by a_i the number of solutions for I where exactly i of the edges from F are selected. This especially means that $a_{|F|}$ denotes the number of solutions for I , that is, we have $a_{|F|} = \#\text{Sol}(I)$, but the solutions that contribute to a_i if $i \neq |F|$ do not correspond to a valid solution for I .

With this notation we get that

$$\#\text{Sol}(I_\ell) = \sum_{i=0}^{|F|} a_i \cdot A_\ell^i \cdot B_\ell^{|F|-i}.$$

Hence, we cannot directly recover the value of $a_{|F|}$ from a single oracle-call as the value $\#\text{Sol}(I_\ell)$ also covers solutions where edges from F are not selected. However, we have

$$\#\text{Sol}(I_\ell) = B_\ell^{|F|} \cdot \sum_{i=0}^{|F|} a_i \cdot \left(\frac{A_\ell}{B_\ell}\right)^i$$

and thus, we can treat $\#\text{Sol}(I_\ell)/B_\ell^{|F|}$ as the evaluation of an unknown polynomial p at value A_ℓ/B_ℓ . Now, we can use the oracle-access to evaluate p at $|F| + 1$ distinct values of ℓ based on the instances I_ℓ for Π . As we know the values for B_ℓ , we can use polynomial interpolation to recover the polynomial p and thus, recover the value $a_{|F|}$.

We also need that the fractions A_ℓ/B_ℓ are different for different ℓ .

This approach serves as a basis for many of the reductions on our way to simulate $\text{HW}_{=1}$ and EQ for the counting version. Other concepts include weights by which we can distinguish “small” from “large” solutions or where, with the help of negative weights, we can create solutions that cancel out other solutions. These weights are then again replaced via Turing-reductions that use polynomial or multivariate interpolation.

1.4 Organization

This thesis is split into three parts corresponding to our toolkit, GENFAC, and (σ, ρ) -DOMSET. We refer to Figure 1.5 for an overview of the structure.

Part I: We present general results and concepts that we use later. In Chapter 2 we first present some general notation. Then, in Chapter 3 we formalize the algorithms to efficiently compute convolutions and representative sets. In Chapter 4 we formalize the concept of the CSP and the half-induced matchings.

Part II: We consider the problem of GENFAC. After a formal introduction to the problem in Chapter 5, we first prove the algorithmic results in Chapter 6 and then prove the lower bounds in Chapter 7.

Part III: We consider the problem of (σ, ρ) -DOMSET and follow the basic outline of the case study for GENFAC. That is, after a formal introduction in Chapter 8, we prove the algorithmic results in Chapter 9 and the hardness results in Chapter 10.

Finally, we conclude in Chapter 11 and provide an outlook with open and related questions.

A Remark on Numbering and Reading. The content of each section of this thesis, for example, the content of Section 10.3, is quite self-contained. Hence, skipping such a section while reading is usually possible without significant complication. Important results of earlier sections are usually restated whenever they are needed.

Goal of most sections is to prove a single *Main Lemma*. We combine these intermediate results to prove a *Theorem* or a *Main Theorem*. The latter ones summarize the important results of each part of this thesis. Thus, we number these results based on the part in which they are proven.

As the results from separate chapters are usually independent of each other, we additionally number the results (that is, lemmas, definitions, ...) based on the section in which they are proven (or stated for definitions and conjectures). For example, Main Lemma 10.3.7 is proven in Section 10.3 which can be found in Chapter 10.

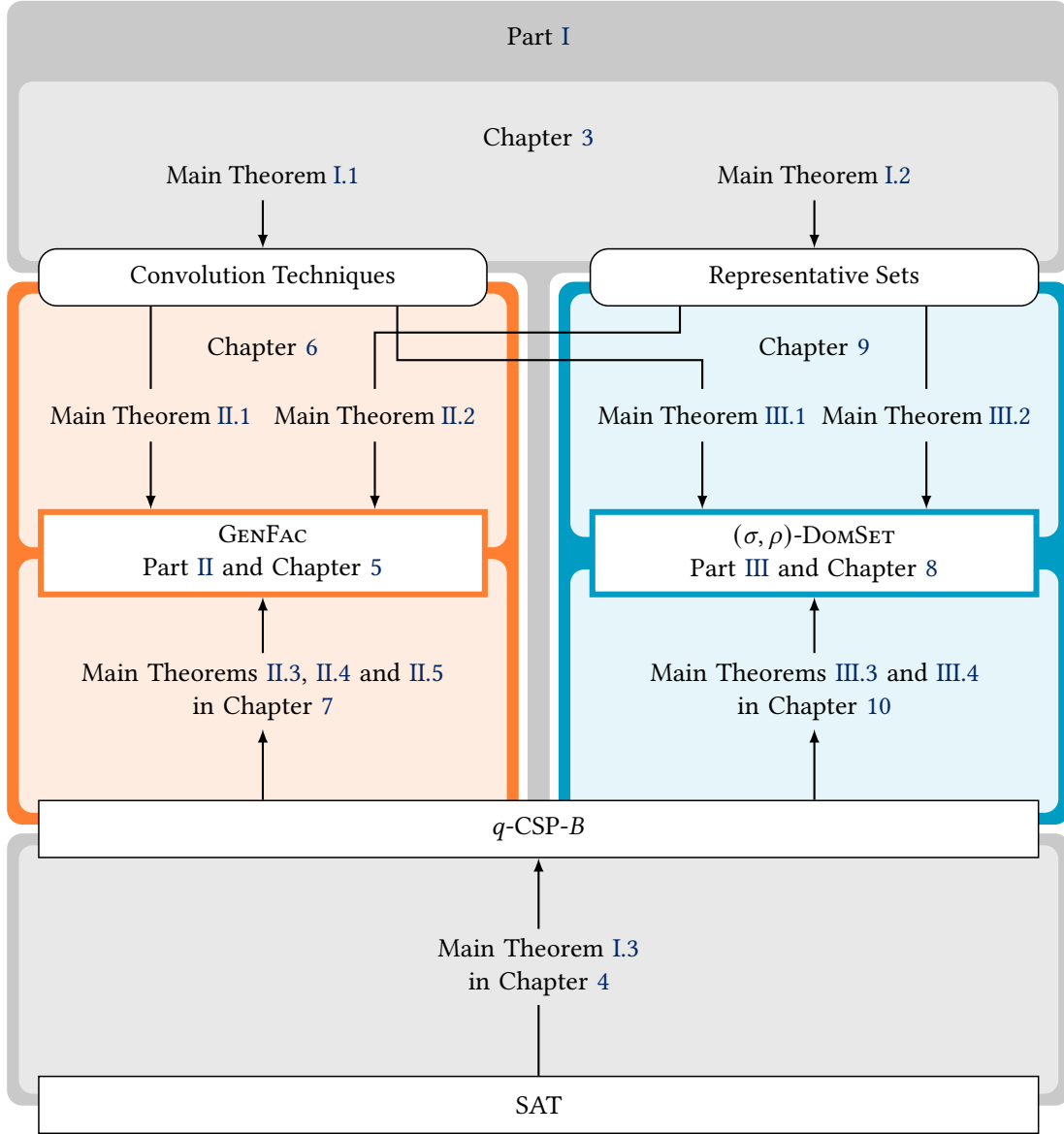


Figure 1.5: Visual representation of how this thesis is organized.

The (gray) boxes at the top and the bottom cover our tools from Part I to prove upper and lower bounds. We use these techniques to prove the results for GENFAC in Part II as visualized on the left (orange) side, and to prove the results for (σ, ρ) -DOMSET in Part III as indicated on the right (blue) side.

Part I

Our Toolkit

Preliminaries

In the following we introduce some notation and concepts that we use frequently in the thesis. More problem-specific notation and concepts are defined when they are used for the first time in the respective sections. We also refer to page 417 for a list of the notation we use.

2.1 General Notation

Numbers and Sets. We write $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ to denote the set of non-negative integers, $\mathbb{N}_{\geq 1} = \{1, 2, 3, \dots\}$ to denote the positive integers, and $\mathbb{Z}$ to denote the set of all integers. We write $\mathbb{Q}$ to denote the set of rational numbers and $\mathbb{R}$ to denote the set of all real numbers.

Definition 2.1.1 (Binary Representation). *For a positive integer $n \in \mathbb{N}$ and a position $i \geq 1$, we define $\langle n \rangle[i] \in \{0, 1\}$ to denote the i th bit in the binary representation of n , that is, $n = \sum_{i \geq 1} \langle n \rangle[i] \cdot 2^{i-1}$.*

For two integers a and b and a positive integer m , we write $a \equiv_m b$ as shorthand for $a \equiv b \pmod{m}$.

For integers i, j , we write $[i..j]$ for the set $\{k \in \mathbb{N} \mid i \leq k \leq j\}$, and we write $[i..j)$ for the set $[i..j-1]$.

Properties of Finite and Cofinite Sets. A set $\tau \subseteq \mathbb{N}$ is *cofinite* if $\mathbb{N} \setminus \tau$ is finite. Also, we say that τ is *simple cofinite* if $\mathbb{N} \setminus \tau = [0..n-1]$ for some $n \in \mathbb{N}$, that is, if $\tau = \{n, n+1, n+2, \dots\}$.

Definition 2.1.2. *Denote by $\tau \subseteq \mathbb{N}$ a finite or cofinite set.*

We define $\text{max-gap } \tau$ as the largest non-negative integer d such that there is an $a \in \tau$ with $[a..a+d+1] \cap \tau = \{a, a+d+1\}$.

For arbitrary sets τ , for example, when $\tau = \{2^i \mid i \in \mathbb{N}\}$, the value $\text{max-gap } \tau$ might be undefined.

The following property describes the largest reasonable element of a set.

Definition 2.1.3 (The Top Element). *Consider a set $\tau \subseteq \mathbb{N}$ that is either finite or cofinite. Based on the set τ , we define the number $\text{top } \tau \in \tau$ as*

$$\text{top } \tau := \begin{cases} \max \tau & \text{if } \tau \text{ is finite,} \\ 1 + \max(\mathbb{Z} \setminus \tau) & \text{if } \tau \text{ is cofinite.} \end{cases}$$

Vectors. For some (finite) alphabet Σ , we write Σ^n for the set, or *language*, of all vectors of length n over alphabet Σ . We use ε to denote the empty vector, that is, the vector of length 0. For a finite set X (for instance, a set of vertices of a graph), we write $\Sigma^X := \Sigma^{|X|}$ to emphasize that we index the vectors from Σ^X by elements from X ; formally, assume we order the elements of X as $X = \{x_1, \dots, x_{|X|}\}$, then, for all $x_i \in X$, we write $v[x_i] := v[i]$.

Sometimes we also interpret a vector $v \in \Sigma^X$, as a function $f: X \rightarrow \Sigma$ where $f(x) := v[x]$, and vice versa.

For a vector v of length n and a set of positions $P \subseteq [1..n]$, we write $v[P] := \bigcirc_{a \in P} x[a]$ for the $|P|$ -dimensional vector that contains precisely the entries of v whose indices are contained in the set P . To simplify notation, for two positions $i \leq j \in [1..n]$, we write $s[i..j] := s[i] \cdots s[j]$ for the restriction of v to the positions from $[i..j]$.

Definition 2.1.4 (Combining Vectors with $\sqcup$). *Consider an alphabet Σ and a set X together with a partition $X = X_1 \dot{\cup} X_2$.*

For two vectors $v_1 \in \Sigma^{X_1}$ and $v_2 \in \Sigma^{X_2}$, we say that v_1 extended by v_2 , written as $v_1 \sqcup v_2$, is the unique vector $v \in \Sigma^X$ that satisfies $v[X_1] = v_1$ and $v[X_2] = v_2$.

We extend this definition to the case when the set X_1 and X_2 are not disjoint such that the right vector v_2 takes precedence. Formally, we write $v_1 \sqcup v_2$ for the vector $v \in \Sigma^X$ where $v[X_1 \setminus X_2] := v_1[X_1 \setminus X_2]$ and $v[X_2] := v_2$.

We observe that the operator $\sqcup$ is *not* commutative.

Sometimes we are interested in the number of occurrences of an element $s \in \Sigma$ in a vector v . We use the notation $\#_s(v)$ to denote the number of occurrences of s in vector v , that is, $\#_s(v) := |\{i \in |v| \mid v[i] = s\}|$. Also, for $S \subseteq \Sigma$, we denote by $\#_S(v)$ the number of occurrences of elements from S in v , that is, we define $\#_S(v) := \sum_{s \in S} \#_s(v)$.

To improve readability, we sometimes use $\star$ as a placeholder to indicate that we consider all possible values at the position of $\star$. For example, if we wish to define a function $f: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$, then we write $f(\star, 4) = 5$ to specify that $f(i, 4) = 5$ for all $i \in \mathbb{N}$.

Graphs. We use standard notation for graphs. Formally, a *graph* G consists of a finite vertex set $V(G)$ and finite edge set $E(G) \subseteq V(G) \times V(G)$, that is, we write $G = (V(G), E(G))$. We denote an edge $\{u, v\} \in E(G)$, also by uv .

Unless explicitly stated, the graphs in this thesis are simple and undirected, that is, there are no parallel edges or loops.

For a vertex $v \in V(G)$, we denote by $N_G(v)$ the (*open*) *neighborhood* of vertex v , and by $N_G[v]$ the *closed neighborhood* of vertex v . Formally, we write $N_G(v) := \{w \in V(G) \mid vw \in E(G)\}$ and $N_G[v] := N_G(v) \cup \{v\}$. We define the *degree* of v as the size of the open neighborhood of v , that is, we define $\deg_G(v) := |N_G(v)|$.

For a vertex set $U \subseteq V(G)$, we define the closed neighborhood of U in G as $N_G[U] := \bigcup_{v \in U} N_G[v]$ and the open neighborhood as $N_G(U) := N_G[U] \setminus U$. We usually omit the index G if it is clear from the context.

For a vertex set $U \subseteq V(G)$, we denote by $G[U]$ the *subgraph induced* by the vertices from U , and we denote by $G - X := G[V(G) \setminus X]$ the subgraph induced by the complement of U .

In this paper we regularly insert graphs into other graphs. To make this operation formal, we make use of *dangling edges*: these are edges that have only one endpoint. We denote a dangling edge with endpoint v by $(?, v)$. For the sake of completeness we now formally define this procedure of replacing the relations, i.e. the insertion of a graph into another graph.

Definition 2.1.5 (Insertion). *Let $G = (V, E)$ be a graph and $v \in V$ be of degree k , with incident edges $e_1 = (v_1, v), \dots, e_k = (v_k, v)$ that are ordered in some fixed way. Let $H = (W, F)$ be a*

graph with dangling edges $d_1 = (?, u_1), \dots, d_k = (?, u_k)$ where the u_i are not necessarily pairwise distinct. Inserting H in G at v gives us a new graph $G' = (V', E')$ where:

$$V' = (V \cup W) \setminus \{v\} \quad \text{and} \quad E' = (E \cup F) \setminus \{e_1, \dots, e_k, d_1, \dots, d_k\} \cup \{(v_1, u_1), \dots, (v_k, u_k)\}$$

Relations. In this thesis, we make extensive use of relations. In the following we introduce different notions and explain how they are related. Depending on the context, we use the definition of relation that fits best in the context.

Denote by $\mathbb{D}$ some domain of the relation. Usually we use either (a subset of) the natural number or the values 0 and 1 as domain. We say that a relation is *Boolean* if $\mathbb{D} = \{0, 1\}$.

1. A relation is a function $R_1: \mathbb{D}^k \rightarrow \{0, 1\}$ where a vector $d \in \mathbb{D}^k$ is accepted if $R_1(d) = 1$ and rejected otherwise.
2. A relation is a subset $R_2 \subseteq \mathbb{D}^k$ where a vector $d \in \mathbb{D}^k$ is accepted if $d \in R_2$ and rejected otherwise.
3. A Boolean relation is a subset $R_3 \subseteq 2^{[1..k]}$ and a set $d \subseteq [1..k]$ is accepted if $d \in R_3$ and rejected otherwise.

The transition from the first to the second definition is immediate by just using the subset of $\mathbb{D}^k$ for which R_1 evaluates to 1, that is, we set $R_2 := \{d \in \mathbb{D}^k \mid R_1(d) = 1\}$. Similarly, we can go from the second definition to the first definition by defining a new function that is 1 precisely on R_2 and 0 otherwise, that is, we set $R_1: \mathbb{D}^k \rightarrow \{0, 1\}$ with $R_1(d) = 1$ if and only if $d \in R_2$.

Now we show how to relate the second and third definition. We relate the boolean relations such that the elements of R_3 contain the positions of the 1s in the vectors of R_2 . That is, we have $\{i \in [1..k] \mid d[i] = 1\} \in R_3$ if and only if $d \in R_2$.

We say that a Boolean relation $R: \{0, 1\}^k \rightarrow \{0, 1\}$ is *symmetric* if its output depends on the Hamming weight, that is, the *number* of 1s, of its input only.

To simplify notation, we introduce the following generic classes of symmetric relations.

Definition 2.1.6. For a k -dimensional vector $x \in \{0, 1\}^k$, we define $\text{hw}(x)$ as the number of 1s in x , that is, we define $\text{hw}(x)$ as the Hamming weight of x with $\text{hw}(x) := \sum_{i=1}^k x[i]$.

For two positive integers $i, k \in \mathbb{N}$ and a set $S \subseteq \mathbb{N}$, we define the relations

$$\text{HW}_{\in S}^{(k)} := \{x \in \{0, 1\}^k \mid \text{hw}(x) \in S\}, \quad \text{HW}_{=i}^{(k)} := \text{HW}_{\in \{i\}}^{(k)} \quad \text{and} \quad \text{EQ}_k := \text{HW}_{\in \{0, k\}}^{(k)}.$$

We say that EQ_k is the equality relation on k inputs.

We use $\text{HW}_{\in S}$ to denote the relation $\text{HW}_{\in S}^{(k)}$ when the arity k of the relation is clear from the context. Additionally, we also write $\text{HW}_{\in S}$ to denote the set of the relations $\text{HW}_{\in S}^{(k)}$ for all $k \in \mathbb{N}$.

We transfer this abuse of notation to $\text{HW}_{=i}$ and EQ .

To reduce the number of case distinctions, we use Iverson bracket notation (see, for example, [87]), that is, for a predicate P , we write

$$\llbracket P \rrbracket := \begin{cases} 1 & \text{if } P \text{ is true,} \\ 0 & \text{otherwise.} \end{cases}$$

For example, $\llbracket x < 3 \rrbracket$ evaluates to 1 whenever x is smaller than 3 and 0 otherwise.

Interpolation. In our proofs and construction, we frequently use polynomial interpolation. For a univariate polynomial of degree at most d we can easily recover the coefficients by evaluating the polynomial for $d + 1$ different values.

For multivariate polynomials, we frequently use the following known result.

Fact 2.1.7 ([33, Lemma 2.5]). *Let $P \in \mathbb{Q}[x_1, \dots, x_q]$ be a multivariate polynomial such that, for all $i \in [1..q]$, the degree of x_i in P is bounded by $d_i \in \mathbb{N}$. Furthermore, assume we are given sets $\Xi_i \subseteq \mathbb{Q}$ for $i \in [1..q]$ such that $|\Xi_i| = d_i + 1$ for all $i \in [1..q]$. Consider the Cartesian product of these sets, that is,*

$$\Xi := \Xi_1 \times \dots \times \Xi_q.$$

Then, we can compute the coefficients of P with $\mathcal{O}(|\Xi|^3)$ arithmetic operations when given as input the set $\{(\xi, P(\xi)) \mid \xi \in \Xi\}$.

2.2 Treewidth and Pathwidth

We use the standard definition of tree decompositions and treewidth as for example given in Chapter 7 of [36]. We repeat these definitions here and emphasize certain known but partially non-standard extensions that we use in this work.

A tree decomposition of an undirected graph G is a pair $\mathcal{T} = (T, \{\beta(t)\}_{t \in V(T)})$, where T is a tree, and every node t of T is assigned a *bag* $\beta(t) \subseteq V(G)$, such that the following holds.

- Every vertex of G is contained in at least one bag, that is, $\bigcup_{t \in V(T)} \beta(t) = V(G)$.
- For every edge $uv \in E(G)$, there exists a node t of T such that $\beta(t)$ contains both vertices u and v .
- For every vertex $v \in V(G)$, the nodes whose corresponding bags contain v , that is, the node set $T_u = \{t \in V(T) \mid v \in \beta(t)\}$, induces a connected subtree of T .

The *width* of the tree decomposition is defined as $\max_{t \in V(T)} |\beta(t)| - 1$, that is, as the maximum size of the bags minus 1. The minimum possible width of a tree decomposition of a graph G defines the *treewidth* $\text{tw}(G)$ of G .

The *path decomposition* leading to the pathwidth $\text{pw}(G)$ uses the same definition, except that we require that T is a path.

Nice Decompositions. Mainly for the algorithmic results, we use nice tree decompositions as they help in simplifying the algorithms. More precisely, we usually use a variant of nice decompositions with *introduce edge nodes*. We say that a rooted tree decomposition $\mathcal{T} = (T, \{\beta(t)\}_{t \in V(T)})$ with root node r is *nice* if the bag of the root is empty, that is, if $\beta(r) = \emptyset$, and all nodes t of T have one of the following types:

Leaf Node: The node t has no children and the bag is empty, that is, $\beta(t) = \emptyset$.

Introduce Vertex Node: The node t has a unique child t' such that $\beta(t) = \beta(t') \cup \{v\}$ for some vertex $v \notin \beta(t')$. We say that vertex v is introduced at node t .

We refer to the vertices of T as *nodes*, to distinguish them from the vertices of G .

Introduce Edge Node: The node t has a unique child t' such that $\beta(t) = \beta(t')$. Further, the node t is labeled with an edge $uv \in E(G)$ where $u, v \in \beta(t)$. We say that edge uv is introduced at node t .

Forget Node: The node t has a unique child t' such that $\beta(t) = \beta(t') \setminus \{v\}$ for some vertex $v \in \beta(t')$. We say that vertex v is forgotten at node t .

Join Node: The node t has exactly two children t_1 and t_2 such that $\beta(t) = \beta(t_1) = \beta(t_2)$.

Compared to the classical definition of nice tree decompositions, we use the *Introduce Edge Nodes* to have one explicit node for each edge of the graph. Moreover, we assume that each edge is introduced exactly once.

Given a tree decomposition, we can efficiently transform this decomposition into a nice tree decomposition by increasing its size at most polynomially [86]. Hence, we always assume without loss of generality that a given tree decomposition is nice.

Fact 2.2.1. *Given a tree decomposition $\mathcal{T} = (T, \{\beta(t)\}_{t \in V(T)})$ of a graph G of width at most k , one can in time $k^{O(1)} \cdot \max\{|V(T)|, |V(G)|\}$ compute a nice tree decomposition of G (that introduces each edge exactly once) of width at most k that has at most $O(k \cdot |V(G)|)$ nodes.*

As this is a classical and well-known result of parameterized complexity, we omit the proof and instead, refer the reader to standard textbooks on this topic, for example, to Chapter 7 in [36].

To simplify notation, for a given node t of a tree decomposition, we denote by V_t the vertices introduced at the subtree rooted at t , and (in the context of introduce edge nodes) by E_t the edges introduced in the subtree rooted at t . Moreover, when we consider introduce edge nodes in the algorithmic results, we write G_t for the graph represented by the edges E_t . Otherwise, we write G_t for the graph induced by the vertices in V_t .

Dummy Nodes. Sometimes we allow that a nice tree decomposition or a nice path decomposition (in this case there is no join node) has an additional node type where no change happens at all, that is, no vertex is introduced, no edge is introduced, no vertex is forgotten but there is also only a single child (thus, no join node). We can clearly create such nodes by duplicating a bag.

Pathwidth-Preserving and Arity-Preserving Reductions

For our lower bound results, we consider computational problems for which treewidth (and thus, pathwidth) is defined. While these concepts are generally only defined for graphs, we extend the definitions also to specific variations of graphs.

For these problems we provide reduction chains that do not significantly change the treewidth of the underlying instances, that is, treewidth changes only by an additive constant. To provide a succinct and formal notation for such reductions, we formally introduce *pathwidth-preserving reductions*. By definition of pathwidth and treewidth, most *natural* pathwidth-preserving reductions, that is, reductions that perform “local” modifications of the graph, are treewidth-preserving as well.

Definition 2.2.2 (Pathwidth-Preserving Reduction). *Consider two computational problems Π_1 and Π_2 for which (some notion of) pathwidth is defined.*

We say that there is a pathwidth-preserving reduction from Π_1 to Π_2 , denoted by $\Pi_1 \leq_{\text{pw}} \Pi_2$, if there is a polynomial-time Turing-reduction from Π_1 to Π_2 that, if applied to an instance I of Π_1 that is given with a path decomposition of width at most pw_I , makes oracle calls to Π_2 only on instances that have size polynomial in $|I|$ and that are given with a path decomposition of width at most $\text{pw}_I + \mathcal{O}(1)$.

Similarly to pathwidth- and treewidth-preserving reductions we define an arity-preserving reduction when the concept of *arity* is defined.

Definition 2.2.3 (Arity-Preserving Reduction). *Consider two computational problems Π_1 and Π_2 for which (some notion of) arity is defined.*

We say that there is an arity-preserving reduction from Π_1 to Π_2 , denoted by $\Pi_1 \leq^{\text{ar}} \Pi_2$, if there is a polynomial-time Turing-reduction from Π_1 to Π_2 that, if applied to an instance I of Π_1 with arity at most $\text{ar}(I)$, makes oracle calls to Π_2 only on instances that have size polynomial in $|I|$ and that have arity at most $\text{ar}(I) + \mathcal{O}(1)$.

We say that Π_1 is the source and Π_2 is the target of the reduction.

We combine a pathwidth-preserving and an arity-preserving reduction into a pa-reduction.

Definition 2.2.4 (pa-Reduction). *Consider two computational problems Π_1 and Π_2 for which (some notion of) pathwidth and (some notion of) arity is defined.*

We say that there is a pa-reduction from Π_1 to Π_2 , denoted by $\Pi_1 \leq_{\text{pw}}^{\text{ar}} \Pi_2$, if there is a reduction from Π_1 to Π_2 that is pathwidth-preserving and arity-preserving.

When we combine reductions, we frequently use that pathwidth-preserving and arity-preserving reductions are transitive as the additive increase of the respective measure is bounded by a constant.

Observation 2.2.5 (Transitivity). *A sequence of pathwidth-preserving reductions $\Pi_1 \leq_{\text{pw}} \Pi_2 \leq_{\text{pw}} \dots \leq_{\text{pw}} \Pi_k$ gives a pathwidth-preserving reduction from Π_1 to Π_k if $k \in \mathcal{O}(1)$ (with respect to the input size of Π_1).*

Similarly, a sequence of arity-preserving reductions $\Pi_1 \leq^{\text{ar}} \Pi_2 \leq^{\text{ar}} \dots \leq^{\text{ar}} \Pi_k$ gives an arity-preserving reduction from Π_1 to Π_k if $k \in \mathcal{O}(1)$.

Hence, the corresponding result holds for pa-reductions.

A Toolkit for Improved Algorithms

3

In the following we present two different tools that we later use to obtain faster algorithms. First, in Section 3.1 we provide faster convolution algorithms, that is, we prove Main Theorem I.1. Second, in Section 3.2 we consider representative sets and provide efficient algorithm to compute them, that is, we prove Main Theorem I.2.

3.1 Convolution Algorithms

When designing fast algorithm parameterized by treewidth, for each type of node there must be a specific procedure which describes how to modify the states of the vertices in the current bag. Such a procedure is usually comparably easy to find for introduce (vertex or edge) nodes and forget nodes. Moreover, the procedures for these nodes usually do not introduce a blow-up of the running time. For the join nodes the situation is different. While a naive computation is frequently straightforward to find, such an algorithm often increases the running time too much. Thus, there is a need for optimized convolution algorithms which compute the results *without* slowing down the final algorithm.

Although the main idea of these convolution algorithms is similar, most of these techniques are specifically tailored to the problem at hand in order to obtain faster algorithms. Thus, they do not allow an application to other problems without repeating the entire construction. See [130, 132] for examples of optimal convolution algorithms.

Our goal of this section is to state a general convolution algorithm that is applicable in multiple settings without further modifications. Our algorithm allows four different types of domains over which the convolution can be computed.

- The cyclic domain denoted by $\mathbb{S}$ agrees with the normal addition modulo some predefined number that might be different for each dimension.
- The addition domain $\mathbb{A}$ with many dimension and small coefficients allows standard addition where the maximum values for each addition are assumed to be small. Whenever the result of the addition would surpass the bound, the sum is undefined.

We remark that there is no formal restriction to small integers but for large integers there is a separate domain which allows for a better performance instead.

- The addition domain $\mathbb{L}$ with few dimensions but large coefficients can be seen as the counterpart of the previous domain.
- The truncated addition domain $\mathbb{T}$. Recall that for the normal addition the sum is undefined if the sum extends some predefined value. For the truncated addition we remove

this requirement and introduce some distinguished element that represents a “large” sum.

Before we state our main result, we formally define the truncated addition.

Definition 3.1.1. For some non-negative integer $t \in \mathbb{N}$, consider the set $T = [0 .. t) \cup \{\top\}$. For all $u, v \in U$, the truncated addition of u and v with respect to t , denoted by $u \oplus_t v$ is defined as

$$u \oplus_t v := \begin{cases} \top & \text{if } \top \in \{u, v\} \text{ or } u + v > t, \\ u + v & \text{otherwise.} \end{cases}$$

If the value of t is clear from the context, we omit t from the notation and just write $u \oplus v$.

With this definition, we can now formally state Main Theorem I.1, which is the main result of this section.

Main Theorem I.1 (General Convolution). Consider a domain $\mathbb{D}$ composed of the four subdomains $\mathbb{S}$, $\mathbb{A}$, $\mathbb{L}$, and $\mathbb{T}$ such that $\mathbb{D} := \mathbb{S} \times \mathbb{A} \times \mathbb{L} \times \mathbb{T}$

- with $\mathbb{S} = \mathbb{Z}_{z_1} \times \cdots \times \mathbb{Z}_{z_\sigma}$ for the first domain,
- with $\mathbb{A} = [0 .. q_1) \times \cdots \times [0 .. q_\alpha)$ for the second domain and define $A_+ := \sum_{i=1}^\alpha q_i$,
- with $\mathbb{L} = [0 .. x_1) \times \cdots \times [0 .. x_\lambda)$ for the third domain, and
- with $\mathbb{T} = [0 .. y_1) \cup \{\top\} \times \cdots \times [0 .. y_\tau) \cup \{\top\}$ for the fourth domain and define $T_+ := \sum_{i=1}^\tau y_i$.

Denote by $f_1, f_2: \mathbb{D} \rightarrow \mathbb{N}$ two functions that are given as table representations.

We denote by $f: \mathbb{D} \rightarrow \mathbb{N}$ the convolution of f_1 and f_2 that satisfies, for all $(s, a, \ell, t) \in \mathbb{S} \times \mathbb{A} \times \mathbb{L} \times \mathbb{T} = \mathbb{D}$,

$$f(s, a, \ell, t) = \sum_{s_1+s_2=s} \sum_{a_1+a_2=a} \sum_{\ell_1+\ell_2=\ell} \sum_{t_1 \oplus t_2=t} f_1(s_1, a_1, \ell_1, t_1) \cdot f_2(s_2, a_2, \ell_2, t_2).$$

If U denotes a non-negative integer such that all images of f_1 , f_2 and f are contained in $[0 .. U)$, then the function f can be computed in time

$$X \cdot \log X \cdot (\log U + \overline{M})^{\mathcal{O}(1)} + 2^\tau \cdot (\log U + \exp(\overline{M}))^{\mathcal{O}(1)}$$

with $X = |\mathbb{D}| \cdot |\mathbb{L}|^{\log(3/2)} \cdot (A_+ + T_+)^{\log(3)} \cdot 3^\lambda$ and $\overline{M} = \prod_{m \in M} m$ where $M = \{z_1, \dots, z_\sigma, q_1, \dots, q_\alpha, 3, y_1, \dots, y_\tau, 3\}$.

We prove this result in four incremental but self-contained steps.

1. We first modify the result by van Rooij [131, Lemma 5] such that it can be used in a self-contained manner. Formally, we prove Lemma 3.1.6, which corresponds to Main Theorem I.1 for the case that $\alpha = \lambda = \tau = 0$, in Section 3.1.1.
2. Second, in Section 3.1.2 we extend the previous result to the case where $\alpha = \tau = 0$ by Lemma 3.1.7.
3. Third, in Section 3.1.3 we lift this result to the case where only $\tau = 0$ by Lemma 3.1.8.
4. Finally, in Section 3.1.4, we extend the previous results and include the truncated addition, that is, we prove Main Theorem I.1 for the case when $\tau > 0$.

3.1.1 Cyclic Addition

To prove faster algorithm for DOMINATING SET and its generalization (σ, ρ) -DOMSET, van Rooij provides a general algorithm to compute a so-called cyclic convolution [130, 131]. Formally, van Rooij proves Fact 3.1.2.

Fact 3.1.2 (Cyclic Convolution [130, Lemma 3] and [131, Lemma 5]). *Let $Z = \mathbb{Z}_{r_1} \times \mathbb{Z}_{r_1} \times \dots \times \mathbb{Z}_{r_k}$, and let $R = \prod_{i=1}^k r_i$.*

Let p denote a prime such that in the field $\mathbb{F}_p$, the r_i -th root of unity exists for each $i \in [1 \dots k]$.

Let $f, g: Z \rightarrow \mathbb{F}_p$. And, let $h: Z \rightarrow \mathbb{F}_p$ be the combined (cyclic) convolution of f and g defined as:

$$h(x) = \sum_{y_1 + y_2 \equiv x} f(y_1) \cdot g(y_2)$$

where the sum $y_1 + y_2 \equiv x$ is evaluated component-wise modulo r_i at coordinate i (sum in Z). Then, the combined convolution h can be computed in $\mathcal{O}(R \log(R))$ arithmetic operations (assuming an r_i -th root of unity ω_i is given for all $i \in [1 \dots k]$).

As this result depends on the existence of a special prime number and of roots of unity, the result cannot be applied in a black-box manner but needs some further preprocessing. Our first intermediate goal is to extend this result such that we do not require the roots of unity to be explicitly given.

To make this possible, we first discuss how to find an appropriate prime p , as well as the roots of unity ω_i . In order to bound the running time of the corresponding algorithm and the computed integers, we use the Prime Number Theorem for Arithmetic Progressions from Fact 3.1.3.

Fact 3.1.3 ([8, Corollary 1.8]). *Let q be an integer with $q \geq 3$. Suppose that $x \geq \exp(8\sqrt{q} \cdot \log^3 q)$. We denote by $\theta: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ the function where, for all $x, q \in \mathbb{N}$ we have*

$$\theta(x, q) = \sum_{\substack{\text{prime } p < x: \\ p \equiv 1 \pmod{q}}} \log p.$$

Then, we have that

$$\theta(x, q) \geq \frac{x}{\phi(q)} - \frac{1}{160} \frac{x}{\log x}$$

where ϕ is Euler's totient function.

Phrased differently, the value $2^{\theta(x, q)}$ denotes the product of those prime numbers p that are smaller than x and satisfy $p \equiv 1 \pmod{q}$.

Now we have everything ready to prove the existence (and computation) of the prime number and the roots of unity needed for the convolution. Instead of focusing on a single prime number, we compute a list of distinct prime numbers and the corresponding roots of unity. We use these numbers to apply the Chinese Remainder Theorem in order to obtain a faster algorithm for the convolution.

We start with the computation of the prime numbers and the roots of unity.

3 A Toolkit for Improved Algorithms

Lemma 3.1.4. Consider a positive integer $U \in \mathbb{N}$ and consider a list of μ pairwise different integers $m_1, \dots, m_\mu \in \mathbb{N} \setminus \{0, 1\}$. We set $M := \prod_{i=1}^\mu m_i$.

We can find a list of $q \leq \log U$ pairwise different primes $p_1, \dots, p_q$ such that $\prod_{i=1}^q p_i > U$ with $p_i \in \mathcal{O}(\log U \cdot \exp(M))$ for all $i \in [1..q]$, and we can find, for all $i \in [1..q]$ and all $j \in [1..\mu]$, an m_j th primitive root of unity $\omega_{i,j}$ in the field $\mathbb{F}_{p_i}$.

All these numbers can be found in time

$$\mathcal{O}(\log^3 U \cdot \exp(2M)).$$

Proof. We first give the algorithm that computes the claimed numbers. Afterward we bound the running time of the procedure and prove the claimed bounds.

1. Iterate over the candidate numbers of the form $1 + M \cdot i$ for $i \geq 1$ and check whether these numbers are prime. If the number $1 + M \cdot i$ is prime, then we add it to the initially empty set P .

As soon as $U \leq \prod_{p \in P} p$, we proceed to the next step and write $P := \{p_1, \dots, p_q\}$ for the $q := |P|$ selected prime numbers where we assume that $p_1 < \dots < p_q$.

2. To find the roots of unity, we repeat the following procedure for all $i \in [1..q]$ and for all $j \in [1..\mu]$.

Iterate over all elements $x \in \mathbb{F}_{p_i} \setminus \{0\}$ and test whether $x^{(p_i-1)/m_j}$ is an m_j th primitive root of unity, that is, we check, for all $\ell \in [0..m_j)$, if

$$\left(x^{(p_i-1)/m_j}\right)^\ell \neq 1.$$

Observe that $x^{p_i-1} \equiv 1 \pmod{p_i}$ and hence, the candidate is always a root of unity that might be non-primitive.

If we conclude that $x^{(p_i-1)/m_j}$ is an m_j th primitive root of unity in $\mathbb{F}_{p_i}$, then we set $\omega_{i,j} := x^{(p_i-1)/m_j}$.

Although the algorithm is comparably simple, bounding the running time is non-trivial. We first state the running time implicitly and afterward prove the explicit bound based on the Prime Number Theorem from Fact 3.1.3.

To simplify notation, we set $\hat{m} = \max_{j \in [1..\mu]} m_j$ in the following.

Claim 3.1.4.1. The algorithm operates in time $q \cdot \mu \cdot p_q \cdot \hat{m} \cdot \log^{\mathcal{O}(1)} p_q$.

Proof of Claim. Since we assume that p_q is the largest prime number, we know that the algorithm tests $(p_q - 1)/M$ candidates to find the prime numbers. As we can test in polynomial-time in the size of the encoding whether a number is prime [2], the check for each candidate takes time at most $\log^{\mathcal{O}(1)} p_q$. We conclude that the first step of the algorithm takes time $p_q/M \cdot \log^{\mathcal{O}(1)} p_q$.

For the second step, we fix some indices $i \in [1..q]$ and $j \in [1..\mu]$. For each such choice, we need to consider p_i elements from $\mathbb{F}_{p_i}$ and compute $m_j \leq \hat{m}$ numbers. As all computations take time polynomial in the size of the encoding, the entire second step takes time

$$q \cdot \mu \cdot p_q \cdot \hat{m} \cdot \log^{O(1)} p_q$$

by using a naive bound.

This concludes the proof as the running time of the second step dominates the running time from the first step. $\blacktriangleleft$

As the running time stated in Claim 3.1.4.1 depends on q and p_q , our next goal is to make these bound explicit such that they only depend on the numbers given as input.

For this we first observe that, by the design of the algorithm, we have $U > \prod_{i=1}^{q-1} p_i \geq 2^{q-1}$. Hence, we immediately get that

$$q < \log U + 1. \quad (3.1.1)$$

As a next step, we consider the bound on p_q . We define

$$B = \max\{\exp(8 \cdot \sqrt{M} \cdot \log^3 M), \exp(M), 2M \log U\} \quad (3.1.2)$$

and prove that the computed prime numbers are bounded by B . For this we show the following auxiliary result.

Claim 3.1.4.2. *It holds that $\theta(B, M) \geq \log U$ with θ as in Fact 3.1.3.*

Proof of Claim. By our choice of B , we get $B \geq \exp(8\sqrt{M} \cdot \log^3 M)$ and can use Fact 3.1.3 to obtain

$$\begin{aligned} \theta(B, M) &\geq \frac{B}{\phi(M)} - \frac{1}{160} \frac{B}{\log B} \\ &= B \cdot \left(\frac{1}{\phi(M)} - \frac{1}{160} \frac{1}{\log B} \right). \end{aligned}$$

By the definition of the totient function ϕ , we have $\phi(M) \leq M$ and, by our choice of B , we know that $B \geq 2d \log U$ and $\log B \geq M$, which yields

$$\begin{aligned} \theta(B, M) &\geq 2M \log U \cdot \left(\frac{1}{M} - \frac{1}{160M} \right) \\ &\geq 2M \cdot \log U \cdot \frac{159}{160M} \\ &\geq \log U \end{aligned}$$

and thus, concludes the proof. $\blacktriangleleft$

It remains to analyze the implications of Claim 3.1.4.2. By the design of the algorithm, we know that $p_1, \dots, p_q$ are the q smallest prime numbers with $p_i \equiv 1 \pmod{M}$ such that $\prod_{i=1}^q p_i \geq U$. We especially get $\sum_{i=1}^q \log p_i \geq \log U$.

By the definition of $\theta(x, q)$, we know that

$$\theta(B, M) = \sum_{\substack{\text{prime } p \leq B: \\ p \equiv 1 \pmod{M}}} \log p.$$

Hence, the sum from $\theta(B, M)$ ranges over all prime numbers p_i . Therefore, we conclude that $p_i \leq B$ for all $i \in [1 .. q]$. Moreover, we observe that the bound on B can be simplified such that $B \leq 2 \log U \cdot \exp(M)$. Hence, we have that

$$p_i \leq 2 \log U \cdot \exp(M) \quad \text{for all } i \in [1 .. q]. \quad (3.1.3)$$

Combining Claim 3.1.4.1 with Equations (3.1.1) and (3.1.3) we conclude that we can compute the prime numbers in time

$$q \cdot \mu \cdot p_q \cdot \hat{m} \cdot \log^{\mathcal{O}(1)} p_q \leq \log U \cdot \mu \cdot (2 \log U \cdot \exp(M)) \cdot \hat{m} \cdot \log^{\mathcal{O}(1)} (2 \log U \cdot \exp(M)).$$

As the final step, we simplify this bound and obtain

$$\begin{aligned} q \cdot \mu \cdot p_q \cdot \hat{m} \cdot \log^{\mathcal{O}(1)} p_q &\leq \log^2 U \cdot \mu \cdot \exp(M) \cdot \hat{m} \cdot (\log^{\mathcal{O}(1)} \log U + M^{\mathcal{O}(1)}) \\ &\leq \log^3 U \cdot \mu \cdot \exp(M) \cdot \hat{m} \cdot M^{\mathcal{O}(1)}. \end{aligned}$$

When we use that M is the product of the μ numbers $2 \leq m_j \leq \hat{m}$, we get

$$\begin{aligned} q \cdot \mu \cdot p_q \cdot \hat{m} \cdot \log^{\mathcal{O}(1)} p_q &\leq \log^3 U \cdot M \cdot \exp(M) \cdot M \cdot M^{\mathcal{O}(1)} \\ &\leq \log^3 U \cdot \exp(M) \cdot M^{\mathcal{O}(1)} \\ &\leq \log^3 U \cdot \exp(2M). \end{aligned}$$

This concludes the proof. $\square$

In the computation of the convolution we make use of the Chinese Remainder Theorem.

Fact 3.1.5 (Chinese Remainder Theorem [133, Theorem 5.8]). *Let $m_1, \dots, m_\ell$ denote a sequence of integers that are pairwise coprime, and define $M := \prod_{i \in [1 .. \ell]} m_i$. Also, let $0 \leq a_i < m_i$ for all $i \in [1 .. \ell]$. Then, there is a unique number $0 \leq s < M$ such that*

$$s \equiv a_i \pmod{m_i}$$

for all $i \in [1 .. \ell]$. Moreover, there is an algorithm that, given $m_1, \dots, m_\ell$ and $a_1, \dots, a_\ell$, computes the number s in time $\mathcal{O}(\log^2 M)$.

Now the idea is as follows. We use Lemma 3.1.4 to compute multiple primes and roots of unity. Then, we use the convolution algorithm from Fact 3.1.2 to compute the convolution modulo each prime number. Finally, we use the Chinese Remainder Theorem to recover the original convolution function. Formally, we prove Lemma 3.1.6 which is the special case of the final convolution from Main Theorem 1.1 for $\alpha = \lambda = \tau = 0$.

Lemma 3.1.6. Consider a list of n integers $d_1, \dots, d_n \geq 2$, with $d'_1, \dots, d'_\ell$ as unique elements of the list. We define $\mathbb{S} := \mathbb{Z}_{d_1} \times \dots \times \mathbb{Z}_{d_n}$ as a domain and let $D' := \prod_{i=1}^\ell d'_i$.

Denote by $f_1, f_2: \mathbb{S} \rightarrow \mathbb{N}$ two functions that are given as table representations. We denote by $f: \mathbb{S} \rightarrow \mathbb{N}$ the convolution of f_1 and f_2 that satisfies, for all $s \in \mathbb{S}$,

$$f(s) = \sum_{s_1+s_2 \equiv s} f_1(s_1) \cdot f_2(s_2)$$

where the addition is computed modulo d_i at position i .

If U denotes a non-negative integer such that all images of f_1 , f_2 and f are contained in $[0 .. U)$, then the function f can be computed in time

$$|\mathbb{S}| \cdot \log|\mathbb{S}| \cdot (\log U + D')^{\mathcal{O}(1)} + (\log U + \exp(D'))^{\mathcal{O}(1)}.$$

Proof. The algorithm works as follows.

1. Denote by $d'_1, \dots, d'_\ell$ the list of integers obtained from $d_1, \dots, d_n$ by removing duplicates. We set $D' := \prod_{i=1}^\ell d'_i$.
2. We use Lemma 3.1.4 to compute $q \leq \log U$ prime numbers $p_1, \dots, p_q$ with $p_i \leq \mathcal{O}(\log U \cdot \exp(D'))$ and roots of unity $\omega_{i,j}$ for all $i \in [1 .. q]$ and $j \in [1 .. \ell]$.
3. For every $i \in [1 .. q]$, we compute the two functions $f_1^{(i)}, f_2^{(i)}: \mathbb{S} \rightarrow \mathbb{F}_{p_i}$ where, for all $a \in \mathbb{S}$, we set

$$f_1^{(i)}(a) := f_1(a) \bmod p_i \quad \text{and} \quad f_2^{(i)}(a) := f_2(a) \bmod p_i.$$

4. For every $i \in [1 .. q]$, we use Fact 3.1.2 to compute the function $f^{(i)}: \mathbb{S} \rightarrow \mathbb{F}_{p_i}$ as the convolution of the two functions $f_1^{(i)}$ and $f_2^{(i)}$.
5. We compute the function $f: \mathbb{S} \rightarrow \mathbb{N}$ as follows. For all $a \in \mathbb{S}$, we use the Chinese Remainder Theorem from Fact 3.1.5 to recover the value $f(a) \in [0 .. U)$ from the values $f^{(1)}(a), \dots, f^{(q)}(a)$.

This concludes the description of the algorithm.

The correctness follows immediately from the correctness of the convolution in Fact 3.1.2, the choice of the prime numbers in Lemma 3.1.4, and the Chinese Remainder Theorem. Indeed, as $\prod_{i=1}^q p_i \geq U$ and the images of the functions f_1 , f_2 , and f are smaller than U by assumption, Fact 3.1.5 correctly recovers the function f as the convolution of the two functions f_1 and f_2 .

It remains to bound the running time of the algorithm.

Claim 3.1.6.1. The algorithm operates in time

$$|\mathbb{S}| \cdot \log|\mathbb{S}| \cdot (\log U + D')^{\mathcal{O}(1)} + (\log U + \exp(D'))^{\mathcal{O}(1)}.$$

Proof of Claim. We first analyze each step individually.

1. The first step operates in time $\mathcal{O}(n^2)$ by a naive comparison of the numbers to identify duplicates.

2. According to Lemma 3.1.4, the second step takes time $\mathcal{O}(\log^3 U \cdot \exp(2D'))$ as we only need to consider the distinct numbers d'_i from the first step.
3. The third step computes the functions $f_1^{(i)}$ and $f_2^{(i)}$ for each $i \in [1..q]$. As reading and writing a single number (or element in $\mathbb{S}$) takes time $\log|\mathbb{S}| \cdot \log U$, this takes time $\mathcal{O}(q \cdot |\mathbb{S}| \cdot \log|\mathbb{S}| \cdot \log U)$ in total, which we can bound by $\mathcal{O}(|\mathbb{S}| \cdot \log|\mathbb{S}| \cdot \log^2 U)$.
4. The fourth step computes the convolution of the functions by Fact 3.1.2. As each arithmetic operation takes time $\log^{\mathcal{O}(1)} p_q$, the computation takes time at most

$$\begin{aligned} q \cdot \mathcal{O}(|\mathbb{S}| \cdot \log|\mathbb{S}|) \cdot \log^{\mathcal{O}(1)} p_q &\leq \log U \cdot \mathcal{O}(|\mathbb{S}| \cdot \log|\mathbb{S}|) \cdot \log^{\mathcal{O}(1)} (\log(U) \cdot \exp(D')) \\ &\leq \log^2 U \cdot |\mathbb{S}| \cdot \log|\mathbb{S}| \cdot (D')^{\mathcal{O}(1)} \end{aligned}$$

5. For the fifth step, we first bound $P = \prod_{i=1}^q p_i$ from above. As we know that $\prod_{i=1}^{q-1} p_i < U$ and $p_q \in \mathcal{O}(\log U \cdot \exp(D'))$, we get that $P \leq \mathcal{O}(U \cdot \log U \cdot \exp(D'))$. Moreover, we have that $\log P \leq \log U + \log \log U + (D')^{\mathcal{O}(1)}$.

Then, the last step of the algorithm takes time

$$\begin{aligned} |\mathbb{S}| \cdot \log|\mathbb{S}| \cdot \log^2 P &\leq |\mathbb{S}| \cdot \log|\mathbb{S}| \cdot (\log U + \log \log U + (D')^{\mathcal{O}(1)})^2 \\ &\leq |\mathbb{S}| \cdot \log|\mathbb{S}| \cdot (\log U + D')^{\mathcal{O}(1)}. \end{aligned}$$

To bound the final running time of the algorithm, we observe that the second and fifth step determine the running time. Hence, we conclude that the algorithm operates in time

$$|\mathbb{S}| \cdot \log|\mathbb{S}| \cdot (\log U + D')^{\mathcal{O}(1)} + (\log U + \exp(D'))^{\mathcal{O}(1)}$$

which concludes the proof of the running time. $\blacktriangleleft$

From the description of the algorithm and the bound on the running time, we conclude the proof of the algorithm computing the convolution. $\square$

As a next step, we include addition in the convolution.

3.1.2 Standard Addition with Large Numbers

In the following we extend Lemma 3.1.6 from Section 3.1.1 such that we also allow addition, that is, we prove Main Theorem I.1 for $\alpha = \tau = 0$. A similar result was already shown by van Rooij [130, 131]. However, said algorithm requires that the prime numbers and roots of unity are given explicitly.

With our previous result it seems tempting to repeat the earlier result by van Rooij (see the proof of Lemma 3 in [130] for example) based on our previous result as starting point.

Before we analyze this approach we first sketch the idea of the procedure. For this, consider a domain $[0..q)$ where we use standard addition to compute the convolution. Whenever we add two numbers $a, b \in [0..q)$ with standard addition, we can obtain the same result by computing the sum $a + b$ modulo $2q$, that is, by addition in the field $\mathbb{Z}_{2q}$. Hence, the idea

of van Rooij is to consider the domain $[0 \dots 2q)$ instead of the domain $[0 \dots q)$. Therefore, by doubling the domain size, we can easily reduce addition to cyclic addition, but we get an additional factor of 2 in the running time for each such dimension.

However, this modification also affects the procedure to compute the primes and roots of unity presented in Section 3.1.1. Thus, whenever the domain is large, and by large we mean a term at least linear in the dimension of the functions convoluted, the running time would increase by an *exponential* factor—too slow for our later applications.

With our approach we avoid this exponential increase of the running time and exploit that only few dimensions might have large domains. Instead of computing the convolution directly, we encode the numbers in binary. Then, we compute the convolution of this binary encoding and recover the represented numbers in a final step. Nevertheless, we still lack a procedure that provides a convolution for the numbers represented in binary.

As a direct application of the cyclic convolution modulo 2 would result in a loss of information because of the overflows appearing, we compute the convolution modulo 3. In this case, whenever we add two binary numbers, we can recover the original number from the point-wise addition of the two representations.

Before we start with the formal proof, we first focus on the problem of computing the sum of the binary representations modulo 3. More specifically, by this choice, multiple vectors can represent the same number. For example, consider a binary representation with four digits. Then, the vector $(0, 0, 0, 1)$ represents the (canonical) binary representation of the integer $8 = 0 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3$. However, since the coordinates may contain entries from $\{0, 1, 2\}$, an alternative representation is $(0, 0, 2, 0)$ since $8 = 0 \cdot 2^0 + 0 \cdot 2^1 + 2 \cdot 2^2 + 0 \cdot 2^3$.

Hence, in the following proof it does not suffice to consider proper (binary) representations of a number only. Instead, we need to consider all possibilities this number could be represented.

With this observation we proceed to the formal proof.

Lemma 3.1.7. *Consider two domains of vectors $\mathbb{S}$ and $\mathbb{L}$*

- *with $\mathbb{S} = \mathbb{Z}_{z_1} \times \dots \times \mathbb{Z}_{z_\sigma}$ for the first domain, and*
- *with $\mathbb{L} = [0 \dots x_1) \times \dots \times [0 \dots x_\lambda)$ for the second domain.*

Denote by $f_1, f_2: \mathbb{S} \times \mathbb{L} \rightarrow \mathbb{N}$ two functions that are given as table representations.

We denote by $f: \mathbb{S} \times \mathbb{L} \rightarrow \mathbb{N}$ the convolution of f_1 and f_2 that satisfies, for all $s \in \mathbb{S}, \ell \in \mathbb{L}$,

$$f(s, \ell) = \sum_{s_1+s_2=s} \sum_{\ell_1+\ell_2=\ell} f_1(s_1, \ell_1) \cdot f_2(s_2, \ell_2). \quad (3.1.4)$$

If U denotes a non-negative integer such that all images of f_1, f_2 and f are contained in $[0 \dots U)$, then the function f can be computed in time

$$X \cdot \log X \cdot (\log U + \overline{M})^{\mathcal{O}(1)} + (\log U + \exp(\overline{M}))^{\mathcal{O}(1)}$$

with $X = |\mathbb{S}| \cdot |\mathbb{L}|^{\log 3} \cdot 3^\lambda$ and $\overline{M} = \prod_{m \in M} m$ where $M = \{z_1, \dots, z_\sigma, 3\}$.

Proof. As previously mentioned, the idea is to encode the vectors from $\mathbb{L}$ in binary and compute the convolution modulo 3. Then, we compute a cyclic convolution and recover the vectors from $\mathbb{L}$ from the representation.

To formally state the algorithm, we first define the binary encoding of a number. For a number $c \in [0 .. x]$, we define the binary encoding of c as the unique vector $\langle c \rangle \in \{0, 1\}^{\lceil \log x \rceil}$ of length $\lceil \log x \rceil$ such that $c = \sum_{j=0}^{\lceil \log x \rceil} \langle c \rangle[j] \cdot 2^{j-1}$.

Moreover, for a vector $a \in \{0, 1, 2\}^{\lceil \log x \rceil}$, we define $\llbracket a \rrbracket = \sum_{j=1}^{\lceil \log x \rceil} a[j] \cdot 2^{j-1}$ as the reversal of the binary representation. Note that we use the same symbol as for the Iversion bracket notation to highlight the similarity with the *interpretation* of the given vector as number.

We emphasize that this operation is defined for vectors over $\{0, 1, 2\}$ while the binary representation just gives a vector over $\{0, 1\}$. We especially get that $c = \llbracket \langle c \rangle \rrbracket$.

We lift the binary encoding of a single number to the binary encoding of a vector of numbers (all possibly from different ranges), by simply concatenating the binary encodings of the individual numbers. Similarly, we define the reversal of the binary representation for a vector consisting of multiple encoded numbers in the natural way.

Now we have everything ready to give the formal description of the algorithm.

1. For all $i \in [1 .. \lambda]$, we define $b_i := \lceil \log x_i \rceil$ and write $\hat{\mathbb{L}}_i := \times_{j=1}^{b_i} \mathbb{Z}_3$.

We write $\hat{\mathbb{L}} := \hat{\mathbb{L}}_1 \times \dots \times \hat{\mathbb{L}}_\lambda$ as a representation of the domain $\mathbb{L}$.

2. We define two new functions $g_1, g_2: \mathbb{S} \times \hat{\mathbb{L}} \rightarrow \mathbb{N}$ where, for all $s \in \mathbb{S}$ and for all $\ell \in \mathbb{L}$, we set

$$g_1(s, \langle \ell \rangle) := f_1(s, \ell) \quad \text{and} \quad g_2(s, \langle \ell \rangle) := f_2(s, \ell).$$

For all $s \in \mathbb{S}$ and $\hat{\ell} \in \hat{\mathbb{L}} \setminus \{\langle \ell \rangle \mid \ell \in \mathbb{L}\}$, we define $g_1(s, \hat{\ell}) = g_2(s, \hat{\ell}) = 0$ instead.

3. We use Lemma 3.1.6 to compute the function $g: \mathbb{S} \times \hat{\mathbb{L}} \rightarrow \mathbb{N}$ as the convolution of the two functions g_1 and g_2 .
4. Based on the function g , we compute the function $f: \mathbb{S} \times \mathbb{L} \rightarrow \mathbb{N}$ which is the output of the algorithm.

First, we initialize f with 0 for all entries.

Then, we iterate over all $s \in \mathbb{S}$ and all $\hat{\ell} \in \hat{\mathbb{L}}$, and set $f(s, \llbracket \hat{\ell} \rrbracket) := f(s, \llbracket \hat{\ell} \rrbracket) + g(s, \hat{\ell})$, that is, we increase the value of $f(s, \llbracket \hat{\ell} \rrbracket)$ by $g(s, \hat{\ell})$.

We first argue that the algorithm is correct.

Claim 3.1.7.1. *The function f from the last step of the algorithm satisfies Equation (3.1.4).*

Proof of Claim. We first define some further notation about the binary representation and the corresponding reversal. For a vector of numbers $\ell \in \mathbb{L}$, we define $\langle\langle \ell \rangle\rangle := \{\hat{\ell} \in \hat{\mathbb{L}} \mid \ell = \llbracket \hat{\ell} \rrbracket\}$ as the set of all vectors from $\hat{\mathbb{L}}$ that represent the vector ℓ .

From the definition of the function f in the last step of the algorithm and by the correctness of the convolution algorithm from Lemma 3.1.6, we know that

$$\begin{aligned} f(s, \ell) &= \sum_{\hat{\ell} \in \langle\langle \ell \rangle\rangle} g(s, \hat{\ell}) \\ &= \sum_{\hat{\ell} \in \langle\langle \ell \rangle\rangle} \sum_{s_1 + s_2 \equiv s} \sum_{\substack{\ell_1 \in \hat{\mathbb{L}}, \ell_2 \in \hat{\mathbb{L}}: \\ \ell_1 + \ell_2 \equiv \hat{\ell}}} g_1(s_1, \ell_1) \cdot g_2(s_2, \ell_2). \end{aligned}$$

We know that for all $s' \in \mathbb{S}$ and $\ell' \in \widehat{\mathbb{L}} \setminus \langle \mathbb{L} \rangle$, where we write $\langle \mathbb{L} \rangle := \{\langle \ell \rangle \mid \ell \in \mathbb{L}\}$ to simplify notation, we have $g_1(s', \ell') = g_2(s', \ell') = 0$. This yields

$$f(s, \ell) = \sum_{\widehat{\ell} \in \langle \ell \rangle} \sum_{s_1 + s_2 \equiv s} \sum_{\substack{\ell_1 \in \langle \mathbb{L} \rangle, \ell_2 \in \langle \mathbb{L} \rangle: \\ \widehat{\ell}_1 + \widehat{\ell}_2 \equiv \widehat{\ell}}} g_1(s_1, \widehat{\ell}_1) \cdot g_2(s_2, \widehat{\ell}_2).$$

Now observe that for each pair $\widehat{\ell}_1, \widehat{\ell}_2 \in \langle \mathbb{L} \rangle$, there is a unique $\widehat{\ell} \in \langle \ell \rangle$ such that $\widehat{\ell}_1 + \widehat{\ell}_2 \equiv \widehat{\ell}$ because for each dimension the sum is computed modulo 3 and $\widehat{\ell}_1$ and $\widehat{\ell}_2$ are 0-1-vectors by definition. Hence, the equivalence is actually equality.

Now consider two vectors $\widehat{\ell}_1, \widehat{\ell}_2 \in \langle \mathbb{L} \rangle$ that satisfy $\widehat{\ell}_1 + \widehat{\ell}_2 \in \langle \ell \rangle$. By the definition of $\langle \star \rangle$, this is equivalent to $\ell = \llbracket \widehat{\ell}_1 + \widehat{\ell}_2 \rrbracket = \llbracket \widehat{\ell}_1 \rrbracket + \llbracket \widehat{\ell}_2 \rrbracket$. Hence, there are two vectors $\ell_1, \ell_2 \in \mathbb{L}$ such that $\ell = \ell_1 + \ell_2$.

Therefore, we obtain

$$\begin{aligned} f(s, \ell) &= \sum_{s_1 + s_2 \equiv s} \sum_{\substack{\ell_1 \in \mathbb{L}, \ell_2 \in \mathbb{L}: \\ \ell_1 + \ell_2 = \ell}} g_1(s_1, \langle \ell_1 \rangle) \cdot g_2(s_2, \langle \ell_2 \rangle) \\ &= \sum_{s_1 + s_2 \equiv s} \sum_{\substack{\ell_1 \in \mathbb{L}, \ell_2 \in \mathbb{L}: \\ \ell_1 + \ell_2 = \ell}} f_1(s_1, \ell_1) \cdot f_2(s_2, \ell_2) \end{aligned}$$

by the definition of the functions g_1 and g_2 which proves the correctness of the algorithm. $\blacktriangleleft$

It remains to analyze the running time of the algorithm.

Claim 3.1.7.2. *The algorithm operates in time*

$$|\mathbb{S}| \cdot |\mathbb{L}|^{\log 3} \cdot 3^\lambda \cdot (\log |\mathbb{S}| + \log |\mathbb{L}| + \lambda) \cdot (\log U + \overline{M})^{\mathcal{O}(1)} + (\log U + \exp(\overline{M}))^{\mathcal{O}(1)}.$$

Proof of Claim. The computations of the functions g_1 and g_2 in the second step and the function f in the fourth step take time $\mathcal{O}(|\mathbb{S}| \cdot |\widehat{\mathbb{L}}| \cdot \log U \cdot (\alpha + \sum_{i=1}^\lambda b_i))$.

By Lemma 3.1.6, the third step takes time

$$|\mathbb{S}| \cdot |\widehat{\mathbb{L}}| \cdot (\log |\mathbb{S}| + \log |\widehat{\mathbb{L}}|) \cdot (\log U + \overline{M})^{\mathcal{O}(1)} + (\log U + \exp(\overline{M}))^{\mathcal{O}(1)}$$

where $\overline{M} = \prod_{m \in M} m$ and $M = \{z_1, \dots, z_\sigma, 3\}$ as, for each dimension of $\widehat{\mathbb{L}}$, the computation is modulo 3.

Now, from the definition of $\widehat{\mathbb{L}}$, we know that

$$|\widehat{\mathbb{L}}| = \prod_{i=1}^\lambda 3^{\lceil \log x_i \rceil} \leq \prod_{i=1}^\lambda 3^{1 + \log x_i} = 3^\lambda \cdot \prod_{i=1}^\lambda x_i^{\log 3} = 3^\lambda \cdot |\mathbb{L}|^{\log 3}.$$

Hence, we conclude that the algorithm operates in the claimed running time. $\blacktriangleleft$

This concludes the proof of Lemma 3.1.7 by Claims 3.1.7.1 and 3.1.7.2. $\square$

Next we include a second type of addition.

3.1.3 Standard Addition

In this section we introduce a second method to compute the addition; this method contributes differently to the running time compared to the previous result about addition.

To understand the difference, we first revisit the addition from Lemma 3.1.7 which we denote by the domain $\mathbb{L}$ of dimension λ . For this addition, the contribution to the running time is larger than the domain size. More precisely, the addition contributes a factor of $\mathbb{L}^{\log 3} \cdot 3^\lambda$. However, the domain does not significantly contribute to the computation of the prime numbers and roots of unity, which is represented by the terms depending on M and thus, on $\overline{M}$. To be more precise, the computation of the primes and roots of unity is independent of the domain $\mathbb{L}$ which allows us that the domain might contain large numbers.

With the addition we introduce in the following, denoted by the domain $\mathbb{A}$ of dimension α , we optimize for the opposing goals. That is, the running time depends linearly (up to a logarithmic factor) on the size of the domain $\mathbb{A}$, but each dimension can possibly contribute to the computation of the prime numbers and roots of unity which are described by the factor $\overline{M}$ in the running time.

The method with domain $\mathbb{L}$ is better when the domain contains potentially large values but contains only few dimensions. The method with domain $\mathbb{A}$ is better when the domain contains small values but consists of many dimensions with few different sizes.

To implement the addition from domain $\mathbb{A}$, we use the domain $\mathbb{L}$ to implement a check sum. Then, we can use the cyclic part of the convolution to compute the addition. Together with the check sum this ensures that no overflow appears.

Lemma 3.1.8. *Consider three domains of vectors $\mathbb{S}$, $\mathbb{A}$, and $\mathbb{L}$*

- *with $\mathbb{S} = \mathbb{Z}_{z_1} \times \cdots \times \mathbb{Z}_{z_\sigma}$ for the first domain,*
- *with $\mathbb{A} = [0 \dots q_1) \times \cdots \times [0 \dots q_\alpha)$ for the second domain, and set $A_+ = \sum_{i=1}^\alpha q_i$, and*
- *with $\mathbb{L} = [0 \dots x_1) \times \cdots \times [0 \dots x_\lambda)$ for the third domain.*

Denote by $f_1, f_2: \mathbb{S} \times \mathbb{A} \times \mathbb{L} \rightarrow \mathbb{N}$ two functions that are given as table representations.

We denote by $f: \mathbb{S} \times \mathbb{A} \times \mathbb{L} \rightarrow \mathbb{N}$ the convolution of f_1 and f_2 that satisfies, for all $s \in \mathbb{S}$, $a \in \mathbb{A}$, and $\ell \in \mathbb{L}$,

$$f(s, a, \ell) = \sum_{s_1+s_2=s} \sum_{a_1+a_2=a} \sum_{\ell_1+\ell_2=\ell} f_1(s_1, a_1, \ell_1) \cdot f_2(s_2, a_2, \ell_2). \quad (3.1.5)$$

If U denotes a non-negative integer such that all images of f_1 , f_2 and f are contained in $[0 \dots U)$, then the function f can be computed in time

$$X \cdot \log X \cdot (\log U + \overline{M})^{\mathcal{O}(1)} + (\log U + \exp(\overline{M}))^{\mathcal{O}(1)}$$

with $X = |\mathbb{S}| \cdot |\mathbb{A}| \cdot (A_+ \cdot |\mathbb{L}|)^{\log 3} \cdot 3^\lambda$ and $\overline{M} = \prod_{m \in M} m$ where $M = \{z_1, \dots, z_\sigma, q_1, \dots, q_\alpha, 3\}$.

Proof. We follow the ideas of van Rooij behind the proof of Lemma 10 in [130]. As already mentioned, we again use the cyclic part of the convolution for the computation. Without any further compensation, we lose the information when the summation of two values “overflows”.

To detect that no such overflow appears, we introduce a check sum that is computed in the domain $\mathbb{L}$ of the convolution.

The algorithm to compute the convolution f works as follows.

1. We define two new functions $g_1, g_2: \mathbb{S} \times \mathbb{A} \times [0 \dots A_+] \times \mathbb{L} \rightarrow \mathbb{N}$ which have one new position for the mentioned check sum. Formally, for all choices of $s \in \mathbb{S}$, $a \in \mathbb{A}$, $a_+ \in [0 \dots A_+)$, and $\ell \in \mathbb{L}$, we set

$$g_1(s, a, a_+, \ell) := \begin{cases} f_1(s, a, \ell) & \text{if } \|a\| = a_+, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$g_2(s, a, a_+, \ell) := \begin{cases} f_2(s, a, \ell) & \text{if } \|a\| = a_+, \\ 0 & \text{otherwise,} \end{cases}$$

where we write $\|a\| = \sum_{i=1}^{\alpha} a[i]$ for the 1-norm of the vector a and the sum is computed over the integers, i.e., not modulo any number.

2. We use Lemma 3.1.7 to compute the function $g: \mathbb{S} \times \mathbb{A} \times [0 \dots A_+] \times \mathbb{L} \rightarrow \mathbb{N}$ as the convolution of the two functions g_1 and g_2 where we use $\mathbb{S} \times \mathbb{A}$ as the first domain and $[0 \dots A_+) \times \mathbb{L}$ for the second domain of the convolution.
3. We compute the function $f: \mathbb{S} \times \mathbb{A} \times \mathbb{L} \rightarrow \mathbb{N}$ based on the function g . For this we iterate over all choices of $s \in \mathbb{S}$, $a \in \mathbb{A}$, and $\ell \in \mathbb{L}$, and set

$$f(s, a, \ell) := g(s, a, \|a\|, \ell). \quad (3.1.6)$$

This completes the description of the algorithm with the function f as output.

As a next step, we prove the correctness, that is, we show that the computed function f is indeed the convolution of the two functions f_1 and f_2 .

Claim 3.1.8.1. *The function f from Equation (3.1.6) satisfies Equation (3.1.5).*

Proof of Claim. We fix three values $s \in \mathbb{S}$, $a \in \mathbb{A}$, and $\ell \in \mathbb{L}$ in the following. From the definition of the function f in Equation (3.1.6) and the computation of the convolution g by Lemma 3.1.7, we get

$$\begin{aligned} f(s, a, \ell) &= g(s, a, \|a\|, \ell) \\ &= \sum_{s_1+s_2 \equiv s} \sum_{a_1+a_2 \equiv a} \sum_{\bar{a}_1+\bar{a}_2=\|a\|} \sum_{\ell_1+\ell_2=\ell} g_1(s_1, a_1, \bar{a}_1, \ell_1) \cdot g_2(s_2, a_2, \bar{a}_2, \ell_2). \end{aligned}$$

Now observe that by the definition of g_1 and g_2 , the value of the functions g_j is 0 whenever $\|a_j\| \neq \bar{a}_j$ for $j = 1, 2$. Hence, we can restrict the choices of $\bar{a}_1$ and $\bar{a}_2$ such that the corresponding sum is only over those values which are not a priori 0. From this we get

$$f(s, a, \ell) = \sum_{s_1+s_2 \equiv s} \sum_{a_1+a_2 \equiv a} \sum_{\substack{\bar{a}_1+\bar{a}_2=\|a\| \\ \text{s.t. } \bar{a}_1=\|a_1\|, \\ \bar{a}_2=\|a_2\|}} \sum_{\ell_1+\ell_2=\ell} g_1(s_1, a_1, \bar{a}_1, \ell_1) \cdot g_2(s_2, a_2, \bar{a}_2, \ell_2)$$

and, by the definition of g_1 and g_2 , it follows that

$$f(s, a, \ell) = \sum_{s_1+s_2 \equiv s} \sum_{\substack{a_1+a_2 \equiv a \\ \|a_1\| + \|a_2\| = \|a\|}} \sum_{\ell_1+\ell_2=\ell} f_1(s_1, a_1, \ell_1) \cdot f_2(s_2, a_2, \ell_2). \quad (3.1.7)$$

We claim that, for all valid choices of $a_1, a_2 \in \mathbb{A}$ such that $a_1 + a_2 \equiv a$ and $\|a_1\| + \|a_2\| = \|a\|$, it also holds that $a_1 + a_2 = a$, that is, the equivalence is actually equality.

Indeed, as the summation for the vectors a_1 and a_2 is the standard addition, the sum can only increase. Therefore, for all positions $i \in [1.. \alpha]$, the vectors a_1 and a_2 satisfy $a_1[i] + a_2[i] \geq a[i]$. If we assume that for at least one position the inequality is strict, then we immediately get that

$$\|a_1\| + \|a_2\| = \sum_{i=1}^{\alpha} a_1[i] + \sum_{i=1}^{\alpha} a_2[i] = \sum_{i=1}^{\alpha} a_1[i] + a_2[i] > \sum_{i=1}^{\alpha} a[i] = \|a\|$$

which contradicts our assumption. Hence, we conclude that $a_1 + a_2 = a$.

With this result we can rewrite Equation (3.1.7) and get

$$\begin{aligned} f(s, a, \ell) &= \sum_{s_1+s_2 \equiv s} \sum_{\substack{a_1+a_2 \equiv a \\ \|a_1\| + \|a_2\| = \|a\|}} \sum_{\ell_1+\ell_2=\ell} f_1(s_1, a_1, \ell_1) \cdot f_2(s_2, a_2, \ell_2) \\ &= \sum_{s_1+s_2 \equiv s} \sum_{a_1+a_2=a} \sum_{\ell_1+\ell_2=\ell} f_1(s_1, a_1, \ell_1) \cdot f_2(s_2, a_2, \ell_2) \end{aligned}$$

which precisely corresponds to the definition of the convolution from Equation (3.1.5). $\triangleleft$

It remains to prove that the algorithm operates in the claimed running time.

Claim 3.1.8.2. *The algorithm takes time $X \cdot \log X \cdot (\log U + \overline{M})^{\mathcal{O}(1)} + (\log U + \exp(\overline{M}))^{\mathcal{O}(1)}$ to compute the function f where $X = |\mathbb{S}| \cdot |\mathbb{A}| \cdot (A_+ \cdot |\mathbb{L}|)^{\log 3} \cdot 3^\lambda$.*

Proof of Claim. The first step of our algorithm computes the functions g_1 and g_2 based on the given values of f_1 and f_2 . The last step of the algorithm recovers the function f from g_2 . For both steps we iterate over all possible combination for s, a, a_+ , and ℓ . As there are $|\mathbb{S}| \cdot |\mathbb{A}| \cdot A_+ \cdot |\mathbb{L}|$ such combinations, this bounds the number of arithmetic operations for these two steps where each operation takes time $\log^{\mathcal{O}(1)} U$. It remains to analyze the computation of the function g in the second step.

From the definition of our algorithm, we know that $\mathbb{S} \times \mathbb{A}$ and $[0..A_+) \times \mathbb{L}$ are the two domains for the application of Lemma 3.1.7. Moreover, the dimension of the second domain is $\lambda + 1$. Hence, from Lemma 3.1.7, we know that the function g can be computed in time

$$Y \cdot \log Y \cdot (\log U + \overline{M})^{\mathcal{O}(1)} + (\log U + \exp(\overline{M}))^{\mathcal{O}(1)}$$

where

$$Y = |\mathbb{S} \times \mathbb{A}| \cdot |[0..A_+) \times \mathbb{L}|^{\log 3} \cdot 3^{\lambda+1}$$

and $\overline{M} = \prod_{m \in M} m$ with $M = \{z_1, \dots, z_\sigma, q_1, \dots, q_\alpha, 3\}$ because of the choice of the domains.

As this dominates the running time of the first and last step of the algorithm, this bounds the total running time of the algorithm. $\triangleleft$

The combination of Claims 3.1.8.1 and 3.1.8.2 concludes the proof of Lemma 3.1.8. $\square$

As a last step, we include the truncated addition.

3.1.4 Truncated Addition

As a further extension of the previous results we now consider the proof of Main Theorem 1.1 which extends our previous result and the result by van Rooij by the truncated addition. Truncated addition is a variant of normal addition where we have some maximum element.

We note that the techniques we use in the following have already been introduced by van Rooij to obtain faster algorithms for (σ, ρ) -DOMSET [130]. In contrast to the result there, we present the final algorithm in a self-contained way which allows us to use this result directly for other problems as well.

We follow the ideas of the proof of Theorem 2 in [130] which involves zeta- and Möbius-transforms to obtain the improved running time when we also allow a truncated addition. Our main goal is to replace the truncated addition by regular addition and then use the previous result from Lemma 3.1.8.

Before we start, we first restate the definition of truncated addition.

Definition 3.1.1. For some non-negative integer $t \in \mathbb{N}$, consider the set $T = [0 .. t] \cup \{\top\}$. For all $u, v \in U$, the truncated addition of u and v with respect to t , denoted by $u \oplus_t v$ is defined as

$$u \oplus_t v := \begin{cases} \top & \text{if } \top \in \{u, v\} \text{ or } u + v > t, \\ u + v & \text{otherwise.} \end{cases}$$

If the value of t is clear from the context, we omit t from the notation and just write $u \oplus v$.

In order to define the zeta- and Möbius-transform, we need a partial ordering. This is what we define next.

Definition 3.1.9 (Partial Order for Truncated Addition). Denote by $U = \mathbb{N} \cup \{\top\}$ the set of natural numbers that is extended by the special symbol $\top$.

We define the reflexive partial ordering $\preceq$ on U as $u \preceq u$ and $u \preceq \top$, for all $u \in U$, and undefined otherwise.

For all $n \in \mathbb{N}$, we extend this ordering to the set $[0 .. n] \cup \{\top\}$ by using the same ordering as for U .

Moreover, by slightly abusing notation, we extend the ordering $\preceq$ in the natural way to vectors, that is, for some non-negative integer $k \geq 0$, and two vectors $v_1, v_2 \in U^k$, we write $v_1 \preceq v_2$ if and only if $v_1[i] \preceq v_2[i]$ for all $i \in [1 .. k]$. Otherwise, the ordering is undefined.

Observe that this partial ordering does not extend the standard ordering $\leq$. Especially, from $u \preceq v$, for some $u, v \in U$, we conclude that $u = v$ unless $v = \top$.

Definition 3.1.10 (Zeta-Transform; Variant of Definition 6 in [130]). Consider two domains $\mathbb{D}$ and $\mathbb{T}$ where $\mathbb{D}$ is arbitrary and $\mathbb{T} = T_1 \times \cdots \times T_\tau$ is a τ -dimensional domain such that, for all $i \in [1 .. \tau]$, we have $T_i = [0 .. t_i] \cup \{\top\}$ for some appropriately chosen integer $t_i \in \mathbb{N}$. Moreover, denote by $\mathbb{F}$ some commutative group with addition.

For a function $f: \mathbb{T} \times \mathbb{D} \rightarrow \mathbb{F}$ we define the zeta-transform of f as the function $\zeta(f): \mathbb{T} \times \mathbb{D} \rightarrow \mathbb{F}$ where, for all $t \in \mathbb{T}$ and all $d \in \mathbb{D}$, we set

$$\zeta(f)(d, t) := \sum_{u \preceq t} f(d, u).$$

Before we use the zeta-transform of a function to compute the convolution, we first show that we can actually compute the zeta-transform and can recover a function from its zeta-transform. The function recovering the function f from $\zeta(f)$ is usually referred to as the Möbius-transform.

Lemma 3.1.11 (Compare [130, Proposition 7]). *Consider a function $f: \mathbb{D} \times \mathbb{T} \rightarrow \mathbb{F}$ as defined in Definition 3.1.10 (especially, τ denotes the dimension of $\mathbb{T}$). Then, both of the following results hold.*

1. *Given the memoization table of the function f , we can compute the zeta-transform $\zeta(f)$ of f with $\mathcal{O}(|\mathbb{D}| \cdot |\mathbb{T}| \cdot \tau)$ arithmetic operations.*
2. *Given the memoization table of $\zeta(f)$, we can reconstruct the function f with $\mathcal{O}(|\mathbb{D}| \cdot |\mathbb{T}| \cdot \tau)$ arithmetic operations.*

Proof. We use a recursive procedure based on the dimension τ of the domain $\mathbb{T}$ to compute the zeta-transform of the function f and to recover the function from its zeta-transform.

The case with $\tau = 0$ serves as base case for both computations. As in this case the zeta-transform agrees with the original function, it suffices to copy the given memoization table to construct the zeta-transform $\zeta(f)$ and to recover the function f .

For values where $\tau > 0$, we consider both cases separately. We first consider the computation of the zeta-transform and then the computation to recover the original function.

Computing the Zeta-Transform. Consider a function $f: T_1 \times \cdots \times T_\tau \times T_{\tau+1} \times \mathbb{D} \rightarrow \mathbb{F}$ where, for all $i \in [1.. \tau + 1]$, we have $T_i = [0.. t_i) \cup \{\top\}$ for some appropriately chosen integer $t_i \in \mathbb{N}$.

We define a new function $f': (T_1 \times \cdots \times T_\tau) \times (T_{\tau+1} \times \mathbb{D}) \rightarrow \mathbb{F}$ where, for all $v \in T_1 \times \cdots \times T_\tau$, all $v_{\tau+1} \in T_{\tau+1}$, and all $d \in \mathbb{D}$, we set

$$f'(v, v_{\tau+1}, d) := \begin{cases} \sum_{u \in T_{\tau+1}} f(v, u, d) & \text{if } v_{\tau+1} = \top, \\ f(v, v_{\tau+1}, d) & \text{if } v_{\tau+1} \neq \top. \end{cases}$$

Then, we recurse and compute the zeta-transform $\zeta(f')$ of f' and return this function as the zeta-transform of the original function f . This concludes the description of the algorithm.

Before we proceed to the second statement, we prove the correctness and analyze the running time.

Claim 3.1.11.1. *The algorithm is correct, that is, the function $\zeta(f')$ satisfies $g = \zeta(f)$.*

Proof of Claim. We prove the claim by induction. The case $\tau = 0$ follows trivially as the function does not change. Now assume that the algorithm is correct for some arbitrary but fixed $\tau \geq 0$.

By the definition of the zeta-transform from Definition 3.1.10 we know that

$$\zeta(f)(v_1, \dots, v_\tau, v_{\tau+1}, d) = \sum_{u_1 \preceq v_1} \cdots \sum_{u_\tau \preceq v_\tau} \sum_{u_{\tau+1} \preceq v_{\tau+1}} f(u_1, \dots, u_\tau, u_{\tau+1}, d)$$

for all $(v_1, \dots, v_\tau, v_{\tau+1}) \in \mathbb{T}$ and $d \in \mathbb{D}$.

By the definition of the function f' and the definition of the partial ordering $\preceq$, we conclude that $\zeta(f) = \zeta(f')$ which proves the claim by the induction hypothesis. $\triangleleft$

Now we analyze the running time of this procedure.

Claim 3.1.11.2. *The zeta-transform of f can be computed in time $\mathcal{O}(|\mathbb{T}| \cdot |\mathbb{D}| \cdot \tau)$.*

Proof of Claim. We observe that the function f' agrees with the function f on almost all inputs. More formally, for all $v \in T_1 \times \cdots \times T_\tau$, all $v_{\tau+1} \in T_{\tau+1} \setminus \{\top\}$, and all $d \in \mathbb{D}$, we have $f'(v, v_{\tau+1}, d) = f(v, v_{\tau+1}, d)$.

For all other positions, that is, for all $v \in T_1 \times \cdots \times T_\tau$, and all $d \in \mathbb{D}$, the computation of $f(v, \top, d)$ requires $|T_{\tau+1}| - 1$ additions to compute the sum.

Therefore, the computation of the function f' takes time

$$|T_1 \times \cdots \times T_\tau| \cdot |T_{\tau+1} \setminus \{\top\}| \cdot |\mathbb{D}| + |T_1 \times \cdots \times T_\tau| \cdot |\mathbb{D}| \cdot |T_{\tau+1}|.$$

By the recursive design of the algorithm, the computation of a function f' takes place τ times which concludes the proof together with the previous observation. $\triangleleft$

Now we proceed to the second part of the statement, that is, we show how to recover a function from the zeta-transform.

Recovering the Function. Assume we are given a function $g: T_1 \times \cdots \times T_\tau \times T_{\tau+1} \times \mathbb{D} \rightarrow \mathbb{F}$ that is the zeta-transform of some unknown function f . As before we assume that, for all $i \in [1.. \tau + 1]$, we have $T_i = [0.. t_i] \cup \{\top\}$ for some appropriately chosen integer $t_i \in \mathbb{N}$.

We interpret g as $g: (T_1 \times \cdots \times T_\tau) \times (T_{\tau+1} \times \mathbb{D}) \rightarrow \mathbb{F}$, that is, we treat g as a function with only τ dimensions of truncated addition, and recurse to recover a function g' from it.

Then, based on this function g' , we recover the function $f: \mathbb{T} \times \mathbb{D} \rightarrow \mathbb{F}$ where, for all $v \in T_1 \times \cdots \times T_\tau$, all $v_{\tau+1} \in T_{\tau+1}$, and all $d \in \mathbb{D}$, we set

$$f(v, v_{\tau+1}, d) := \begin{cases} g'(v, v_{\tau+1}, d) - \sum_{u \in T_{\tau+1} \setminus \{\top\}} f(v, u, d) & \text{if } v_{\tau+1} = \top, \\ g'(v, v_{\tau+1}, d) & \text{if } v_{\tau+1} \neq \top. \end{cases}$$

We return f as the function that is recovered from the zeta-transform g .

As a next step, we show that this second algorithm is correct.

Claim 3.1.11.3. *The algorithm is correct, that is, the function f recovered by the algorithm from the zeta-transform g is such that $\zeta(f) = g$.*

Proof of Claim. We prove the claim by induction. The case $\tau = 0$ follows trivially as the function and its zeta-transform are equal. Now assume that the algorithm is correct for some arbitrary but fixed value $\tau \geq 0$.

For the induction step, consider some $v \in T_1 \times \cdots \times T_\tau$, and some $d \in \mathbb{D}$. By the definition of the zeta-transform from Definition 3.1.10, the partial ordering $\preceq$, and the definition of the function f , we know that

$$\begin{aligned} \zeta(f)(v, \top, d) &= \sum_{u \preceq v} \sum_{u_{\tau+1} \preceq \top} f(u, u_{\tau+1}, d) \\ &= \sum_{u \preceq v} \sum_{u_{\tau+1} \in T_{\tau+1}} f(u, u_{\tau+1}, d) \\ &= \sum_{u \preceq v} \left(\sum_{u_{\tau+1} \in T_{\tau+1} \setminus \{\top\}} g'(u, u_{\tau+1}, d) \right) + \left(g'(u, \top, d) - \sum_{w_{\tau+1} \in T_{\tau+1} \setminus \{\top\}} f(u, w_{\tau+1}, d) \right) \\ &= \sum_{u \preceq v} g'(u, \top, d). \end{aligned}$$

Next, consider some vector $v \in T_1 \times \cdots \times T_\tau$, some value $v_{\tau+1} \in T_{\tau+1} \setminus \{\top\}$, and some vector $d \in \mathbb{D}$. Again by the definition of the zeta-transform, the partial ordering $\preceq$, and the definition of the function f , we know that

$$\begin{aligned} \zeta(f)(v, v_{\tau+1}, d) &= \sum_{u \preceq v} \sum_{u_{\tau+1} \preceq v_{\tau+1}} f(u, u_{\tau+1}, d) \\ &= \sum_{u \preceq v} f(u, v_{\tau+1}, d) = \sum_{u \preceq v} g'(u, v_{\tau+1}, d). \end{aligned}$$

Combining the previous two observations, we conclude that the zeta-transform of f is equal to the zeta-transform of g' . As the zeta-transform of g' is equal to g by the induction hypothesis, this concludes the proof. $\triangleleft$

As a last step we analyze the running time of the algorithm.

Claim 3.1.11.4. *Given the zeta-transform, we can recover the original function in time $\mathcal{O}(|\mathbb{T}| \cdot |\mathbb{D}| \cdot \tau)$.*

Proof of Claim. Similar to the earlier bound on the running time, we observe that the computation of the value $f(v, \top, d)$ takes time $\mathcal{O}(|T_{\tau+1}|)$ each. As all other values can be computed in time $\mathcal{O}(1)$ each, the entries running time can be bounded by

$$|T_1 \times \cdots \times T_\tau| \cdot |T_{\tau+1} \setminus \{\top\}| \cdot |\mathbb{D}| + |T_1 \times \cdots \times T_\tau| \cdot |\mathbb{D}| \cdot |T_{\tau+1}|.$$

As this is bounded by $|\mathbb{T}| \cdot |\mathbb{D}|$ which stays invariant for all τ recursive calls, the claim follows. $\triangleleft$

This finishes the proof of Lemma 3.1.11 about the computation of the zeta-transform from a given function and the recovery of the original function from its given zeta-transform. $\square$

With Lemma 3.1.11 we have direct access to the zeta-transforms of the involved functions and can switch between the function and their zeta-transform. Hence, to compute the convolution of two functions f_1 and f_2 , we can first compute the zeta-transform of f_1 and f_2 , then compute the convolution of these two new functions, and finally, recover the actual convolution function from its zeta-transform.

However, this still does not allow us to directly apply the convolution we have seen earlier. Indeed, we still have to deal with the truncated part of the addition and especially with the special symbol $\top$.

The main idea is to fix a subset $Z \subseteq [1.. \tau]$ of the dimensions from $\mathbb{T}$ where the symbol $\top$ does not appear as value of the function. Then, we can “hard-code” (the complement of) these positions into the computation procedure which allows us to remove the appearance of $\top$ from the computation. By iterating over all choices of the set Z , we can then recover the original function.

With this idea in mind, we move to the main result of this section, that is, the convolution described in Main Theorem I.1.

Main Theorem I.1 (General Convolution). *Consider a domain $\mathbb{D}$ composed of the four subdomains $\mathbb{S}$, $\mathbb{A}$, $\mathbb{L}$, and $\mathbb{T}$ such that $\mathbb{D} := \mathbb{S} \times \mathbb{A} \times \mathbb{L} \times \mathbb{T}$*

- *with $\mathbb{S} = \mathbb{Z}_{z_1} \times \cdots \times \mathbb{Z}_{z_\sigma}$ for the first domain,*
- *with $\mathbb{A} = [0..q_1) \times \cdots \times [0..q_\alpha)$ for the second domain and define $A_+ := \sum_{i=1}^\alpha q_i$,*
- *with $\mathbb{L} = [0..x_1) \times \cdots \times [0..x_\lambda)$ for the third domain, and*
- *with $\mathbb{T} = [0..y_1) \cup \{\top\} \times \cdots \times [0..y_\tau) \cup \{\top\}$ for the fourth domain and define $T_+ := \sum_{i=1}^\tau y_i$.*

Denote by $f_1, f_2: \mathbb{D} \rightarrow \mathbb{N}$ two functions that are given as table representations.

We denote by $f: \mathbb{D} \rightarrow \mathbb{N}$ the convolution of f_1 and f_2 that satisfies, for all $(s, a, \ell, t) \in \mathbb{S} \times \mathbb{A} \times \mathbb{L} \times \mathbb{T} = \mathbb{D}$,

$$f(s, a, \ell, t) = \sum_{s_1+s_2 \equiv s} \sum_{a_1+a_2=a} \sum_{\ell_1+\ell_2=\ell} \sum_{t_1 \oplus t_2=t} f_1(s_1, a_1, \ell_1, t_1) \cdot f_2(s_2, a_2, \ell_2, t_2).$$

If U denotes a non-negative integer such that all images of f_1 , f_2 and f are contained in $[0..U)$, then the function f can be computed in time

$$X \cdot \log X \cdot (\log U + \overline{M})^{\mathcal{O}(1)} + 2^\tau \cdot (\log U + \exp(\overline{M}))^{\mathcal{O}(1)}$$

with $X = |\mathbb{D}| \cdot |\mathbb{L}|^{\log(3/2)} \cdot (A_+ + T_+)^{\log(3)} \cdot 3^\lambda$ and $\overline{M} = \prod_{m \in M} m$ where $M = \{z_1, \dots, z_\sigma, q_1, \dots, q_\alpha, 3, y_1, \dots, y_\tau, 3\}$.

Proof. The algorithm, that computes the convolution of the two functions f_1 and f_2 , works as follows:

1. We invoke Lemma 3.1.11 to compute the zeta-transform of the two functions f_1 and f_2 .
2. Iterate over all subsets $Z \subseteq [1.. \tau]$ and perform the following three steps.

3 A Toolkit for Improved Algorithms

- a) We define $\mathbb{T}^Z := \times_{i \in Z} [0 \dots y_i]$ as the product of all those dimensions $T_i \setminus \{\top\} = [0 \dots y_i]$ which are contained in Z .
- b) We define two new functions $g_1^Z, g_2^Z: \mathbb{S} \times \mathbb{A} \times \mathbb{L} \times \mathbb{T}^Z \rightarrow \mathbb{N}$ based on the zeta-transform of f_1 and f_2 as follows. For all $s \in \mathbb{S}$, $a \in \mathbb{A}$, $\ell \in \mathbb{L}$, and $t \in \mathbb{T}$ with $t[i] = \top$ if and only if $i \notin Z$, we set

$$g_1^Z(s, a, \ell, t[Z]) := \zeta(f_1)(s, a, \ell, t) \quad (3.1.8)$$

and

$$g_2^Z(s, a, \ell, t[Z]) := \zeta(f_2)(s, a, \ell, t). \quad (3.1.9)$$

- c) We use Lemma 3.1.8 to compute the function $g^Z: \mathbb{S} \times \mathbb{A} \times \mathbb{L} \times \mathbb{T}^Z \rightarrow \mathbb{N}$ as the convolution of g_1^Z and g_2^Z where $\mathbb{T}^Z$ is used as part of the convolution over standard addition, that is, $\mathbb{A} \times \mathbb{T}^Z$ forms the *single* subdomain $\mathbb{A}$ in the application of Lemma 3.1.8.
3. We combine the functions g^Z for all possible choices of Z to compute the function $g: \mathbb{S} \times \mathbb{A} \times \mathbb{L} \times \mathbb{T} \rightarrow \mathbb{N}$. For all $s \in \mathbb{S}$, $a \in \mathbb{A}$, $\ell \in \mathbb{L}$, and $t \in \mathbb{T}$, we set
- $$g(s, a, \ell, t) := g^{Z(t)}(s, a, \ell, t[Z(t)]) \quad \text{where} \quad Z(t) := \{z \in [1 \dots \tau] \mid t[z] \neq \top\}.$$
4. We apply Lemma 3.1.11 to the function g to recover the function $f: \mathbb{S} \times \mathbb{A} \times \mathbb{L} \times \mathbb{T} \rightarrow \mathbb{N}$ under the assumption that g is the zeta-transform of f .

This concludes the description of the algorithm with output f .

Next, we prove the correctness of this algorithm. For this we first define some notation.

For a set $Z \subseteq [1 \dots \tau]$, we define $\mathbb{T}[Z]$ as the restriction of $\mathbb{T}$ to the dimensions from Z . That is, we formally set $\mathbb{T}[Z] := \times_{i \in Z} T_i$. It is crucial to observe that $\mathbb{T}^Z$ and $\mathbb{T}[Z]$ are *not* the same sets but $\mathbb{T}^Z \subseteq \mathbb{T}[Z]$ as the latter set allows for the appearance of the symbol $\top$ in the vectors.

In the following we use the operator $\sqcup$ to combine two vectors into a new vector in the natural way. We refer to Definition 2.1.4 for a formal definition.

We start with the following structural observation.

Claim 3.1.11.5. *Fix an arbitrary set $Z \subseteq [1 \dots \tau]$. For all $s \in \mathbb{S}$, $a \in \mathbb{A}$, $\ell \in \mathbb{L}$, and $t \in \mathbb{T}$ with $t[z] \neq \top$ for all $z \in Z$ and $t[z] = \top$ for all $z \in \bar{Z}$, the following holds:*

$$\zeta(f)(s, a, \ell, t) = \sum_{\substack{s_1 + s_2 = s \\ a_1 + a_2 = a \\ \ell_1 + \ell_2 = \ell}} \sum_{\substack{u_1, u_2 \in \mathbb{T}^Z \text{ s.t.} \\ u_1 + u_2 = t[Z]}} \zeta(f_1)(s_1, a_1, \ell_1, u_1 \sqcup t[\bar{Z}]) \cdot \zeta(f_2)(s_2, a_2, \ell_2, u_2 \sqcup t[\bar{Z}]) \quad (3.1.10)$$

where $\bar{Z} = [1 \dots \tau] \setminus Z$ denotes the complement of the set Z with respect to $[1 \dots \tau]$.

Proof of Claim. To simplify notation, we assume that there exists some integer $k \in [0 .. \tau]$ such that $Z = [1 .. k]$. Then, for two vectors $t_1 \in \mathbb{T}[Z]$ and $t_2 \in \mathbb{T}[\bar{Z}]$, we omit the operator $\sqcup$ from the notation $t_1 \sqcup t_2$ and just write $t_1 t_2$ instead. Moreover, we write $f(\star, \star, \star, t_1, t_2)$ instead of $f(\star, \star, \star, t_1 t_2)$ and similarly for f_1 and f_2 .

Based on the definition of the partial ordering $\preceq$ and the zeta-transform from Definitions 3.1.9 and 3.1.10, respectively, we rewrite the left-hand side of Equation (3.1.10) as

$$\begin{aligned} \zeta(f)(s, a, \ell, t) &= \sum_{\substack{u \in \mathbb{T} \\ u \preceq t}} f(s, a, \ell, u) \\ &= \sum_{\substack{u' \in \mathbb{T}[Z]: \\ u' \preceq t[Z]}} \sum_{\substack{u^\top \in \mathbb{T}[\bar{Z}]: \\ u^\top \preceq t[\bar{Z}]}} f(s, a, \ell, u', u^\top) \\ &= \sum_{u^\top \in \mathbb{T}[\bar{Z}]} f(s, a, \ell, t[Z], u^\top) \end{aligned}$$

where, for the last equation, we use that $t[z] = \top$ if and only if $z \in \bar{Z}$, which implies that (1) the condition $u^\top \preceq t[\bar{Z}]$ is always satisfied and (2) the only possible choice for $u' \in \mathbb{T}[Z]$ such that $u' \preceq t[Z]$ is $u' = t[Z]$ by the definition of the partial ordering $\preceq$.

As a next step, we expand the definition of the function f (where $\oplus$ denotes the truncated addition) to get

$$\begin{aligned} \zeta(f)(s, a, \ell, t) &= \sum_{u^\top \in \mathbb{T}[\bar{Z}]} \sum_{\substack{s_1+s_2 \equiv s \\ a_1+a_2=a \\ \ell_1+\ell_2=\ell}} \sum_{\substack{u_1, u_2 \in \mathbb{T}: \\ u_1 \oplus u_2 = t[Z] u^\top}} f_1(s_1, a_1, \ell_1, u_1) \cdot f_2(s_2, a_2, \ell_2, u_2) \\ &= \sum_{u^\top \in \mathbb{T}[\bar{Z}]} \sum_{\substack{s_1+s_2 \equiv s \\ a_1+a_2=a \\ \ell_1+\ell_2=\ell}} \sum_{\substack{u'_1, u'_2 \in \mathbb{T}[Z]: \\ u'_1 \oplus u'_2 = t[Z]}} \sum_{\substack{u_1^\top, u_2^\top \in \mathbb{T}[\bar{Z}]: \\ u_1^\top \oplus u_2^\top = u^\top}} f_1(s_1, a_1, \ell_1, u'_1, u_1^\top) \cdot f_2(s_2, a_2, \ell_2, u'_2, u_2^\top). \end{aligned}$$

Next, we observe that, for each pair $u_1^\top, u_2^\top \in \mathbb{T}[\bar{Z}]$, there is a unique $u^\top \in \mathbb{T}[\bar{Z}]$ such that $u_1^\top \oplus u_2^\top = u^\top$. Thus, we sum over each such pair exactly once. Hence, we get from the previous equation that

$$\begin{aligned} \zeta(f)(s, a, \ell, t) &= \sum_{\substack{s_1+s_2 \equiv s \\ a_1+a_2=a \\ \ell_1+\ell_2=\ell}} \sum_{\substack{u'_1, u'_2 \in \mathbb{T}[Z]: \\ u'_1 \oplus u'_2 = t[Z]}} \sum_{u_1^\top, u_2^\top \in \mathbb{T}[\bar{Z}]} f_1(s_1, a_1, \ell_1, u'_1, u_1^\top) \cdot f_2(s_2, a_2, \ell_2, u'_2, u_2^\top) \end{aligned}$$

and when we reorder the terms, since $u_1^\top$ and $u_2^\top$ are chosen independently of u'_1 and u'_2 , we get

$$= \sum_{\substack{s_1+s_2 \equiv s \\ a_1+a_2=a \\ \ell_1+\ell_2=\ell}} \sum_{\substack{u'_1, u'_2 \in \mathbb{T}[Z]: \\ u'_1 \oplus u'_2 = t[Z]}} \left(\sum_{u_1^\top \in \mathbb{T}[\bar{Z}]} f_1(s_1, a_1, \ell_1, u'_1, u_1^\top) \right) \cdot \left(\sum_{u_2^\top \in \mathbb{T}[\bar{Z}]} f_2(s_2, a_2, \ell_2, u'_2, u_2^\top) \right).$$

For the next step, we exploit that, by our choice of Z , we have $t[i] \neq \top$ for all $i \in Z$. Hence, for all choices of $u'_1, u'_2 \in \mathbb{T}[Z]$ considered in the previous sum, the vectors u'_1 and u'_2 must also satisfy this property. Hence, it suffices to sum over all vectors $u'_1, u'_2 \in \mathbb{T}^Z$.

Moreover, by this observation, we can apply the definition of the zeta-transform from Definition 3.1.10 for f_1 and f_2 and use that $t[i] = \top$ if and only if $i \in \bar{Z}$ to get

$$\zeta(f)(s, a, \ell, t) = \sum_{\substack{s_1+s_2 \equiv s \\ a_1+a_2=a \\ \ell_1+\ell_2=\ell}} \sum_{\substack{u'_1, u'_2 \in \mathbb{T}^Z: \\ u'_1+u'_2=t[Z]}} \zeta(f_1)(s_1, a_1, \ell_1, u'_1, t[\bar{Z}]) \cdot \zeta(f_2)(s_2, a_2, \ell_2, u'_2, t[\bar{Z}]).$$

This concludes the proof of the structural property. $\blacktriangleleft$

With Claim 3.1.11.5 it is straightforward to prove the correctness of the algorithm.

Claim 3.1.11.6. *The algorithm is correct, that is, the function f from the last step of the algorithm is the convolution of the two functions f_1 and f_2 .*

Proof of Claim. By the correctness of Lemma 3.1.11, we know that we can compute the zeta-transform correctly and can also recover the original function correctly. Hence, it suffices to prove that function g from the third step of the algorithm is the zeta-transform of the convolution of f_1 and f_2 .

For this we fix four index vectors $s \in \mathbb{S}$, $a \in \mathbb{A}$, $\ell \in \mathbb{L}$, and $t \in \mathbb{T}$. We denote by $Z(t)$ the set of those dimensions of t that are not $\top$, that is, formally we define $Z(t) := \{z \in [1.. \tau] \mid t[z] \neq \top\}$. From the computation of g in the third step of the algorithm and the computation of the function g^Z as the convolution of g_1^Z and g_2^Z in the second step, we directly get that

$$\begin{aligned} g(s, a, \ell, t) &= g^{Z(t)}(s, a, \ell, t[Z(t)]) \\ &= \sum_{\substack{s_1+s_2 \equiv s \\ a_1+a_2=a \\ u_1+u_2=t[Z(t)]}} \sum_{\substack{a_1+a_2=a \\ u_1+u_2=t[Z(t)]}} \sum_{\ell_1+\ell_2=\ell} g_1^{Z(t)}(s_1, a_1, \ell_1, u_1) \cdot g_2^{Z(t)}(s_2, a_2, \ell_2, u_2) \\ &= \sum_{\substack{s_1+s_2 \equiv s \\ a_1+a_2=a \\ \ell_1+\ell_2=\ell}} \sum_{u_1+u_2=t[Z(t)]} \zeta(f_1)(s_1, a_1, \ell_1, u_1 \sqcup t[\bar{Z}]) \cdot \zeta(f_2)(s_2, a_2, \ell_2, u_2 \sqcup t[\bar{Z}]) \end{aligned}$$

where the last equality follows from the definition of g_1^Z and g_2^Z as the zeta-transforms of f_1 and f_2 , respectively, and a reordering of the terms. By Claim 3.1.11.5, we get that $g(s, a, \ell, t) = \zeta(f)(s, a, \ell, t)$ which concludes the proof of the correctness. $\blacktriangleleft$

It remains to bound the running time of the procedure. By assumption, we know that the images of the three functions f_1 , f_2 , and f are contained in $[0..U)$.

Claim 3.1.11.7. *The algorithm computes the function f in time*

$$\begin{aligned} &\left(|\mathbb{D}| \cdot |\mathbb{L}|^{\log(3/2)} \cdot (A_+ + T_+)^{\log 3} \cdot 3^\lambda \cdot (\log |\mathbb{D}| + \lambda + \log(A_+ + T_+)) \right) \cdot (\log U + \bar{M})^{\mathcal{O}(1)} \\ &+ 2^\tau \cdot (\log U + \exp(\bar{M}))^{\mathcal{O}(1)}. \end{aligned}$$

Proof of Claim. We first analyze each step of the algorithm individually.

By Lemma 3.1.11, the first step of the algorithm uses

$$\mathcal{O}((|\mathbb{S}| \cdot |\mathbb{A}| \cdot |\mathbb{L}|) \cdot |\mathbb{T}| \cdot \tau) \quad (3.1.11)$$

arithmetic operations to compute the zeta-transform of the two functions f_1 and f_2 .

For the second step, we first fix some set $Z \subseteq [1.. \tau]$. To compute the functions g_1^Z and g_2^Z , the algorithm iterates over all possible values for $s \in \mathbb{S}$, $a \in \mathbb{A}$, $\ell \in \mathbb{L}$, and $t \in \mathbb{T}$ where $t[i] = \top$ if and only if $i \notin Z$. Hence, instead of iterating over all values in $\mathbb{T}$, it suffices to iterate over the values $t \in \mathbb{T}^Z$ as the values of $t[\bar{Z}]$ are fixed by assumption.

With this observation, each of the functions g_1^Z and g_2^Z can be computed with $\mathcal{O}(|\mathbb{S}| \cdot |\mathbb{A}| \cdot |\mathbb{L}| \cdot |\mathbb{T}^Z|)$ operations.

To compute the function g^Z as the convolution of the two functions g_1^Z and g_2^Z , the algorithm uses Lemma 3.1.8 where we treat $\mathbb{T}^Z$ as part of domain $\mathbb{A}$ from Lemma 3.1.8. Hence, this computation takes time

$$\begin{aligned} & \left(|\mathbb{S}| \cdot |\mathbb{A} \times \mathbb{T}^Z| \cdot (A_+ + T_+^Z)^{\log 3} \cdot |\mathbb{L}|^{\log 3} \cdot 3^\lambda \right. \\ & \quad \cdot \left(\log |\mathbb{S}| + \log |\mathbb{A} \times \mathbb{T}^Z| + \log(A_+ + T_+^Z) + \log |\mathbb{L}| + \lambda \right) \cdot (\log U + \bar{M})^{\mathcal{O}(1)} \\ & \quad \left. + (\log U + \exp(\bar{M}))^{\mathcal{O}(1)} \right) \end{aligned}$$

where we set $T_+^Z := \sum_{i \in Z} y_i$.

We first simplify this bound. For this we use that $T_+^Z \leq T_+$ and $|\mathbb{T}^Z| \leq |\mathbb{T}|$. Moreover, we set $Y := |\mathbb{S}| \cdot |\mathbb{A}| \cdot |\mathbb{L}|^{\log 3} \cdot (A_+ + T_+)^{\log 3} \cdot 3^\lambda$. Hence, the running time for a fixed value of Z is bounded by

$$Y \cdot |\mathbb{T}^Z| \cdot (\log Y + \log |\mathbb{T}|) \cdot (\log U + \bar{M})^{\mathcal{O}(1)} + (\log U + \exp(\bar{M}))^{\mathcal{O}(1)}.$$

Next, we bound the running time for the entire second step. Using the previous bound for a fixed set Z , the second step takes time

$$\begin{aligned} & \sum_{Z \subseteq [1.. \tau]} \left(Y \cdot |\mathbb{T}^Z| \cdot (\log Y + \log |\mathbb{T}|) \cdot (\log U + \bar{M})^{\mathcal{O}(1)} + (\log U + \exp(\bar{M}))^{\mathcal{O}(1)} \right) \\ & \leq Y \cdot (\log Y + \log |\mathbb{T}|) \cdot (\log U + \bar{M})^{\mathcal{O}(1)} \cdot \sum_{Z \subseteq [1.. \tau]} |\mathbb{T}^Z| \\ & \quad + 2^\tau \cdot (\log U + \exp(\bar{M}))^{\mathcal{O}(1)}. \end{aligned} \quad (3.1.12)$$

When we unfold the definition of $\mathbb{T}^Z$ in the previous bound, we get

$$\begin{aligned} \sum_{Z \subseteq [1.. \tau]} |\mathbb{T}^Z| &= \sum_{Z \subseteq [1.. \tau]} \left| \bigtimes_{i \in Z} [0.. y_i) \right| \\ &= \sum_{Z \subseteq [1.. \tau]} \prod_{i \in Z} y_i \\ &= \prod_{i \in [1.. \tau]} (y_i + 1) = |\mathbb{T}|. \end{aligned} \quad (3.1.13)$$

When we combine the bound from Equation (3.1.12) with the observation from Equation (3.1.13), we conclude that all functions g^Z can be computed in time

$$Y \cdot |\mathbb{T}| \cdot (\log Y + \log |\mathbb{T}|) \cdot (\log U + \overline{M})^{\mathcal{O}(1)} + 2^\tau \cdot (\log U + \exp(\overline{M}))^{\mathcal{O}(1)} \quad (3.1.14)$$

in total.

For the analysis of the third and forth step, we first observe that, for all choices of Z , the domains of the functions g^Z are pairwise disjoint. Hence, the procedure from the fourth step of the algorithm, which computes the function f based on the combination of the functions g^Z , can directly use the values computed in the second step. Thus, as we use Lemma 3.1.11 in the fourth step to revert the zeta-transform, the number of operations needed is equal to that from the first step.

The final bound on the running time is determined by the maximum of the number of operations from the first step in Equation (3.1.11), which agrees with the bound from the fourth step, and the number of operations from the second step in Equation (3.1.14). With the observation that $\tau \leq T_+$, this concludes the proof. $\blacktriangleleft$

The proof of Main Theorem I.1 now follows from Claims 3.1.11.6 and 3.1.11.7. $\square$

3.2 Representative Sets

The results from Section 3.1 provide efficient procedures to handle the join nodes in dynamic programs on tree decompositions. Since the running time of these procedures is essentially the size of the domain of the involved functions, a significant improvement is quite unlikely. Hence, to improve a dynamic program even further, a completely different approach is needed. One such technique is based on representative sets which allow us to not store all obtained information but only a small *representative set* which is still sufficient to solve the problem correctly.

To define the concepts of representative sets in full generality, we use an approach based on bipartite graphs.

Definition 3.2.1 (*H-Compatibility*). Consider an undirected, bipartite graph $H = (L \dot{\cup} R, E)$ that might not be finite. For two vertices $a \in L$ and $b \in R$, we say that a is H -compatible with b , denoted by $a \sim_H b$, if $(a, b) \in E$.

Based on the notion of compatibility, we define the H -representation of a set.

Definition 3.2.2 (*H-Representation*). Consider an undirected, bipartite graph $H = (L \dot{\cup} R, E)$ that might not be finite.

For two sets $\mathcal{S}, \mathcal{S}' \subseteq L$, we say that $\mathcal{S}'$ H -represents $\mathcal{S}$, denoted by $\mathcal{S}' \subseteq_{H\text{-rep}} \mathcal{S}$, if $\mathcal{S}' \subseteq \mathcal{S}$ and, for every $b \in R$, the sets satisfy that

$$\exists a \in \mathcal{S} : a \sim_H b \quad \text{if and only if} \quad \exists a' \in \mathcal{S}' : a' \sim_H b.$$

A first concrete example for a graph which we use for the compatibility and representation, is the k - r -graph $\mathcal{G}_r^k$ defined as follows.

To emphasize that the graph is bipartite, we denote the edges by tuples where the vertices come from the left and right vertex set of the graph.

Definition 3.2.3 (*k-r-Representation*). Denote by k and r two positive integers.

We define the k - r -graph $\mathcal{G}_r^k = (L \dot{\cup} R, E)$ as the bipartite graph with left side $L := \mathbb{N}^k$, with right side $R := \binom{\mathbb{N}}{r}^k$, and with edges $E := \{(a, B) \in L \times R \mid \forall i \in [1..k] : a[i] \notin B[i]\}$.

By Definition 3.2.2, we denote

- by $\sim_r^k$ the $\mathcal{G}_r^k$ -compatibility, k - r -compatibility for short, and
- by $\subseteq_{r\text{-rep}}^k$ the $\mathcal{G}_r^k$ -representation, k - r -representation for short.

For the notation of k - r -representation, we omit the value k from the notation if $k = 1$.

We summarize our result to compute representative sets by Main Theorem I.2.

Main Theorem I.2. Consider k integer universes $U_1, \dots, U_k$ of size n each. Given a set $\mathcal{S} \subseteq U_1 \times \dots \times U_k$, we can compute a set $\mathcal{S}' \subseteq_{r\text{-rep}}^k \mathcal{S}$ of size $|\mathcal{S}'| \leq (r+1)^{\mathcal{O}(k)}$ in time $|\mathcal{S}| \cdot (r+1)^{\mathcal{O}(k)} \cdot k$.

The proof of Main Theorem I.2 follows from Lemmas 3.2.8 and 3.2.10 which both present more precise bounds on the size of the representative set and the running time to compute them.

Matroids. To compute the representative sets, we use matroids. They allow us to formally state the operations we are using.

Definition 3.2.4 (Matroid). A matroid is a pair $\mathcal{M} = (E, \mathcal{I})$ with ground set E and where we refer to $\mathcal{I} \subseteq 2^E$ as the independent sets of $\mathcal{M}$, satisfying the following three axioms:

1. $\emptyset \in \mathcal{I}$
2. if $I_1 \subseteq I_2$ and $I_2 \in \mathcal{I}$, then $I_1 \in \mathcal{I}$
3. if $I_1, I_2 \in \mathcal{I}$ and $|I_1| < |I_2|$, then there is some $x \in I_2 \setminus I_1$ such that $I_1 \cup \{x\} \in \mathcal{I}$

Because of their general definition, there are many variations of matroids as, for example, uniform matroids, graphical matroids, or transversal matroids, and they are extensively studied. See Chapter 12 in [36] for an introduction of matroids in the context of parameterized algorithm or [110, 134] for a general introduction to the field.

In this work, we only consider uniform matroids as they are sufficient and well-suited to compute representative sets.

Definition 3.2.5 (Uniform Matroid). Let U be some universe with n elements and $r \in \mathbb{N}$. Then, $\mathcal{U}_{r,n} = (U, \binom{U}{\leq r})$ is the uniform matroid of rank r , that is, the matroid over the ground set U where the independent sets are all subsets of U of size at most r .

When we work with matroids, we also need a way to represent them. For this we consider the matrix representation of a matroid which might not exist for all matroids. A matroid $\mathcal{M} = (U, \mathcal{I})$ can be represented if there is a matrix M (over some field) with $|U|$ columns where each column corresponds to one element of the universe U and precisely those sets of columns are *linearly independent* that correspond to some *independent set* $I \in \mathcal{I}$. The rank of the matroid $\mathcal{M}$ is the minimum number of rows of a matrix M representing the matroid $\mathcal{M}$.

It is well-known and straightforward to prove that uniform matroids are representable.

Fact 3.2.6. *Every uniform matroid $\mathcal{U} = (U, \mathcal{I})$ of rank r can be represented by an $r \times |U|$ Vandermonde matrix M (where the first row consists only of 1s) over the finite field with p elements, where p must be larger than $|U|$.*

The basic idea is as follows. Assume that we want to compute a representative set of a set $\mathcal{S} \subseteq U$. In this case we consider some suitable (uniform) matroid with ground set U . We use the properties of the ground set to find a basis which still encodes that the ground set U of the given matroid serves as the set where we chose the representative set $\mathcal{S}$ from.

When it comes to the computation of small representative sets for matroids, several results are already known [60, 61, 93, 122] which are somewhat perpendicular to each other.

- For the case of a single *uniform* matroid, improved size bounds are possible by exploiting the special structure of the matroid (see [60, Section 4] and [122, Section 3]).
- Kratsch and Wahlström consider not only a single matroid but a *product* of matroids. When these underlying matroids are general, they prove an improved size bound for the representative sets [93, Lemma 3.4].

Although the result for general matroids also works in our case, there might be room for improvements when exploiting the structure of the uniform matroids as done in the first case.

It seems to be a natural task to combine these two directions to obtain an even faster algorithm for *products of uniform* matroids. However, the result by Kratsch and Wahlström heavily depends on the algorithm for general matroids, that is, the algorithm is not used as a black box. Therefore, it is unlikely that both approaches can be combined easily.

In the following we consider the case of products of uniform matroids and present two different approaches to compute the representative sets. When it comes to the application of these techniques, we show that both approaches are incomparable to each other, that is, the first method gives the faster algorithm when parameterizing by treewidth while the second method gives the faster algorithm when parameterizing by pathwidth.

Our first approach uses matrix multiplication and other techniques to obtain a better running time.

Lemma 3.2.8. *Consider k integer universes $U_1, \dots, U_k$ of dimension n each. Given a set $\mathcal{S} \subseteq U_1 \times \dots \times U_k$, we can compute a set $\mathcal{S}' \subseteq_{r\text{-rep}}^k \mathcal{S}$ of size $(r+1)^k$ in time $\mathcal{O}(|\mathcal{S}| \cdot (r+1)^{k \cdot (\omega-1)} \cdot k)$ where ω is the matrix multiplication exponent.*

The second approach uses an existing result which computes a representative sets for a single uniform matroid.

Lemma 3.2.10. *Consider k integer universes $U_1, \dots, U_k$ of size n each. Given two sets $\mathcal{S}_1, \mathcal{S}_2 \subseteq U_1 \times \dots \times U_k$, we can compute*

1. *a set $\mathcal{S}' \subseteq_{r\text{-rep}}^k \mathcal{S}_1$ of size $(2r+1)^k \cdot (1+1/2r)^{r \cdot k} \cdot 2^{o(r \cdot k)}$ in time $|\mathcal{S}_1| \cdot (1+1/2r)^{r \cdot k} \cdot 2^{o(r \cdot k)} \cdot \log n$, and*
2. *a set $\mathcal{S}'' \subseteq_{r\text{-rep}}^k \mathcal{S}_1 + \mathcal{S}_2$ of size $(3r/2+1)^k \cdot (1+2/3r)^{r \cdot k} \cdot 2^{o(r \cdot k)}$ in time $|\mathcal{S}_1| \cdot |\mathcal{S}_2| \cdot (1+2/3r)^{r \cdot k} \cdot 2^{o(r \cdot k)} \cdot \log n$.*

Then, both results prove Main Theorem 1.2.

Main Theorem 1.2. *Consider k integer universes $U_1, \dots, U_k$ of size n each. Given a set $\mathcal{S} \subseteq U_1 \times \dots \times U_k$, we can compute a set $\mathcal{S}' \subseteq_{r\text{-rep}}^k \mathcal{S}$ of size $|\mathcal{S}'| \leq (r+1)^{\mathcal{O}(k)}$ in time $|\mathcal{S}| \cdot (r+1)^{\mathcal{O}(k)} \cdot k$.*

Proof. The proof follows from Lemma 3.2.8 and from Lemma 3.2.10 independently. $\square$

We start with the proof of Lemma 3.2.8 in Section 3.2.1, and continue with the proof of Lemma 3.2.10 in Section 3.2.2. Finally, in Section 3.2.3 we define a more specific compatibility graph which we use for our algorithmic results.

3.2.1 Computing Representative Sets via Matrix Multiplication

Our first algorithm is derived from known results for computing representative sets and makes use of matrix multiplication techniques.

Our main idea is to exploit the property that we consider a *product of uniform matroids*. Kratsch and Wahlström consider such a product of matroids in the general setting and show a bound on the size of the representative sets [93, Lemma 3.4]. However, their result does not explicitly provide an algorithm to compute such a representative set.

In the following we focus on uniform matroids and provide a constructive bound, that is, give an algorithm to compute the representative sets. For this we follow the ideas by Fomin et al. [60, Theorem 3.7], in which they consider a *single* uniform matroid, that is, they especially show how to compute a 1 - r -representative set. We use the proof of the result by Fomin et al. as presented in Theorem 12.15 in [36] as the basis for our construction.

Intuitively, we adjust the result by Fomin et al. in two directions. First, we consider multiple (namely k) matroids simultaneously. This modification can be implemented in a straightforward way as the different matroids do not interfere with each other.

Second, we make the result more specific. While for the result by Fomin et al. the given set did not only contain a single integer (for each dimension) but could possibly contain multiple, we only allow a single value from each universe to be given. This allows for a much simpler presentation of the results. Indeed, since the algorithm is heavily based on the computation of determinants, we avoid the usage of minors in favor of directly using the matrix entries.

The idea of our algorithm is to construct a matrix from the given set $\mathcal{S}$. By an appropriate construction of the matrix, the columns correspond to elements of the set $\mathcal{S}$. With this idea it then suffices to compute a column basis of the matrix from which we derive the representative set $\mathcal{S}$. As the size of the basis can be bounded, we directly obtain a bound on the size of the new set $\mathcal{S}'$.

To compute this column basis, we use the following result. As Fact 3.2.7 is based on fast matrix multiplication, we henceforth denote by $\omega \leq 2.371552$ the matrix multiplication exponent [135].

Fact 3.2.7 ([13, Lemma 3.15]). *There is an algorithm that, given an $n \times m$ matrix with $m \leq n$ and with entries from the field $\mathbb{F}$ and weights $w: [1..n] \rightarrow \mathbb{N}$, finds a basis $X \subseteq [1..n]$ of the row space minimizing $w(X)$ with $\mathcal{O}(n \cdot m^{\omega-1})$ arithmetic operations.*

Although, Fact 3.2.7 is originally stated for $\mathbb{F} = \mathbb{F}_2$ only, the procedure does not rely on this fact.

Now we have everything ready to formally state and prove our main result.

Lemma 3.2.8. *Consider k integer universes $U_1, \dots, U_k$ of dimension n each. Given a set $\mathcal{S} \subseteq U_1 \times \dots \times U_k$, we can compute a set $\mathcal{S}' \subseteq_{r\text{-rep}}^k \mathcal{S}$ of size $(r+1)^k$ in time $\mathcal{O}(|\mathcal{S}| \cdot (r+1)^{k \cdot (\omega-1)} \cdot k)$ where ω is the matrix multiplication exponent.*

Proof. Let $\mathcal{M}_1, \dots, \mathcal{M}_k$ denote k uniform matroids, each of rank $r+1$, with integer universes $U_1, \dots, U_k$. Since these matroids are uniform, they are representable by Fact 3.2.6. Hence, there are k matrices $M_1, \dots, M_k$ of dimensions $(r+1) \times |U|$ each that represent the k matroids.

With these preliminaries, the algorithm works as follows.

1. We enumerate all vectors I of $[1 \dots r+1]^k$ in an arbitrary order and denote the ordered vectors by $I_1, \dots, I_{(r+1)^k}$.
2. For all $A \in \mathcal{S}$, we compute an $(r+1)^k$ -dimensional vector v_A over the natural numbers, that is, $v_A \in \mathbb{N}^{(r+1)^k}$, where we set

$$v_A[j] := \prod_{i=1}^k M_i[I_j[i], A[i]] \quad \text{for all } j \in [1 \dots (r+1)^k].$$

From these vectors we construct a matrix Q of dimension $(r+1)^k \times |\mathcal{S}|$, where each vector v_A corresponds to one column of the matrix Q .

3. We use Fact 3.2.7, to compute a column basis B_Q of the matrix Q .
4. We output the set $\mathcal{S}' := \{A \mid v_A \in B_Q\}$ as the new representative set.

This concludes the description of the algorithm. As the next steps, we prove the bound on the running time and the correctness of the algorithm.

Claim 3.2.8.1. *The algorithm operates in time $\mathcal{O}(|\mathcal{S}| \cdot (r+1)^{k \cdot (\omega-1)} \cdot k)$.*

Proof of Claim. Enumerating all vectors in $[1 \dots r+1]^k$ takes time $\mathcal{O}((r+1)^k)$ as we have to iterate over them once.

For each set $A \in \mathcal{S}$, the computation of the corresponding vector v_A takes time $\mathcal{O}((r+1)^k \cdot k)$. Hence, the second step of the algorithm takes time $\mathcal{O}(|\mathcal{S}| \cdot (r+1)^k \cdot k)$ in total.

By Fact 3.2.7 and the fact that we compute a column basis, the computation of the basis from the third step takes time $\mathcal{O}(|\mathcal{S}| \cdot (r+1)^{k \cdot (\omega-1)})$. Note that when $(r+1)^k \leq |\mathcal{S}|$, the algorithm is not needed.

Since the last step operates in time $\mathcal{O}(|B_Q|)$ which is bounded by $\mathcal{O}(|\mathcal{S}|)$, the entire algorithm takes time $\mathcal{O}(|\mathcal{S}| \cdot (r+1)^{k \cdot (\omega-1)} \cdot k)$. $\blacktriangleleft$

Before we prove the correctness, we first introduce some notation. For all sets $C \subseteq U$, we define $M_i[\star, C]$ as the submatrix of M_i that contains only those columns that correspond to elements from C . Similarly, for a vector $I \subseteq [1 \dots r+1]$, we write $M_i[I, \star]$ for the submatrix of M_i that contains only those rows that correspond to elements from I .

For all vectors I_j , we define the “complement” $\bar{I}_j \in \binom{[1 \dots r+1]}{r}$ of I_j such that $\bar{I}_j[i] := [1 \dots r+1] \setminus \{I_j[i]\}$ for all $i \in [1 \dots k]$. With this notation, for all $B \in \binom{U_1}{r} \times \dots \times \binom{U_k}{r}$, we set

$$u_B[j] := \prod_{i=1}^k \det(M_i[\bar{I}_j[i], B[i]]).$$

With these definitions, we start with the following structural property that relates the vectors v_A and u_B to the representative sets.

Claim 3.2.8.2. *For all $A \in U_1 \times \dots \times U_k$ and $B \in \binom{U_1}{r} \times \dots \times \binom{U_k}{r}$, it holds that*

$$A \sim_r^k B \iff 0 \neq \sum_{j=1}^{(r+1)^k} \sigma_j \cdot v_A[j] \cdot u_B[j] \quad \text{where} \quad \sigma_j := (-1)^{\sum_{i=1}^k I_j[i]}. \quad (3.2.1)$$

Proof of Claim. Consider two vectors $A \in U_1 \times \dots \times U_k$ and $B \in \binom{U_1}{r} \times \dots \times \binom{U_k}{r}$ in the following. We first show that these two vectors satisfy

$$A \sim_r^k B \iff \prod_{i=1}^k \det(M_i[\star, A[i]] \mid M_i[\star, B[i]]) \neq 0. \quad (3.2.2)$$

By the coordinate-wise definition of k - r -compatibility, it suffices to show that, for all $i \in [1 \dots k]$, we have

$$A[i] \sim_r B[i] \iff \det(M_i[\star, A[i]] \mid M_i[\star, B[i]]) \neq 0.$$

Indeed, we first observe that $M'_i := (M_i[\star, A[i]] \mid M_i[\star, B[i]])$ is still a Vandermonde matrix and the dimension is $(r+1) \times (r+1)$. Hence, the determinant of this matrix M'_i is

$$\det(M'_i) = \sum_{1 \leq i < j \leq r+1} (x_j - x_i)$$

where the values x_i and x_j are associated with the i th and j th column of the matrix M'_i . Thus, the determinant of the matrix M'_i is 0 if and only if there are two columns that share the same x_i which equivalent to the fact that $A[i] \in B[i]$. Hence, we conclude that Equation (3.2.2) holds.

As a next step, we show that the inequality of the right-hand side of Equation (3.2.2) is equivalent to the right-hand side of Equation (3.2.1). More precisely, we show

$$\sum_{j=1}^{(r+1)^k} \sigma_j \cdot v_A[j] \cdot u_B[j] = (-1)^{(r+1) \cdot k} \cdot \prod_{i=1}^k \det(M_i[\star, A[i]] \mid M_i[\star, B[i]]) \quad (3.2.3)$$

where $\sigma_j := (-1)^{\sum_{i=1}^k I_j[i]}$.

3 A Toolkit for Improved Algorithms

We first analyze the right-hand side. Fix some value $i \in [1 \dots k]$ for this. By the Laplace expansion of the determinant, we have

$$\begin{aligned} & (-1)^{r+1} \cdot \det (M_i[\star, A[i]] \mid M_i[\star, B[i]]) \\ &= \sum_{\ell=1}^{r+1} (-1)^\ell \cdot M_i[\ell, A[i]] \cdot \det (M_i[[1 \dots r+1] \setminus \{\ell\}, B[i]]) \\ &= \sum_{\ell=1}^{r+1} m_{i,\ell} \end{aligned}$$

where, to simplify notation, we set

$$m_{i,j} := (-1)^\ell \cdot M_i[\ell, A[i]] \cdot \det (M_i[[1 \dots r+1] \setminus \{\ell\}, B[i]]).$$

Next, we consider a value $j \in [1 \dots (r+1)^k]$ and analyze the left-hand side of Equation (3.2.3). From the definition of v_A and v_B , we get

$$\begin{aligned} \sigma_j \cdot v_A[j] \cdot u_B[j] &= (-1)^{\sum_{i=1}^k I_j[i]} \cdot \prod_{i=1}^k M_i[I_j[i], A[i]] \cdot \prod_{i=1}^k \det (M_i[\bar{I}_j[i], B[i]]) \\ &= \prod_{i=1}^k (-1)^{I_j[i]} \cdot M_i[I_j[i], A[i]] \cdot \det (M_i[\bar{I}_j[i], B[i]]) \\ &= \prod_{i=1}^k m_{i,I_j[i]}. \end{aligned}$$

Hence, by the laws of distributivity and since the vectors $I_\star$ enumerate all vectors from $[1 \dots r+1]^k$, we have

$$\begin{aligned} \sum_{j=1}^{(r+1)^k} \sigma_j \cdot v_A[j] \cdot u_B[j] &= \sum_{j=1}^{(r+1)^k} \prod_{i=1}^k m_{i,I_j[i]} \\ &= \prod_{i=1}^k \sum_{\ell=1}^{r+1} m_{i,\ell} \\ &= \prod_{i=1}^k (-1)^{r+1} \cdot \det (M_i[\star, A[i]] \mid M_i[\star, B[i]]). \end{aligned}$$

Therefore, Equation (3.2.3) follows, which concludes the proof. $\triangleleft$

With the structural observation from Claim 3.2.8.2 we can now prove the correctness of the algorithm.

Claim 3.2.8.3. *The algorithm is correct, that is, $S' \subseteq_{r\text{-rep}}^k S$ and $|S'| \leq (r+1)^k$.*

Proof of Claim. From the description of the algorithm it directly follows that $\mathcal{S}' \subseteq \mathcal{S}$. Indeed, as the column basis B_Q contains only vectors v_A and for each such vector there is a corresponding element in $\mathcal{S}$, the set $\mathcal{S}'$ contains only elements from $\mathcal{S}$.

Now consider a set $A \in \mathcal{S}$ such that $A \sim_r^k B$ for some suitable set B . From the structural observation in Claim 3.2.8.2, we know that this is equivalent to the inequality

$$0 \neq \sum_{j=1}^{(r+1)^k} \sigma_j \cdot v_A[j] \cdot u_B[j]. \quad (3.2.4)$$

As we are given a column basis B_Q for the matrix Q , there are values $\lambda_{A'}$ for all $A' \in \mathcal{S}'$ such that we can write the vector v_A as

$$v_A = \sum_{A' \in \mathcal{S}'} \lambda_{A'} v_{A'}.$$

Hence, we can substitute this in the sum of Equation (3.2.4), to get

$$\begin{aligned} 0 &\neq \sum_{j=1}^{(r+1)^k} \sigma_j \cdot v_A[j] \cdot u_B[j] \\ &= \sum_{j=1}^{(r+1)^k} \sum_{A' \in \mathcal{S}'} \lambda_{A'} \cdot \sigma_j \cdot v_{A'}[j] \cdot u_B[j] \\ &= \sum_{A' \in \mathcal{S}'} \lambda_{A'} \cdot \sum_{j=1}^{r^k} \sigma_j \cdot v_{A'}[j] \cdot u_B[j]. \end{aligned}$$

Therefore, there must be at least one $A' \in \mathcal{S}'$ such that

$$0 \neq \sum_{j=1}^{(r+1)^k} \sigma_j \cdot v_{A'}[j] \cdot u_B[j].$$

By our structural observation from Claim 3.2.8.2, this implies that $A' \sim_r^k B$ and hence, the set $\mathcal{S}'$ indeed k - r -represents $\mathcal{S}$.

As the last step, we prove the size bound on $\mathcal{S}'$. Since the rank of the matrix Q is at most $(r+1)^k$, the basis B_Q has size at most $(r+1)^k$. Hence, the set $\mathcal{S}'$ contains at most $(r+1)^k$ elements. $\blacktriangleleft$

The combination of Claims 3.2.8.1 and 3.2.8.3 concludes the proof of Lemma 3.2.8. $\square$

Our main idea is to use the representative sets to improve the running time of dynamic programs on tree decompositions. Besides a tight bound on the number of states that we have to store, we also need an efficient algorithm to handle the join nodes.

Unfortunately, so far the only way to do this is a naive combination of the states followed by a computation of the representative set.

It remains open to find a more efficient procedure to compute the representative set of the join of two representative sets.

3.2.2 Computing Representative Sets via a Single Matroid

In the previous section we gave a modified algorithm for products of matroids where we exploit the properties of uniform matroids. In this section, we directly use a faster algorithm for uniform matroids to give algorithms that do not depend on matrix multiplication. For the new approach we do not work with the product of k matroids of rank $r + 1$ but embed them in a single matroid of rank $(r + 1) \cdot k$.

We observe that this new matroid is still uniform and hence, we can use an existing result to compute the representative set. Formally, we use Fact 3.2.9 where the parameter x provides a trade-off between the size of the representative set and the time needed to compute it.

Fact 3.2.9 ([60, Theorem 1.2]). *Let $\mathcal{S}$ be a family of sets of size p over a universe of size n and let $0 < x < 1$. For a given q , a q -representative family $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ for $\mathcal{S}$ with at most $x^{-p} \cdot (1-x)^{-q} \cdot 2^{o(p+q)}$ sets can be computed in time $\mathcal{O}((1-x)^{-q} \cdot 2^{o(p+q)} \cdot |\mathcal{S}| \cdot \log n)$.*

With this intermediate results, we can now show a second method to compute representative sets. Instead of providing just one algorithm, we provide two algorithms, one general algorithm and one specifically tailored for tree decompositions. Here we use that the set which is given as input is of approximately (i.e., up to a constant) the same size as the resulting representative set. The first algorithm is ideal for the case of path decompositions. Surprisingly, in this case the size of the resulting representative set is not minimal. In order to compute such a minimal representative set, the running time of the algorithm increases by more than what we gain from the smaller size of the representative set. Thus, we avoid this additional computational overhead and use representative sets that are slightly larger instead.

The second algorithm is optimized for tree decompositions where we additionally take join nodes into account. Since the only currently known technique for this is the naive computation of the convolution, we want to decrease the size of the representative sets by taking into account the quadratic blow-up at the join nodes. When not considering the join operation, this method is slightly slower than the first algorithm. However, as this is clearly dominated by the computation at a join node, it eventually has no effect on the overall running time.

We formally state the two algorithm we just sketched by the following result.

Lemma 3.2.10. *Consider k integer universes $U_1, \dots, U_k$ of size n each. Given two sets $\mathcal{S}_1, \mathcal{S}_2 \subseteq U_1 \times \dots \times U_k$, we can compute*

1. *a set $\mathcal{S}' \subseteq_{r\text{-rep}}^k \mathcal{S}_1$ of size $(2r+1)^k \cdot (1+1/2r)^{r \cdot k} \cdot 2^{o(r \cdot k)}$ in time $|\mathcal{S}_1| \cdot (1+1/2r)^{r \cdot k} \cdot 2^{o(r \cdot k)} \cdot \log n$, and*
2. *a set $\mathcal{S}'' \subseteq_{r\text{-rep}}^k \mathcal{S}_1 + \mathcal{S}_2$ of size $(3r/2 + 1)^k \cdot (1 + 2/3r)^{r \cdot k} \cdot 2^{o(r \cdot k)}$ in time $|\mathcal{S}_1| \cdot |\mathcal{S}_2| \cdot (1 + 2/3r)^{r \cdot k} \cdot 2^{o(r \cdot k)} \cdot \log n$.*

Proof. We observe that in both cases, we have to compute a representative set of some set $\mathcal{S}$. In the first case we can use $\mathcal{S} = \mathcal{S}_1$ and in the second case we use $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$.

Hence, it suffices to give one algorithm where we later adjust the parameter x to obtain the claimed running times.

With this observation our algorithm works as follows.

1. We define the new universe U as the union of the given universes $U_1, \dots, U_k$ where we assume that all elements of each universe are given some identifier which allows us to determine the original universe this element came from. Formally, we define $\hat{U} = \hat{U}_1 \cup \dots \cup \hat{U}_k$ as the new universe where we set $\hat{U}_i := \{u^{(i)} \mid u \in U_i\}$.
2. Next, we use the relabeling of the universe, to represent the *vectors* in $\mathcal{S}$ by *sets*. Formally, for all $A \in \mathcal{S}$, we define $\hat{A} := \{u^{(i)} \mid A[i] = u \in U_i\}$ as the corresponding set and then, define $\hat{\mathcal{S}} := \{\hat{A} \mid A \in \mathcal{S}\}$.
3. We apply Fact 3.2.9 with universe U , parameter $p = k$, parameter $q = r \cdot k$, and where we choose the value for x later depending on the set $\mathcal{S}$. This provides us with a set $\hat{\mathcal{S}}' \subseteq_{r \cdot k\text{-rep}} \hat{\mathcal{S}}$ of size

$$|\hat{\mathcal{S}}'| \leq x^{-k} (1-x)^{-r \cdot k} \cdot 2^{o(r \cdot k + k)} \quad (3.2.5)$$

in time

$$|\hat{\mathcal{S}}| \cdot (1-x)^{-r \cdot k} \cdot 2^{o(r \cdot k + k)} \cdot \log(k \cdot n). \quad (3.2.6)$$

4. We recover the final set $\mathcal{S}' \subseteq U_1 \times \dots \times U_k$ as $\mathcal{S}' := \{A \mid \hat{A} \in \hat{\mathcal{S}}'\}$.

This concludes the algorithm except that we have to choose the appropriate value for x . We set the value of x for both cases separately. Our goal is to minimize the running time assuming that the given sets already satisfy the final size bound up to a constant factor.

Case 1. Since we are given only one set as input, we can simply plug in the size bound from Equation (3.2.5) into the bound of the running time from Equation (3.2.6) and then, minimize

$$(O(1) \cdot x^{-k} (1-x)^{-r \cdot k} \cdot 2^{o(r \cdot k + k)}) \cdot ((1-x)^{-r \cdot k} \cdot 2^{o(r \cdot k + k)} \cdot \log(k \cdot n)).$$

As the values for n and k are given as input (and positive), it suffices to minimize $x^{-1} \cdot (1-x)^{-r}$. By using the standard approach, it is straightforward to show that $x = 1/(2r+1)$ minimizes this value.

It remains to prove the final size bound and the bound on the running time. By Equation (3.2.6), the running time is bounded by

$$\begin{aligned} |\hat{\mathcal{S}}| \cdot (1-x)^{-r \cdot k} \cdot 2^{o(r \cdot k + k)} \cdot \log(k \cdot n) &\leq |\hat{\mathcal{S}}| \cdot \left(\frac{2r+1}{2r}\right)^{r \cdot k} \cdot 2^{o(r \cdot k)} \cdot \log n \\ &\leq |\hat{\mathcal{S}}| \cdot \left(\left(1 + \frac{1}{2r}\right)^r\right)^k \cdot 2^{o(rk)} \cdot \log n \end{aligned}$$

By Equation (3.2.5), the size of the output is bounded by

$$\begin{aligned} x^{-k} \cdot (1-x)^{-r \cdot k} \cdot 2^{o(r \cdot k + k)} &\leq (2r+1)^k \cdot \left(\frac{2r+1}{2r}\right)^{r \cdot k} \cdot 2^{o(r \cdot k)} \\ &\leq (2r+1)^k \cdot \left(1 + \frac{1}{2r}\right)^{r \cdot k} \cdot 2^{o(r \cdot k)} \end{aligned}$$

which concludes this case.

Case 2. For the second case we have to take into account that we first (naively) compute $S = S_1 + S_2$. Hence, the size of S is quadratic in the original bound. Therefore, our goal is to minimize

$$\begin{aligned} & \left(x^{-k} \cdot (1-x)^{-r \cdot k} \cdot 2^{o(r \cdot k + k)} \right)^2 \cdot \left((1-x)^{-r \cdot k} \cdot 2^{o(r \cdot k + k)} \cdot \log(k \cdot n) \right) \\ & \leq x^{-2k} \cdot (1-x)^{-3r \cdot k} \cdot 2^{3 \cdot o(r \cdot k)} \cdot \log n. \end{aligned}$$

Similar to the previous case, it suffices to minimize $x^{-2} \cdot (1-x)^{-3r}$. Again, when we use the standard approach, it is straightforward to check that $x = 2/(3r+2)$ minimizes this product.

For the bound on the running time, we consider Equation (3.2.6) which gives

$$\begin{aligned} |\widehat{S}| \cdot (1-x)^{-r \cdot k} \cdot 2^{o(r \cdot k + k)} \cdot \log(k \cdot n) & \leq |S_1| \cdot |S_2| \cdot \left(\frac{3r+2}{3r} \right)^{r \cdot k} \cdot 2^{o(r \cdot k)} \cdot \log n \\ & \leq |S_1| \cdot |S_2| \cdot (1 + 2/3r)^{r \cdot k} \cdot 2^{o(r \cdot k)} \cdot \log n. \end{aligned}$$

By Equation (3.2.5), the size of the new representative set is bounded by

$$\begin{aligned} x^{-k} \cdot (1-x)^{-rk} \cdot 2^{o(rk+k)} & \leq \left(\frac{3r+2}{2} \right)^k \cdot \left(\frac{3r+2}{3r} \right)^{rk} \cdot 2^{o(rk)} \\ & \leq (3r/2 + 1)^k \cdot (1 + 2/3r)^{rk} \cdot 2^{o(rk)} \end{aligned}$$

which concludes the second case.

Since the correctness follows directly from the correctness of Fact 3.2.9 and the fact that $S' \subseteq S$, this concludes the proof. $\square$

3.2.3 Compatibility Based on Forbidden and Allowed Values

Our previous results are stated for the robust but rather abstract notion of k - r -representation. In the following we introduce a more applied version of compatibility that we later use for the algorithmic results.

We first define a compatibility graph in Definition 3.2.11, which gives rise to the corresponding notions of compatibility and representation as defined in Definitions 3.2.1 and 3.2.2.

Definition 3.2.11 (Compatibility Graph). *Consider a list $\mathcal{F} = (F_1, \dots, F_k)$ of k sets $F_i \subseteq \mathbb{N}$, for all $i \in [1 \dots k]$, which are considered to be sets of forbidden numbers.*

We define the (infinite) compatibility graph $\mathcal{C}_{\mathcal{F}}$ of $\mathcal{F}$ with

- *vertex set $V(\mathcal{C}_{\mathcal{F}}) = L \dot{\cup} R$ where the two sets L and R are disjoint copies of $\mathbb{N}^k$, and*
- *edge set $E(\mathcal{C}_{\mathcal{F}}) = \{((a_1, \dots, a_k), (b_1, \dots, b_k)) \mid \forall i \in [1 \dots k] : a_i + b_i \notin F_i\}$.*

To motivate this compatibility notion, we focus on dynamic programming based algorithm for tree decompositions. We remark that this restriction is purely for illustrative reasons and the concepts can also be applied to more general settings.

An Intuitive Interpretation. Intuitively, one can think of $\mathcal{C}_{\mathcal{F}}$ -compatibility as follows. Imagine a problem where the objective is to find a structure in the graph such that every vertex obeys some degree constraint. These constraints are specified by the sets F_i for each vertex v_i . More precisely, for each vertex v_i , the degree in the final solution *must not* be contained in the set F_i , that is, the set F_i specifies the *forbidden* degrees of the corresponding vertex v_i .

Having set the stage, we assume that we are given a partial solution which is described by some state-vector for the vertices in a bag β . This state-vector, say a , consists of a sequence of integers which describe the degree of the vertices in the bag. In light of Definition 3.2.11, we assume that this vector a comes from the set $L = \mathbb{N}^k$.

A partial solution (and the corresponding state-vector) is “good” if there is some other partial solution for the remaining graph such that the union of both partial solutions forms one valid solution. More formally, this means that there is some state-vector $b \in R = \mathbb{N}^k$ (representing a partial solution for the remaining graph) such that $a + b$ are *compatible*, that is, for each vertex, the degrees from the two partial solutions sum to some value that is not forbidden (i.e., is allowed). This notion of compatibility is precisely characterized by Definition 3.2.11 as we do not have an edge between two (vectors of) values if and only if they sum to a forbidden value (for some dimension).

As a next step we provide a connection between $\mathcal{C}_{\mathcal{F}}$ -compatibility and the k - r -compatibility we introduced earlier in Definition 3.2.3.

A Link to k - r -Representation. Formally, by the definition of the compatibility graphs, the existence of an edge depends on the fact whether the sum $a + b$ is not contained in F_i for each vertex v_i . Naturally, for this check we have to know b and the sets F_i . Thus, this compatibility notion depends on the current node and hence, we need a (slightly) different procedure to compute the representative set for each node of the tree decomposition. We overcome this restriction by the following insight.

The main observation is that $a + b \notin F$ is equivalent to $a \notin F - b := \{f - b \mid f \in F\}$. Now, assume for a moment that we do not only consider sets of the form $F - b$ for all b , but consider *all* possible sets of size at most $r = |F|$ for the right-hand side. By this change, this new notion of compatibility is independent of b and F but only depends on $|F| = r$. Now, we can use the same compatibility for each node of the tree decomposition. Moreover, as we consider all sets of a certain size, this notion precisely corresponds to k - r -compatibility from Definition 3.2.3.

For the intuition, we only consider one vertex and thus, omit the index from F_i .

Definition 3.2.3 (k - r -Representation). Denote by k and r two positive integers.

We define the k - r -graph $\mathcal{G}_r^k = (L \dot{\cup} R, E)$ as the bipartite graph with left side $L := \mathbb{N}^k$, with right side $R := \binom{\mathbb{N}}{r}^k$, and with edges $E := \{(a, B) \in L \times R \mid \forall i \in [1..k] : a[i] \notin B[i]\}$.

By Definition 3.2.2, we denote

- by $\sim_r^k$ the $\mathcal{G}_r^k$ -compatibility, k - r -compatibility for short, and
- by $\subseteq_{r\text{-rep}}^k$ the $\mathcal{G}_r^k$ -representation, k - r -representation for short.

Based on our observation, we conclude that the notion of $\mathcal{C}_{\mathcal{F}}$ -compatibility is a more restrictive version of k - r -compatibility. We formalize this intuition in Lemma 3.2.12, and show that k - r -compatibility implies $\mathcal{C}_{\mathcal{F}}$ -compatibility.

Lemma 3.2.12. Consider a list $\mathcal{F} = (F_1, \dots, F_k)$ of k sets each containing at most r numbers (which are considered to be forbidden), that is, for all $i \in [1..k]$, we have $F_i \subseteq \mathbb{N}$ and $|F_i| \leq r$.

Then, for all $S, S' \subseteq \mathbb{N}^k$, the following holds: If $S' \subseteq_{r\text{-rep}}^k S$, then $S' \subseteq_{\mathcal{C}_{\mathcal{F}}\text{-rep}} S$.

Proof. To simplify notation we set $\mathcal{C} = \mathcal{C}_{\mathcal{F}}$ in the following.

Consider a vector $a \in S \subseteq \mathbb{N}^k$ such that there is a second vector $d \in \mathbb{N}^k$ with $a \sim_{\mathcal{C}} d$. By Definition 3.2.11, this implies that $a[i] + d[i] \notin F_i$ for all $i \in [1..k]$. Rearranging terms yields $a[i] \notin F_i - d[i]$ where we define $(F_i - d[i]) := \{f - d[i] \mid f \in F_i\}$. From Definition 3.2.3, we get that $a \sim_r^k (F_1 - d[1], \dots, F_k - d[k])$.

By the definition of representation and the assumption that $S' \subseteq_{r\text{-rep}}^k S$, there is some state-vector $a' \in S'$ such that $a' \sim_r^k (F_1 - d[1], \dots, F_k - d[k])$. We directly get that $a'[i] \notin F_i - d[i]$ or equivalently $a'[i] + d[i] \notin F_i$, for all $i \in [1..k]$. By the definition of $\mathcal{C}$ -representation, this is equivalent to $a' \sim_{\mathcal{C}} d$ which concludes the proof. $\square$

With Lemma 3.2.12, we can reuse the earlier results from Section 3.2 to compute the $\mathcal{C}_{\mathcal{F}}$ -representative sets. We state the result only for the algorithm based on matrix multiplication from Lemma 3.2.8, which we restate here for convenience.

Lemma 3.2.8. Consider k integer universes $U_1, \dots, U_k$ of dimension n each. Given a set $S \subseteq U_1 \times \dots \times U_k$, we can compute a set $S' \subseteq_{r\text{-rep}}^k S$ of size $(r+1)^k$ in time $\mathcal{O}(|S| \cdot (r+1)^{k \cdot (\omega-1)} \cdot k)$ where ω is the matrix multiplication exponent.

We combine Lemma 3.2.8 with Lemma 3.2.12 to obtain Lemma 3.2.13.

Lemma 3.2.13. Consider a list $\mathcal{F} = (F_1, \dots, F_k)$ of k sets each containing at most r numbers (which are considered to be forbidden), that is, for all $i \in [1..k]$, we have $F_i \subseteq \mathbb{N}$ and $|F_i| \leq r$.

Further, let $S \subseteq \mathbb{N}^k$ denote a finite set. Then, one can compute a $\mathcal{C}_{\mathcal{F}}$ -representative set S' of S such that $|S'| \leq (r+1)^k$ in time $\mathcal{O}(|S| \cdot (r+1)^{k \cdot (\omega-1)} \cdot k)$.

Proof. For a given set S , we use Lemma 3.2.8 to compute a k - r -representative set S' . For each of the universes we use $[0..M]$ where $M = \max\{\max_i F_i, \max_j P_j\}$.

The size bound on S' follows directly. By Lemma 3.2.12, we know that S' is also a $\mathcal{C}_{\mathcal{F}}$ -representative set of S which concludes the proof. $\square$

Including Allowed Degrees. All previous results consider the case where we specify values that are *not valid* for a solution. While the corresponding results are useful for those cases where only the forbidden values are specified, they are unpractical for the cases where many values are forbidden, that is, where only few values is allowed. Phrased differently, the case of the forbidden sets corresponds to a cofinite set of allowed values while the currently missing case captures finite sets of allowed values.

As a next step, we extend the previous results such that we also permit finite sets of allowed (that is, positive) values.

Definition 3.2.14. Consider a list $\mathcal{F} = (F_1, \dots, F_k)$ of k sets $F_i \subseteq \mathbb{N}$ for all $i \in [1..k]$, which are considered to be sets of forbidden numbers and a list $\mathcal{P} = (P_1, \dots, P_\ell)$ of ℓ sets $P_j \subseteq \mathbb{N}$ for all $j \in [1..\ell]$, which are considered to be sets of positive numbers. The compatibility graph for forbidden sets $\mathcal{F}$ and positive sets $\mathcal{P}$ is the infinite graph $\mathcal{C}_{\mathcal{F}, \mathcal{P}}$ with

- vertex set $V(\mathcal{C}_{\mathcal{F};\mathcal{P}}) := L^{k+\ell} \cup R^{k+\ell}$ where L and R are disjoint sets both identified with $\mathbb{N}$, and
- edge set $E(\mathcal{C}_{\mathcal{F};\mathcal{P}}) := \{((a_1, \dots, a_{k+\ell}), (b_1, \dots, b_{k+\ell})) \mid \forall i \in [1..k] : a_i + b_i \notin F_i \text{ and } \forall j \in [1..\ell] : a_{k+j} + b_{k+j} \in P_j\}$.

As before, this directly gives rise to the corresponding notions of $\mathcal{C}_{\mathcal{F};\mathcal{P}}$ -compatibility and $\mathcal{C}_{\mathcal{F};\mathcal{P}}$ -representation according to Definitions 3.2.1 and 3.2.2. To simplify notation, we just write $(\mathcal{F}; \mathcal{P})$ -compatibility and $(\mathcal{F}; \mathcal{P})$ -representation.

By a brute-force approach for the positions with positive sets, we generalize Lemma 3.2.13 to forbidden and positive sets. It is important to observe that in the following we bound the size of the sets F_i but the *maximum value* of the sets P_i . To see that this is the correct choice, we refer to the algorithmic results in Section 9.2 and, in particular, to the proof of Main Theorem III.2.

Lemma 3.2.15. *Consider a list $\mathcal{F} = (F_1, \dots, F_k)$ of k sets $F_i \subseteq \mathbb{N}$ with $|F_i| \leq r$ for all $i \in [1..k]$, and a list $\mathcal{P} = (P_1, \dots, P_\ell)$ of ℓ sets $P_j \subseteq \mathbb{N}$ with $\max P_j \leq r$ for all $j \in [1..\ell]$.*

Further, write $\mathcal{S} \subseteq \mathbb{N}^k$ for a finite set. Then, one can compute an $(\mathcal{F}; \mathcal{P})$ -representative set $\mathcal{S}'$ of $\mathcal{S}$ such that $|\mathcal{S}'| \leq (r+1)^{k+\ell}$ in time $\mathcal{O}(|\mathcal{S}| \cdot (r+1)^{\max\{\ell, k \cdot (\omega-1)\}} \cdot (k+\ell))$.

Proof. The algorithm starts with some preprocessing. We first compute the set

$$\hat{\mathcal{S}} := \{(a_1, \dots, a_{k+\ell}) \in \mathcal{S} \mid \forall j \in [1..\ell] : a_{k+j} \leq r\}.$$

Clearly, the set $\hat{\mathcal{S}}$ can be computed in time $\mathcal{O}(|\mathcal{S}| \cdot \ell)$. Also, every vector $(a_1, \dots, a_{k+\ell}) \in \mathcal{S} \setminus \hat{\mathcal{S}}$ is an isolated vertex in $\mathcal{C}_{\mathcal{F};\mathcal{P}}$ because $\max P_j \leq r$ for all $j \in [1..\ell]$. Hence, it suffices to compute an $(\mathcal{F}; \mathcal{P})$ -representative set of $\hat{\mathcal{S}}$ as the vectors from $\mathcal{S} \setminus \hat{\mathcal{S}}$ can be removed safely.

Therefore, for ease of notation, we assume that $\mathcal{S} = \hat{\mathcal{S}}$.

With this preprocessing, the algorithm proceeds as follows.

1. We say that two elements $(a_1, \dots, a_{k+\ell}), (b_1, \dots, b_{k+\ell}) \in \mathcal{S}$ are *positive-equivalent* if $a_{k+j} = b_{k+j}$ for all $j \in [1..\ell]$. We denote by $\mathcal{S}_1, \dots, \mathcal{S}_p$ the equivalence classes based on this relation for the vectors from $\mathcal{S}$.

Note that, by the preprocessing of $\mathcal{S}$, we have $p \leq (r+1)^\ell$. Moreover, we can compute the sets $\mathcal{S}_1, \dots, \mathcal{S}_p$ in time $\mathcal{O}(|\mathcal{S}| \cdot (r+1)^\ell \cdot (k+\ell))$ by iterating over $\mathcal{S}$ and assigning each vector to the correct equivalence class.

2. For each $q \in [1..p]$, we use Lemma 3.2.13 to compute a $(F_1, \dots, F_k)$ -representative set $\mathcal{S}'_q$ of $\mathcal{S}_q$ where we simply ignore the positions from $[k+1..k+\ell]$.

This can be done in time $\mathcal{O}(|\mathcal{S}_q| \cdot (r+1)^{k \cdot (\omega-1)} \cdot k)$. Additionally, we directly get that $|\mathcal{S}'_q| \leq (r+1)^k$. Moreover, as all vectors share the same values on positions from $[k+1..k+\ell]$, the set $\mathcal{S}'_q$ is also an $(\mathcal{F}; \mathcal{P})$ -representative set of $\mathcal{S}_q$.

3. For the last step of the computation we define

$$\mathcal{S}' := \bigcup_{q \in [1..p]} \mathcal{S}'_q$$

and return S' as the result of the algorithm.

Observe that $|S'| \leq p \cdot (r+1)^k \leq (q+1)^{k+\ell}$.

By the running time of the individual steps, the entire procedure to compute S' takes time

$$\begin{aligned} & \mathcal{O}\left(|S| \cdot \ell + |S| \cdot (r+1)^\ell \cdot (k+\ell) + \sum_{q \in [1..p]} |S_q| \cdot (r+1)^{k \cdot (\omega-1)} \cdot k\right) \\ &= \mathcal{O}\left(|S| \cdot (r+1)^{\max\{\ell, k \cdot (\omega-1)\}} \cdot (k+\ell)\right) \end{aligned}$$

overall, which concludes the proof. $\square$

Operations that Preserve Representation. So far we have seen how to compute an $(\mathcal{F}; \mathcal{P})$ -representative set. In the remainder we show that certain operations (which typically appear in algorithms operating on tree decompositions) preserve the representation. We refer to [13, Definition 3.2] for a similar result regarding a different notation of representation and compatibility.

Lemma 3.2.16. *Let $F_1, \dots, F_k, P_1, \dots, P_\ell \subseteq \mathbb{N}$ denote $k + \ell$ finite sets and write $\mathcal{F} = (F_1, \dots, F_k)$ and $\mathcal{P} = (P_1, \dots, P_\ell)$. Further, write $S \subseteq \mathbb{N}^{k+\ell}$ for a finite set and let S' denote an $(\mathcal{F}; \mathcal{P})$ -representative set of S .*

- (a) *For every vector $d \in \mathbb{Z}^{k+\ell}$, the set $S' + d := \{s + d \mid s \in S'\}$ is an $(\mathcal{F}; \mathcal{P})$ -representative set of $S + d$.*
- (b) *For every $i \in [1..k]$, the set $S' \boxplus i := \{s[1..i-1] \, s[i+1..k+\ell] \mid s \in S', s[i] \notin F_i\}$ is an $(\mathcal{F}'; \mathcal{P})$ -representative set of $S \boxplus i$ where $\mathcal{F}' := (F_1, \dots, F_{i-1}, F_{i+1}, \dots, F_k)$.*
- (c) *For every $j \in [1..\ell]$, the set $S' \boxplus (k+j) := \{s[1..k+j-1] \, s[k+j+1..k+\ell] \mid s \in S', s[k+j] \in P_j\}$ is an $(\mathcal{F}; \mathcal{P}')$ -representative set of $S \boxplus (k+j)$ where $\mathcal{P}' := (P_1, \dots, P_{j-1}, P_{j+1}, \dots, P_\ell)$.*

For every finite set $Q \subseteq \mathbb{N}$, and

- (d) *for every $i \in [1..k+1]$, the set $S' \boxplus i := \{s[1..i-1] \, 0 \, s[i..k+\ell] \mid s \in S'\}$ is an $(\mathcal{F}'; \mathcal{P})$ -representative set of $S \boxplus i$ where $\mathcal{F}' := (F_1, \dots, F_{i-1}, Q, F_i, \dots, F_k)$, and*
- (e) *for every $j \in [1..\ell+1]$, the set $S' \boxplus (k+j) := \{s[1..k+j-1] \, 0 \, s[k+j..k+\ell] \mid s \in S'\}$ is an $(\mathcal{F}; \mathcal{P}')$ -representative set of $S \boxplus (k+j)$ where $\mathcal{P}' := (P_1, \dots, P_{j-1}, Q, P_j, \dots, P_\ell)$.*

For every finite set $\mathcal{R} \subseteq \mathbb{N}^{k+\ell}$ and every $(\mathcal{F}; \mathcal{P})$ -representative sets $\mathcal{R}'$ of $\mathcal{R}$,

- (f) *the set $S' \cup \mathcal{R}'$ is an $(\mathcal{F}; \mathcal{P})$ -representative set of $S \cup \mathcal{R}$, and*
- (g) *the set $S' + \mathcal{R}' := \{s + r \mid s \in S', r \in \mathcal{R}'\}$ is an $(\mathcal{F}; \mathcal{P})$ -representative set of $S + \mathcal{R}$.*

Proof. For two vectors $a, b \in \mathbb{N}^{k+\ell}$, we write $a \sim_{\mathcal{F}; \mathcal{P}} b$ if $(a, b) \in E(\mathcal{C}_{\mathcal{F}; \mathcal{P}})$, that is, if $a[i] + b[i] \notin F_i$ for all $i \in [1..k]$ and $a[k+j] + b[k+j] \in P_j$ for all $j \in [1..\ell]$.

We first observe that we only have to prove the forward direction of the equivalence in Definition 3.2.2 since $S' \subseteq S$. We prove the different cases independently.

- (a) Fix some vector $d \in \mathbb{Z}^{k+\ell}$ and consider two vectors $a \in \mathcal{S} + d$ and $b \in \mathbb{N}^{k+\ell}$ such that $a \sim_{\mathcal{F};\mathcal{P}} b$.

By the definition of the set $\mathcal{S} + d$, there is some vector $\hat{a} \in \mathcal{S}$ such that $a = \hat{a} + d$. Therefore, $\hat{a} + d \sim_{\mathcal{F};\mathcal{P}} b$ is equivalent to $\hat{a} \sim_{\mathcal{F};\mathcal{P}} b + d$. As we know by assumption that $\mathcal{S}'$ is an $(\mathcal{F}; \mathcal{P})$ -representative set of $\mathcal{S}$, there is some vector $\hat{a}' \in \mathcal{S}'$ such that $\hat{a}' \sim_{\mathcal{F};\mathcal{P}} b + d$. It follows that $\hat{a}' + d \sim_{\mathcal{F};\mathcal{P}} b$ and that $\hat{a}' + d \in \mathcal{S}' + d$, which concludes this case.

- (b) Without loss of generality, we assume $i = 1$ to simplify notation. Consider two vectors $a \in \mathcal{S} \boxplus 1$ and $b \in \mathbb{N}^{k-1+\ell}$ such that $a \sim_{\mathcal{F}';\mathcal{P}} b$ where $\mathcal{F}' = (F_2, \dots, F_k)$.

By the definition of the set $\mathcal{S} \boxplus i$, there is some vector $\hat{a} \in \mathcal{S}$ such that $\hat{a}[2..k+\ell] = a$ and $\hat{a}[1] \notin F_1$. In particular, we know that $\hat{a} \sim_{\mathcal{F};\mathcal{P}} 0 \cdot b$. Since $\mathcal{S}'$ is an $(\mathcal{F}; \mathcal{P})$ -representative set of $\mathcal{S}$, there is some vector $\hat{a}' \in \mathcal{S}'$ such that $\hat{a}' \sim_{\mathcal{F};\mathcal{P}} 0 \cdot b$. This implies that $\hat{a}'[1] \notin F_1$ and thus, $\hat{a}'[2..k+\ell] \sim_{\mathcal{F}';\mathcal{P}} b$ and $\hat{a}'[2..k+\ell] \in \mathcal{S}' \boxplus 1$.

- (c) This case is analogous to the previous case except that we use that $\hat{a}[k+j] \in P_j$.
- (d) Fix a finite set $Q \subseteq \mathbb{N}$ and denote by $i \in [1..k]$ some position. To simplify notation, we assume without loss of generality that $i = 1$. Moreover, consider two vectors $a \in \mathcal{S} \boxplus i$ and $b \in \mathbb{N}^{k+\ell+1}$ such that $a \sim_{Q,\mathcal{F};\mathcal{P}} b$.
- By the definition of the set $\mathcal{S} \boxplus 1$, we get $a[1] = 0$ and hence, $b[1] \notin Q$. Moreover, we know that $a[2..k+\ell+1] \sim_{\mathcal{F};\mathcal{P}} b[2..k+\ell+1]$. Since $\mathcal{S}'$ is an $(\mathcal{F}; \mathcal{P})$ -representative set of $\mathcal{S}$, there is some vector $a' \in \mathcal{S}'$ such that $a' \sim_{\mathcal{F};\mathcal{P}} b[2..k+\ell+1]$. Then $0 \cdot a' \in \mathcal{S}' \boxplus 1$ and $0 \cdot a' \sim_{Q,\mathcal{F};\mathcal{P}} b$ because $b[1] \notin Q$.
- (e) This case is analogous to the previous case except that we use that $a[k+j] = 0$ and $b[k+j] \in Q$.
- (f) This case holds trivially since the vectors are not changed.
- (g) Consider two vectors $a \in \mathcal{S} + \mathcal{R}$ and $b \in \mathbb{N}^{k+\ell}$ such that $a \sim_{\mathcal{F};\mathcal{P}} b$.

By the definition of the set $\mathcal{S} + \mathcal{R}$, there are two vectors $a_1 \in \mathcal{S}$ and $a_2 \in \mathcal{R}$ such that $a_1 + a_2 = a$. Hence, we especially have $a_1 \sim_{\mathcal{F};\mathcal{P}} b + a_2$. Since $\mathcal{S}'$ is an $(\mathcal{F}; \mathcal{P})$ -representative set of $\mathcal{S}$, there is some $a'_1 \in \mathcal{S}'$ such that $a'_1 \sim_{\mathcal{F};\mathcal{P}} b + a_2$ which is equivalent to $a_2 \sim_{\mathcal{F};\mathcal{P}} b + a'_1$.

Now, since $\mathcal{R}'$ is an $(\mathcal{F}; \mathcal{P})$ -representative set of $\mathcal{R}$, we obtain a vector $a'_2 \in \mathcal{R}'$ such that $a'_2 \sim_{\mathcal{F};\mathcal{P}} b + a'_1$. As this is equivalent to $a'_1 + a'_2 \sim_{\mathcal{F};\mathcal{P}} b$, and we know that $a'_1 + a'_2 \in \mathcal{S}' \oplus \mathcal{R}'$, this concludes the proof. $\square$

With Lemma 3.2.16, it is straightforward to design dynamic programs that operate on a given tree decomposition and store representative sets for each node. Depending on the type of the current node, we apply the computations from one of the cases of Lemma 3.2.16 and use Lemma 3.2.15 to compute a small representative set of the output. More precisely, we use the following correspondence between the types of nodes and the operations.

Leaf Node: No procedure is needed for this case as we construct the solutions explicitly.

Introduce Vertex Node: Depending on the vertex introduced, we use the procedure from Case (d) or (e) as the new vertex is not adjacent to any other vertices.

Introduce Edge Node: We use the procedure from Case (a) where the vector depends on the endpoints of the edges.

Forget Node: Depending on the position of the forgotten vertex, we use the procedure from Case (b) or (c).

Join Node: We use the procedure from Case (g) to combine the two representative sets associated with the child nodes.

We note that the procedure from Case (f) is not associated to a specific node type but is used for multiple types of nodes.

A Toolkit for Proving Lower Bounds

4

In Section 4.1 we extend SETH to constraint satisfaction problems to establish MONO-pw-CSPH, that is, we prove Main Theorem I.3.

In Section 4.2 we consider half-induced matchings and prove some useful properties that we use later.

4.1 From Satisfiability to Constraint Satisfaction

The lower bounds we prove in this work are based on a variant of SETH or its counting version. However, instead of repeating the original statement we use the following formulation which is more useful to work with.

Conjecture 4.1.1 (STRONG EXPONENTIAL TIME HYPOTHESIS (SETH) [23, 80]). *For all $\delta > 0$, there is a $k \geq 3$ such that satisfiability of k -CNF formulas on n variables requires more than $(2 - \delta)^n$ time.*

Similarly, we can define the counting version of the SETH based on the counting version of SAT, that is, the problem where we want to *count* all satisfying assignments of a given formula.

Conjecture 4.1.2 (#STRONG EXPONENTIAL TIME HYPOTHESIS (#SETH)). *For all $\delta > 0$, there is a $k \geq 3$ such that counting the number of satisfying assignments of a k -CNF formulas on n variables requires more than $(2 - \delta)^n$ time.*

During the long line of research for proving lower bounds under SETH when parameterizing by treewidth (or rather pathwidth) [16, 35, 55, 97] many reductions use similar conceptual ideas. Unfortunately, these ideas also come with shared weaknesses that complicate the proofs without providing any significant insight into the problem at hand. Additionally, these difficulties also prohibit that the reductions work out-of-the-box for different variants of the problems such as the counting version for example.

By a close observation, we can identify the following three difficulties that are independent of the considered problem.

1. A plain reduction from SAT results in a lower bound of the form $(2 - \epsilon)^{pw}$. However, in most cases, the (matching) lower bound is of the form $(c - \epsilon)^{pw}$ for some constant c depending on the problem. Thus, the reductions need some technical overhead that makes the transition between these two different bases.
2. The assumption that SETH holds is comparably strong. Indeed, the reductions frequently construct a grid-like structure with the rows representing the variables and the columns corresponding to the clauses. In this case, the bags correspond roughly to the columns of

the resulting structure. As the reductions usually start with a k -CNF, only a small portion of the variables is used at the same time but all variables (or rather their representation) appears in each bag of the constructed path decomposition.

3. By the design of the reductions there is usually the possibility that the encoded assignment is not consistent at all places but might change at some point. Luckily, these changes do not happen in a completely uncontrolled way. Instead, their number is usually bounded by a constant and thus, repeating the construction sufficiently often ensures that in some repetition of the construction, the assignments are encoded consistently. It is not hard to imagine that these variations of the assignment are not beneficial when considering the counting version.

In the following we analyze these obstacles in more details and explain how they can be avoided. The first problem was already identified and addressed by Lampis in [94]. In a recent work, Lampis considered the other two issues as well and stated a way to resolve them [96]. We revisit these results but also extend them to the counting version.

After our intuitive description, we formally prove these results in Sections 4.1.1 to 4.1.3. To stay in line with the later reductions, we start with the second problem, then consider the first, and finally, consider the third problem.

Pathwidth SETH. One potential idea to by-pass the limitations of SETH could be to identify highly connected groups of variables, that is, identify groups of variables that appear in multiple clauses together. Then, the remaining variables are only loosely connected to these groups. This could potentially give rise to a dynamic program that starts at the highly-connected groups and then extends these assignments to the other variables. Such an algorithm might solve SAT significantly faster than brute force especially when the variables can be partitioned into groups that do not interact with each other.

To formalize this idea, Lampis introduces the *primal pathwidth* version of the SETH which we denote by pw-SETH [96]. The intuitive idea is to consider the primal graph of the formula, that is, the graph with one vertex for each variable and an edge between two vertices if the corresponding variables appear together in one clause. Then, the primal pathwidth is the pathwidth of the primal graph. Intuitively, the lower the primal pathwidth, the more structured the formula is and hence, the faster we can check if the formula can be satisfied. More formally, the resulting conjecture pw-SETH claims that SAT cannot be solved in time $(2 - \varepsilon)^{pw}$ when a path decomposition of width pw of the primal graph is given along with the formula.

This conjecture pw-SETH is certainly more plausible than the plain SETH. Lampis also proves that this new conjecture is *equivalent* to the known lower bounds for COLORING, DOMINATING SET, INDEPENDENT SET, for example, that were previously shown under SETH. Hence, disproving pw-SETH becomes more unlikely as otherwise many other problems would have faster algorithms parameterized by pathwidth, although the current algorithms are assumed to be optimal.

We dedicate Section 4.1.1 to the pw-SETH and introduce the concepts and hypotheses formally.

Fixing the Base. Although the pw-SETH provides a better starting point for the reductions than SETH, the change of bases is still necessary. Phrased differently, a naive encoding of the variables provides a lower bound of the form $(2 - \varepsilon)^{\text{pw}}$, but the fastest algorithm has running time c^{pw} . To close this gap, the reductions frequently form groups of d variables and then encode the 2^d assignment by the states of g vertices, where each vertex can have c states. When choosing the parameters such that $2^d = c^g$, all assignments can be encoded and a faster algorithm for the problem at hand would contradict SETH.

Because of rounding issues and other obstacles, the choice of the parameters is not as straightforward and results in a tedious overhead that does not contribute any insight into the problem.

Lampis identified this common issue of the reductions and introduced the q -CSP- B problem [94]. Instead of two values as for SAT, each variable can now take up to B different values. Each clause or rather constraint (as we refer to them in the context of a CSP) now involves at most q variables. Lampis showed that assuming pw-SETH, the problem cannot be solved in time $(B - \varepsilon)^{\text{pw}}$ for a CSP-instance of primal pathwidth pw. We later refer to this hypothesis as pw-CSPH.

With this problem we can establish a straightforward correspondence between the assignments to the variables and the states of the vertices. Hence, the reduction can be simplified significantly which simultaneously emphasizes the ideas used to realize the different states of the vertices. The lower bound then directly follows from pw-CSPH.

We introduce q -CSP- B formally in Section 4.1.2. There we also prove that pw-SETH implies pw-CSPH and similarly for the corresponding counting versions.

Monotone Assignments. When we encode the assignments of the variables by the states of the vertices, we would like to ensure that the state of the vertices does not change. By the structure of the underlying problems, this might not necessarily be the case. Instead, the vertices might change their state between two clauses. Luckily these changes can usually happen in one direction only, that is, the state changes monotonically. Adding additional constraints or gadgets that ensure that the vertices do not change their state is not possible in most cases as this would increase the pathwidth significantly. Instead, a common solution is to repeat the construction several times such that in at least one repetition no change happens. This trick works as the number of changes is bounded and thus, one repetition can be guaranteed to have no change.

Lampis considered this problem and defined a somewhat artificial version of the q -CSP- B problem [96]. In this version, the variables might receive a different assignment for each bag, a so-called *multi-assignment*. As this would make the problem trivial if we do not impose further restrictions, we restrict ourselves to those assignments where the states are monotone. Additionally, we also add few variables where the assignment must not change. These variables are then used to ensure that eventually all variables have a consistent assignment which does not change.

We formally introduce this variant in Definition 4.1.13 and prove that this restriction does not make the problem easier, that is, based on arguments by Lampis [96], we show that pw-CSPH also holds in this special setting.

4.1.1 Primal Pathwidth

Instead of using (the counting version of) SETH as the condition for our lower bounds, we start from the variant pw-SETH that is based on the pathwidth of a formula. As mentioned earlier, Lampis introduced this variant and showed that the lower bounds for COLORING, INDEPENDENT SET, and DOMINATING SET parameterized by pathwidth are equivalent to pw-SETH [96].

To formally state this conjecture, we first define the primal graph of a SAT-formula or CSP-instance.

Definition 4.1.3 (Primal Graph and Primal Pathwidth). *The primal graph G_P of a SAT-formula (resp. CSP-instance) ϕ contains one vertex for each variable of ϕ and an edge xy if and only if the variables x and y appear together in one clause (resp. constraint) of ϕ .*

The primal pathwidth of ϕ , denoted by $\text{pw}(\phi)$, is the pathwidth of the primal graph G_P of ϕ .

Unless explicitly mentioned, we refer to the primal pathwidth of a formula ϕ just as the pathwidth of ϕ . To simplify notation, we sometime also refer to the path decomposition of the primal graph of a formula ϕ as the path decomposition of ϕ .

Now we can state the primal pathwidth variant of SETH as defined by Lampis and the corresponding counting version.

Conjecture 4.1.4 (PRIMAL PATHWIDTH SETH (pw-SETH) [96, Conjecture 1.1]). *For all $\varepsilon > 0$ we have the following: there exists no algorithm which takes as input a 3-SAT instance ϕ and a path decomposition of its primal graph of width pw and correctly decides if ϕ is satisfiable in time $(2 - \varepsilon)^{\text{pw}} \cdot |\phi|^{\mathcal{O}(1)}$.*

Similarly, we can define the counting version of the pw-SETH based on the counting version of SAT.

Conjecture 4.1.5 (PRIMAL PATHWIDTH COUNTING SETH (pw-#SETH)). *For all $\varepsilon > 0$ we have the following: there exists no algorithm which takes as input a 3-SAT instance ϕ and a path decomposition of its primal graph of width pw and correctly counts the number of satisfiable assignments for ϕ in time $(2 - \varepsilon)^{\text{pw}} \cdot |\phi|^{\mathcal{O}(1)}$.*

In the next section, we extend pw-SETH and pw-#SETH to the case when the variables can take more than two values.

4.1.2 Larger Alphabets

To simplify the reductions for the lower bounds, we extend pw-SETH (and pw-#SETH) to CONSTRAINT SATISFACTION PROBLEMS (CSPs) where every variable can now potentially take more than two values. These modifications allow us to reduce the technical overhead appearing in the constructions for the lower bounds.

We first formally define the problem we use.

Definition 4.1.6 (q -CSP- B [94]). *Fixed two positive numbers $q \geq 2$ and $B \geq 2$.*

A q -CSP- B instance consists of a set X of n variables which take values in $[1..B]$ and a set C of q -constraints on X . A q -constraint C is defined by an ordered tuple $\text{scp}(C)$ of q variables of X , and a set $R(C) \subseteq [1..B]^q$ of satisfying assignments for C .

The task of q -CSP- B is to decide if such a given q -CSP- B instance is satisfiable, that is, decide if there is an assignment $\pi: X \rightarrow [1..B]$ which satisfies all constraints $C \in \mathcal{C}$ where a constraint $C \in \mathcal{C}$ is satisfied if π restricted to $\text{scp}(C)$ appears in $R(C)$.

The parity of a constraint is the parity of the 1-norm of the accepted assignments restricted to the scope. The parity can be even, odd or undefined if not all accepted assignment restricted to the scope have the same 1-norm.

Analogously to the widely known SETH for the satisfiability problem, Lampis introduces the corresponding hypothesis for q -CSP- B [94].

Conjecture 4.1.7 (CSPH). *For all $B \geq 2$ and $\varepsilon > 0$, there exists a q such that q -CSP- B with n variables cannot be solved in time $(B - \varepsilon)^n \cdot n^{O(1)}$.*

Instead of using this hypothesis, we state a version introduced by Lampis based on the primal pathwidth of the given instance of q -CSP- B .

Conjecture 4.1.8 (pw-CSPH). *For all $B \geq 2$ and $\varepsilon > 0$, there exists a q such that no algorithm can decide the satisfiability of a q -CSP- B instance ϕ that is given with a path decomposition of its primal graph of width pw in time $(B - \varepsilon)^{\text{pw}} \cdot |\phi|^{O(1)}$.*

The hypothesis CSPH from Conjecture 4.1.7 clearly generalizes (and thus, implies) SETH as the latter follows for $B = 2$. However, Lampis showed that the converse is also true, that is, SETH implies CSPH [94, Theorem 3.1]. In a later work, Lampis extends this equivalence to the pathwidth-variants of the problems [96, Theorem 3.2].

Unfortunately, both results cover only the decision version of the problem. Hence, to extend these results to the counting version, we repeat the arguments in a more robust way such that we obtain the corresponding results for the counting version as well.

Before we give the formal proof, we first define the counting version of pw-CSPH, which is based on the counting version of q -CSP- B , where we *count* the number of satisfying assignments.

Conjecture 4.1.9 (pw-#CSPH). *For all $B \geq 2$ and $\varepsilon > 0$, there exists a q such that no algorithm can count the number of satisfying assignment for a q -CSP- B instance ϕ that is given with a path decomposition of its primal graph of width pw in time $(B - \varepsilon)^{\text{pw}} \cdot |\phi|^{O(1)}$.*

Given this definition, we prove that pw-SETH implies pw-CSPH and that pw-#SETH implies pw-#CSPH. We refer to Lemma 4.1.10 for the formal statement.

Intuition. The basic idea of the proof is as follows. This idea agrees with the technical adjustment of several SETH-based reductions that are needed to change the base of the lower bound from 2^{pw} to c^{pw} .

Assume we are given a SAT instance ϕ as a CNF. We partition the n variables of ϕ into n/q groups of size q each. For each such group, we encode the 2^q assignments by the assignments to g new variables that can take values from $[1..B]$. When we set $q := g \cdot \log B$, we immediately get that $2^q = B^g$ which allows us to define this encoding. Moreover, with this encoding we reduce the number of variables from n to

$$\frac{n}{q} \cdot g = \frac{n}{g \cdot \log B} \cdot g = \frac{n}{\log B}.$$

Additionally, we define new constraints which are based on the clauses of ϕ to take care of this encoding. We denote the resulting q -CSP- B instance by ψ . Then, the results from Lemma 4.1.10 follow immediately.

With this intuition, we move to the formal proof. However, instead of partitioning all vertices at once, we partition the bags of the given path decomposition. Otherwise, the size of the path decomposition might increase too much. Then, we additionally need to ensure that the encoded assignment does not change between two bags which we do by adding additional constraints. Moreover, due to rounding issues and other technicalities, the choice of g and q needs to be more subtle.

Lemma 4.1.10. *If pw-SETH holds, then pw-CSPH holds.*

Moreover, if pw-#SETH holds, then pw-#CSPH holds.

Both results hold even if all constraints have a fixed parity.

Proof. We provide a parsimonious, many-one reduction from k -SAT to q -CSP- B that does not significantly change pathwidth of the primal graphs to prove both results simultaneously. We use the proof of Lemma 3.7 in [96] as basis but extend and modify it such that the result also follows for the counting version.

We first fix some values $B \geq 2$ and $\varepsilon > 0$. Now assume that, for some value q , we can solve q -CSP- B (respectively the counting version thereof) in time $(B - \varepsilon)^{\text{pw}(\psi)} \cdot |\psi|^{\mathcal{O}(1)}$ for a formula ψ that is given with a path decomposition of the primal graph of width $\text{pw}(\psi)$.

For a value δ to be chosen later, we show that, for all $k \geq 3$, we can solve k -SAT (respectively the counting version thereof) in time $(2 - \delta)^{\text{pw}(\phi)} \cdot |\phi|^{\mathcal{O}(1)}$ for a k -CNF formula ϕ that is given with a path decomposition of the primal graph of width $\text{pw}(\phi)$.

Constructing the q -CSP- B Instance. Let ϕ be the given formula in k -CNF with n variables and m clauses together with a path decomposition with bags $\beta(1), \beta(2), \dots, \beta(b)$ of width pw . We denote by $V(\phi)$ the set of variables and by $C_1, \dots, C_m$ the clauses of ϕ . We can assume that all variables appear in some clause as we can remove them otherwise. Note that for the counting version we have to adjust for this in the final number of solutions.

For an integer value q to be chosen later, we partition the variables of ϕ into $t := \lceil (\text{pw}+1)/q \rceil$ groups of size at most q such that the intersection of each group with each bag is of similar size. Formally, we prove the following claim.

Claim 4.1.10.1. *There is a partition $V_1, \dots, V_t \subseteq V(\phi)$ such that, for all $i \in [1..t]$ and all $\ell \in [1..b]$, it holds that $|V_i \cap \beta(\ell)| \leq q$.*

This partition can be found in time polynomial in the size of the decomposition.

Proof of Claim. We construct the partition greedily where we iterate over all bags of the path decomposition, which we assume to be nice, in sequential order.

For the first bag, which contains only a single vertex v , we add v to the group V_1 . Clearly, the size bound follows.

Now assume that we partitioned all vertices appearing in the bags $\beta(1), \dots, \beta(\ell - 1)$. If bag $\beta(\ell)$ introduces some vertex v , that is, if $\beta(\ell) = \beta(\ell - 1) \cup \{v\}$, then we add v to the group V_i with the *smallest* intersection with $\beta(\ell)$ where we break ties arbitrarily. Assume without loss

of generality that we add v to group V_1 . Then, we get that $|V_1 \cap \beta(\ell)| = |V_1 \cap \beta(\ell - 1)| + 1$. If $|V_1 \cap \beta(\ell - 1)| = q$, then we know that $|\beta(\ell - 1)| \geq q \cdot t \geq pw + 1$ which leads to the contradiction that

$$pw + 1 \geq |\beta(\ell)| = |\beta(\ell - 1)| + 1 \geq pw + 2.$$

Hence, we know that $|V_1 \cap \beta(\ell - 1)| \leq q - 1$, and we can safely add v to V_1 .

If a vertex is forgotten at the bag $\beta(\ell)$, then the claim follows directly as the size bound depends only on q which does not change for the different bags. $\triangleleft$

For all $i \in [1..t]$ and all $\ell \in [1..b]$, we say that a partial assignment $\pi_{i,\ell}: V_i \cap \beta(\ell) \rightarrow \{0, 1\}$ satisfies a clause C_j if at least one literal in the clause C_j is satisfied under the assignment $\pi_{i,\ell}$. Note, that the set $V_i \cap \beta(\ell)$ might not cover all variables of the clause C_j but can still satisfy the clause. To simplify notation, we identify the assignment $\pi_{i,\ell}$ with a vector in $\{0, 1\}^{V_i \cap \beta(\ell)}$ in the natural way whenever needed.

For all sets $V_i \cap \beta(\ell)$, we define an encoding $\sigma_{i,\ell}$ of the at most 2^q partial assignments for the variables in $V_i \cap \beta(\ell)$ by vectors from $[1..B]^g$, where g is chosen later. Instead of using all B^g possible encodings, we only use those vectors where the parity of the 1-norm is equal.

Formally, for all $i \in [1..t]$ and all $\ell \in [1..b]$, we pick a value $a_{i,\ell} \in \{0, 1\}$ and choose an injective function $\sigma_{i,\ell}: \{0, 1\}^{V_i \cap \beta(\ell)} \rightarrow [1..B]^g$ such that, for all partial assignments $\pi_{i,\ell}: V_i \cap \beta(\ell) \rightarrow \{0, 1\}$, it holds that

$$a_{i,\ell} = \sum_{\gamma=1}^g \sigma_1(\pi_{i,\ell})[\gamma] \pmod{2}.$$

Note that we allow different encodings and parities for each pair (i, ℓ) .

To achieve that the functions $\sigma_{i,\ell}$ satisfy the two required properties, the values g and q must satisfy $2^q \leq B^{g-1} \cdot \lfloor B/2 \rfloor$. Indeed, with this assumption we can choose the first $g - 1$ positions of the values of $\sigma_{i,\ell}$ without any restrictions and then, can choose half of the B possible values for the last position to satisfy the parity constraint.

Based on these observations, we define

$$q := \lfloor g \cdot \log B \rfloor - 2 = \lfloor \log(B^g/4) \rfloor. \quad (4.1.1)$$

As we know that $B \geq 2$, we get that $\lfloor B/2 \rfloor \geq (B-1)/2 \geq B/4$. Thus, we have that $B^{g-1} \cdot \lfloor B/2 \rfloor \geq B^g/4$ and hence, the functions $\sigma_{i,\ell}$ can be defined as claimed.

Before we define the new q -CSP- B instance, we first introduce some notation. For all $j \in [1..m]$, we denote by $V(C_j) \subseteq [1..t]$ the indices of the groups V_i that contain vertices from C_j . Formally, assume that the vertices of the clause C_j are from the groups $V_{i_1}, \dots, V_{i_\kappa}$ for some indices $i_1, \dots, i_\kappa \in [1..t]$ where $\kappa \leq k$. Then, we define $V(C_j) := \{i_1, \dots, i_\kappa\}$.

Now we have everything ready to define the new CSP-instance ψ as follows.

1. For all $i \in [1..t]$, all $\ell \in [1..b]$, and all $\gamma \in [1..g]$, we create a variable $x_{i,\ell,\gamma}$.

As we do not introduce further variables, we denote by $V(\psi)$ the set of all variables of ψ .

2. For all $i \in [1..t]$ and all $\ell \in [1..b]$, we create a constraint $A_{i,\ell}$ that checks that the values of the variables $x_{i,\ell,\star}$ correspond to an *assignment* for the variables in $V_i \cap \beta(\ell)$.

Formally, the constraint $A_{i,\ell}$ has scope $\text{scp}(A_{i,\ell}) := \{x_{i,\ell,\gamma} \mid \gamma \in [1..g]\}$ and accepts if and only if there is a partial assignment $\pi_{i,\ell} \in \{0,1\}^{V_i \cap \beta(\ell)}$ such that $(x_{i,\ell,1}, \dots, x_{i,\ell,g}) = \sigma_{i,\ell}(\pi_{i,\ell})$.

3. For all $i \in [1..t]$ and all $\ell \in [1..b-1]$, we create a constraint $C_{i,\ell}$ that checks that the encoded assignment for $V_i \cap \beta(\ell)$ agree with the encoded assignment for $V_i \cap \beta(\ell+1)$ for the shared vertices, that is, the encoded assignment is *consistent*.

Formally, the constraint $C_{i,\ell}$ has scope $\text{scp}(C_{i,\ell}) := \{x_{i,\ell,\gamma}, x_{i,\ell+1,\gamma} \mid \gamma \in [1..g]\}$ and accepts if and only if we have $\pi_{i,\ell}[\beta(\ell) \cap \beta(\ell+1)] = \pi_{i,\ell+1}[\beta(\ell) \cap \beta(\ell+1)]$ where $\pi_{i,\ell}$ and $\pi_{i,\ell+1}$ are defined as above.

4. For all $j \in [1..m]$, we create a new constraint R_j that mimics the behavior of the clause C_j .

First, fix some arbitrary index $\ell \in [1..b]$ such that all variables of clause C_j appear in bag $\beta(\ell)$. This is possible as the primal graph contains a clique on the vertices of a single clause.

The constraint R_j observes the variables corresponding the all groups $V_i \cap \beta(\ell)$ where $i \in V(C_j)$. Formally, we define $\text{scp}(R_j) := \{x_{i,\ell,\gamma} \mid i \in V(C_j), \gamma \in [1..g]\}$.

The constraint R_j accepts if and only if there exists some $i \in V(C_j)$ such that $\pi_{i,\ell}$ satisfies the clause C_j .

For all other inputs, the constraint R_j rejects.

We denote the resulting instance of $q\text{-CSP-}B$ as ψ .

Correctness. As a next step, we prove the correctness of the reduction. That is, we show that the number of solutions of ϕ and ψ is identical. For this we show that the functions $\sigma_{\star,\star}$ provide a bijection between the satisfying assignments of ϕ and the satisfying assignments of ψ .

Claim 4.1.10.2. *Assume that $\pi: V(\phi) \rightarrow \{0,1\}$ satisfies ϕ . Then, the assignment*

$$(\sigma_{1,1}, \dots, \sigma_{t,b})(\pi) = \left(\sigma_{1,1}(\pi[V_1 \cap \beta(1)]), \dots, \sigma_{t,b}(\pi[V_t \cap \beta(b)]) \right)$$

satisfies ψ . Hence, if ϕ is satisfiable, then ψ is satisfiable.

Proof of Claim. Let $\pi: V(\phi) \rightarrow \{0,1\}$ be an assignment that satisfies ϕ . For ease of notation, for all $i \in [1..t]$ and for all $\ell \in [1..b]$, we denote by $\pi_{i,\ell}: V_i \cap \beta(\ell) \rightarrow \{0,1\}$ the restriction of π to the variables from the set $V_i \cap \beta(\ell)$, that is, we write $\pi_{i,\ell} := \pi[V_i \cap \beta(\ell)]$.

We first observe that all constraints $A_{i,\ell}$ and $C_{i,\ell}$ are satisfied by the assignments $\sigma_{i,\ell}(\pi_{i,\ell})$ by the definition of the functions $\sigma_{i,\ell}$.

Now consider a constraint R_j for some $j \in [1..m]$. By the construction of ψ , this constraint corresponds to the clause C_j of ϕ .

Since π satisfies ϕ , the assignment π especially satisfies clause C_j . Thus, there is some group $i \in V(C_j) \subseteq [1..t]$ such that $\pi_{i,\ell}$ satisfies the clause C_j where ℓ is chosen appropriately. Now, by the definition of the functions $\sigma_{i,\ell}$, we immediately conclude that the constraint R_j is satisfied. $\triangleleft$

As a next step, we prove the reverse direction of the correctness.

Claim 4.1.10.3. *Assume the assignment $\tau: V(\psi) \rightarrow [1..B]$ satisfies ψ . Then, there is a unique satisfying assignment $\pi: V(\phi) \rightarrow \{0, 1\}$ for ϕ such that $(\sigma_{1,1}, \dots, \sigma_{t,b})(\pi) = \tau$.*

Hence, if ψ is satisfiable, then ϕ is satisfiable.

Proof of Claim. Let $\tau: V(\psi) \rightarrow [1..B]$ be a satisfying assignment for ψ .

By the definition of the constraints $A_{i,\ell}$, we know that the assignment to $\text{scp}(R_j)$ under τ must encode a valid assignment to the variables of the set $V_i \cap \beta(\ell)$ for all $i \in V(C_j)$ and $\ell \in [1..b]$. We denote these assignments by $\pi_{i,\ell}$.

Moreover, as the constraints $C_{i,\ell}$ are satisfied, we know that the assignments $\pi_{i,\ell}$ and $\pi_{i,\ell+1}$ agree on the variables from $\beta(\ell) \cap \beta(\ell+1)$. From the fact that, for each variable v of ϕ , the bags containing this variable form a subpath of the path of the path decomposition, the variable v receives a well-defined value. Moreover, as each variable of ϕ appears in at least one clause and thus, in at least one bag, every variable receives a well-defined value.

We define the assignment $\pi: V(\phi) \rightarrow \{0, 1\}$ such that $\pi(v)$ is the unique value for v that is encoded by τ , that is, intuitively the assignment π is the union of all $\pi_{i,\ell}$.

Hence, we also get that $\sigma_{i,\ell}(\pi[V_i \cap \beta(\ell)]) = (\tau(x_{i,\ell,1}), \dots, \tau(x_{i,\ell,g}))$ for all $i \in [1..t]$ and $\ell \in [1..b]$. By the definition of the functions $\sigma_{i,\ell}$ as injective mappings, we conclude that there is no other assignment π' with $(\sigma_{1,1}, \dots, \sigma_{t,b})(\pi') = \tau$.

It remains to show that π satisfies ϕ . By the definition of R_j , there must be some index $i \in V(C_j)$ and a bag $\ell \in [1..b]$ such that the assignment $\pi_{i,\ell}$ satisfies the clause C_j . As this holds for all clauses, this concludes the proof. $\triangleleft$

As a last step, we analyze the constructed instance ψ .

Claim 4.1.10.4. *The CSP-instance ψ contains $t \cdot b \cdot g$ variables and $2 \cdot t \cdot b + m$ constraints and the arity of the constraints is bounded by $k \cdot g$.*

The instance ψ and a path decomposition of the primal graph of width at most $(t+1) \cdot g$ and can be constructed in time $2^q \cdot |\phi|^{\mathcal{O}(1)}$.

Proof of Claim. The bound on the number of variables and clauses follows immediately from the construction as we introduce g variables for each of the sets $V_i \cap \beta(\ell)$ of variables of ϕ for all choices of $i \in [1..t]$ and $\ell \in [1..b]$.

As the construction starts with a formula in k -CNF, each constraint R_j observes at most k sets $V_i \cap \beta(\ell)$ each of which has size at most g . Moreover, all other constraints observe at most two of these sets which concludes the bound on the arity.

For the bound of the pathwidth, we iterate over all $i \in [1..t]$ and $\ell \in [1..b]$ and replace the variables in $V_i \cap \beta(\ell)$ by the variables $x_{i,\ell,\star}$. Then, this covers all edges created by the constraints C_j and the constraints $A_{i,\ell}$.

It remains to cover the edges introduced by the constraints $C_{i,\ell}$. For this we first modify the path decomposition from the previous step. Consider the bags $\beta(\ell)$ and $\beta(\ell + 1)$. We create g new bags between these two bags such that the i th bag contains the vertex set

$$\left(\beta(\ell) \setminus \bigcup_{i=1}^{i-1} \{x_{i,\ell,\gamma} \mid \gamma \in [1 \dots g]\} \right) \cup \bigcup_{i=1}^i \{x_{i,\ell+1,\gamma} \mid \gamma \in [1 \dots g]\}.$$

Intuitively, these bags interpolate between the bags $\beta(\ell)$ and $\beta(\ell + 1)$.

With this construction we also directly get that the pathwidth of the primal graph of ψ is bounded by $(t + 1) \cdot g$. $\blacktriangleleft$

Now we have everything ready to finish the proof based on the previous construction.

Finalizing the Proof. Recall that for some $\varepsilon > 0$, we assume that for all q , there is an algorithm that decides if a given q -CSP- B instance ξ that is given with a path decomposition of the primal graph of width $\text{pw}(\xi)$ is satisfiable (respectively computes the number of satisfying assignments) in time $(B - \varepsilon)^{\text{pw}(\xi)} \cdot |\xi|^{\mathcal{O}(1)}$.

It remains to choose the parameter g which we have to do carefully because of the rounding and the fact that we restrict the parity of the constraints.

We set $\varepsilon' := \log_B(B - \varepsilon) < 1$. Now, choose an $\alpha > 1$ such that $\alpha \cdot \varepsilon' < 1$ which is always possible since $\varepsilon' < 1$. Then, we set $\delta' := \log(2 - \delta) := \alpha \cdot \varepsilon'$.

Next, we choose g large enough such that $g \cdot \log B \leq \alpha \cdot \lfloor g \cdot \log B - 2 \rfloor$ which is possible for sufficiently large g since $\alpha > 1$.

With all parameters set, we first analyze the width of the path decomposition for ψ . By Claim 4.1.10.4 and our choice of parameters ($t = \lceil (\text{pw}(\phi) + 1)/q \rceil$ and $q = \lfloor g \cdot \log B - 2 \rfloor$), we get

$$\begin{aligned} \text{pw}(\psi) &\leq (t + 1) \cdot g = \left(\left\lceil \frac{\text{pw}(\phi) + 1}{q} \right\rceil + 1 \right) \cdot g \leq \left(\frac{\text{pw}(\phi) + 1}{q} + 2 \right) \cdot g \leq \left(\frac{\text{pw}(\phi)}{q} + 3 \right) \cdot g \\ &\leq \frac{\text{pw}(\phi) \cdot g}{\lfloor g \cdot \log B - 2 \rfloor} + 3g \\ &\leq \frac{\alpha \cdot \text{pw}(\phi)}{\log B} + 3g. \end{aligned}$$

Now we apply the algorithm for q -CSP- B on the previously constructed instance ψ to solve k -SAT on the given instance ϕ . This procedure runs in time

$$2^q \cdot |\phi|^{\mathcal{O}(1)} + (B - \varepsilon)^{\text{pw}(\psi)} \cdot |\psi|^{\mathcal{O}(1)} \leq 2^{g \cdot \log B} \cdot |\phi|^{\mathcal{O}(1)} + (B - \varepsilon)^{\frac{\alpha \cdot \text{pw}(\phi)}{\log B} + 3g} \cdot |\psi|^{\mathcal{O}(1)}.$$

Further, the value of g depends only on B and ε . Moreover, the size of ψ depends only on the number of variables and clauses of the formula ϕ . Thus, the factors $(B - \varepsilon)^{3g}$ and $2^{g \cdot \log B} = B^g$ contribute a constant (depending only on B and ε) to the overall running time which is hidden by the polynomial term.

With this observation we bound the running time by

$$\begin{aligned}
 2^{g \cdot \log B} \cdot |\phi|^{\mathcal{O}(1)} + (B - \varepsilon)^{\frac{\alpha \cdot \text{pw}(\phi)}{\log B} + 3g} \cdot |\psi|^{\mathcal{O}(1)} &\leq (B - \varepsilon)^{\frac{\alpha \cdot \text{pw}(\phi)}{\log B}} \cdot |\phi|^{\mathcal{O}(1)} \\
 &= 2^{\log(B - \varepsilon) \cdot \frac{\alpha \cdot \text{pw}(\phi)}{\log B}} \cdot |\phi|^{\mathcal{O}(1)} \\
 &= 2^{\varepsilon' \cdot \alpha \cdot \text{pw}(\phi)} \cdot |\phi|^{\mathcal{O}(1)} \\
 &= 2^{\delta' \cdot \text{pw}(\phi)} \cdot |\phi|^{\mathcal{O}(1)} \\
 &= (2 - \delta)^{\text{pw}(\phi)} \cdot |\phi|^{\mathcal{O}(1)}.
 \end{aligned}$$

We conclude that this algorithm violates pw-SETH and pw-#SETH as stated in Conjecture 4.1.4 and Conjecture 4.1.5, respectively. $\square$

In the next section, we extend pw-CSPH and pw-#CSPH to the variant of q -CSP- B where a variable might receive different assignments for different bags.

4.1.3 Multi-Assignments

As a next step, we consider CSPs with *multi-assignments*. That is, instead of assigning a single value to each variable, we can now assign multiple values to a single variable. More precisely, we allow that every variable might receive a different value for each bag it appears in. Lampis formalized this variant together with the definition of pw-CSPH and pw-SETH, and proved a lower bound for the variant under pw-SETH [96].

Note that this relaxation with multi-assignments can directly lead to a satisfiable CSP-instance by duplicating the bags sufficiently often. Hence, we need some further restriction. We additionally require that the multi-assignments are *monotone*. Assume a variable v receives values $b_1, \dots, b_t \in [1..B]$ in the t bags it appears in, then the assignment is monotone on v if $b_1 \leq b_2 \leq \dots \leq b_t$.

In addition to these variables that might receive multiple values, we also consider variables where the assignment must be consistent, that is, the variables receive only a single value for all bags. Naturally, we only allow few of these variables. Otherwise, we could replace all variables with a monotone assignment by these variables with a consistent assignment.

We use the variables with a consistent assignment to ensure that the other variables *do not* receive multiple values but actually just receive a single value. That is, we ensure that all variables receive a consistent assignment *without* imposing this restriction explicitly.

The basic idea to achieve this consistency is as follows. We use the variables which are promised to be consistent, to encode a check sum. This check sum counts, for each possible value of the variables, the number of variables that receive this particular value when they are introduced. Similarly, we also count the variables according to their value when they are forgotten. Then, because the assignment is monotone, we conclude that all variables receive a single consistent assignment if the check sums are equal.

We show that pw-SETH and the corresponding counting version pw-#SETH imply Conjectures 4.1.11 and 4.1.12.

Conjecture 4.1.11 (MONO-pw-CSPH). *For all $B \geq 2$ and $\varepsilon > 0$, there exists a q such that no algorithm can, for a q -CSP- B instance ϕ that is given with*

4 A Toolkit for Proving Lower Bounds

- a partition $V_1 \dot{\cup} V_2$ of the variables,
- a nice path decomposition of the primal graph of ϕ of width w
- where every bag contains at most $\mathcal{O}(B \cdot \log|V_1|)$ variables from V_2 ,
- an injective mapping c from the constraints of ϕ to the bags such that the bag contains the variables of the respective constraint,
- the promise that every satisfying multi-assignment that is consistent on V_2 is also consistent on V_1 ,

decide if there is a monotone satisfying multi-assignment that is consistent on V_2 in time

$$(B - \varepsilon)^w \cdot |\phi|^{\mathcal{O}(1)}.$$

We define the conjecture for the counting version analogously.

Conjecture 4.1.12 (MONO-pw-#CSPH). *For all $B \geq 2$ and $\varepsilon > 0$, there exists a q such that no algorithm can, for a q -CSP- B instance ϕ that is given with*

- a partition $V_1 \dot{\cup} V_2$ of the variables,
- a nice path decomposition of the primal graph of ϕ of width pw
- where every bag contains at most $\mathcal{O}(B \cdot \log|V_1|)$ variables from V_2 ,
- an injective mapping c from the constraints of ϕ to the bags such that the bag contains the variables of the respective constraint,
- the promise that every satisfying multi-assignment that is consistent on V_2 is also consistent on V_1 ,

count the number of monotone satisfying multi-assignment that are consistent on V_2 in time

$$(B - \varepsilon)^{\text{pw}} \cdot |\phi|^{\mathcal{O}(1)}.$$

Formally, the goal of this section is the proof of Main Theorem I.3.

Main Theorem I.3. *If pw-SETH holds, then MONO-pw-CSPH holds.*

Moreover, if pw-#SETH holds, then MONO-pw-#CSPH holds.

Both results hold even if all constraints have a fixed parity.

We prove Main Theorem I.3 by a reduction from the original q -CSP- B problem and combine it with pw-CSPH and pw-#CSPH.

Notation and Preliminaries. As a first step toward the formal proof, we define a *multi-assignment* formally. This closely mirrors Definition 3.2 in [96].

Definition 4.1.13 (Multi-Assignment). Denote by ψ a q -CSP-B instance with variables $V(\psi)$ and constraints $C(\psi)$ that is given with a nice path decomposition with bags $\beta(1), \dots, \beta(b)$ and an injective function $c: C(\psi) \rightarrow [1..b]$ such that $\beta(c(C))$ contains all variables that appear in the constraint $C \in C(\psi)$.

We say a function τ that maps an index $\ell \in [1..b]$ of a bag and a variable $v \in \beta(\ell)$ to a value in $[1..B]$ is a multi-assignment for ψ .

We define the following properties of a multi-assignment.

Definition 4.1.14 (Properties of Multi-Assignments). Let τ be a multi-assignment as in Definition 4.1.13. We say that

- τ is monotone if, for all variables $v \in V(\psi)$ and bags $\beta(\ell), \dots, \beta(\ell+r)$ containing v we have $\tau(\ell, v) \leq \tau(\ell+1, v) \leq \dots \leq \tau(\ell+r, v)$,
- τ is consistent for a variable $v \in V(\psi)$ if there is a value $\pi_v \in [1..B]$ such that for all bags $\beta(\ell)$ containing v we have $\tau(\ell, v) = \pi_v$,
- τ is consistent if τ is consistent for all variables $v \in V(\psi)$, and
- τ is satisfying if, for all constraints $C \in C(\psi)$, the assignment $\tau_C: \beta(c(C)) \rightarrow [1..B]$ with $\tau_C(v) := \tau(c(C), v)$ satisfies the constraint C .

Reduction. The reduction from Lemma 4.1.15 is the main ingredient for the proof of Main Theorem I.3.

Lemma 4.1.15. The following holds for $q \geq 4$. Given a q -CSP-B instance ψ along with a path decomposition of the primal graph of width pw . In polynomial-time we can construct another instance ϕ of q -CSP-B along with a path decomposition of its primal graph such that

- the variables of ϕ can be partitioned as $V_1 \dot{\cup} V_2$ with $V_1 = V(\psi)$ and $|V_2| \in \mathcal{O}(|V_1| \cdot B \cdot \log|V_1|)$,
- each bag contains at most $\text{pw}(\phi) + 1$ variables from V_1 and at most $2B \cdot \log|V_1| + \mathcal{O}(1)$ variables from V_2 ,
- every satisfying multi-assignment that is consistent on V_2 is also consistent on V_1 ,
- the number of satisfying assignments for ψ is equal to the number of monotone satisfying assignments for ϕ that are consistent on V_2 , and
- if the parity of each constraint of ψ is defined, then the parity of each constraint of ϕ is defined.

Proof. We use the existing reduction from Lemma 3.9 in [96] as a basis but adjust it non-trivially such that the result also follows for the counting version.

We denote by $C(\psi)$ the set of constraints of ψ and by $V(\psi)$ the set of variables of ψ .

We assume that the given path decomposition is nice and denote the bags by $\beta(1), \dots, \beta(r)$. Moreover, we assume that the path decomposition is such that there is some injective function $c: C(\psi) \rightarrow [1..r]$ such that, for all constraints $C \in C(\psi)$, the bag $\beta(c(C))$ contains all variables of the clause of C . This assumption is possible as the vertices of a constraint induce a clique in the primal graph, and we can duplicate bags to make the function c injective.

For all variables $v \in V(\psi)$, we define the two indices $\text{intro}(v) \in [1..r]$ and $\text{forget}(v) \in [1..r]$ as the indices of the two unique bags such that v is introduced in bag $\beta(\text{intro}(v))$ and v is forgotten in bag $\beta(\text{forget}(v))$. That is, we have $\beta(\text{intro}(v)) \setminus \beta(\text{intro}(v) - 1) = \{v\}$ and $\beta(\text{forget}(v) - 1) \setminus \beta(\text{forget}(v)) = \{v\}$. We extend the functions $\text{intro}(\cdot)$ and $\text{forget}(\cdot)$ in the natural way to sets of variables.

We denote by $N + 1$ the number of bits that are necessary to encode a number from $[0..n]$ in binary, that is, we set $N = \lfloor \log(n) \rfloor$.

We construct the new instance ϕ as follows. Figure 4.1 contains an illustration of parts of this construction.

1. We copy all variables and constraints of the CSP-instance ψ . The variables of ψ form the variable set V_1 of ϕ . Then, all remaining variables, that is, the following variables, belong to the set V_2 .
2. For all $v \in V_1$, we create a new variable $x_{\text{intro}(v)}$ and a new constraint D_v that requires $(x_{\text{intro}(v)} = v)$. Similarly, we create a new variable $\bar{x}_{\text{forget}(v)}$ and a new constraint $\bar{D}_v$ that requires $(\bar{x}_{\text{forget}(v)} = v)$.

Because of the multi-assignment, the constraints D_v and $\bar{D}_v$ are not redundant.

As the path decomposition is nice, at most one variable is introduced or forgotten per bag and hence, these new variables are well-defined.

Note that for the constraints D_v and $\bar{D}_v$, the parity is even.

3. For all $i \in [0..|V_1|]$, all $b \in [1..B]$, and all $v \in [0..N]$, we create a variable $y_{i,b}^v$ and assume it takes values from $\{0, 1\}$ only. To achieve this, we simply interpret the value 2 as 0 and ensure that the variable has value 1 or 2 whenever it appears in a constraint.
4. We order $\text{intro}(V_1)$ and write $\text{intro}(V_1) = \{\ell_1, \dots, \ell_{|V_1|}\}$ such that $\ell_i < \ell_j$ if and only if $i < j$. Then, for all $i \in [1..|V_1|]$, and all $b \in [1..B]$, we create a set of constraints $A_{i,b}$ which ensures that
 - if $x_{\ell_i} = b$, then $(y_{i,b}^0, \dots, y_{i,b}^N)$ represents the sum of 1 and the number represented by $(y_{i-1,b}^0, \dots, y_{i-1,b}^N)$, and
 - otherwise, that is, if $x_{\ell_i} \neq b$, then $y_{i-1,b}^v = y_{i,b}^v$ for all $v \in [0..N]$.

When $B \geq 3$, we can interpret the value 3 as “0” to simplify the proof that the parity of the new constraints is defined.

Formally, for all $i \in [1..|V_1|]$, all $b \in [1..B]$, and all $v \in [0..N]$, we create another variable $z_{i,b}^v$ that takes values $\{0, 1\}$. Intuitively this variable encodes the carry-over for the addition with 1 or with 0 depending on x_{ℓ_i} .

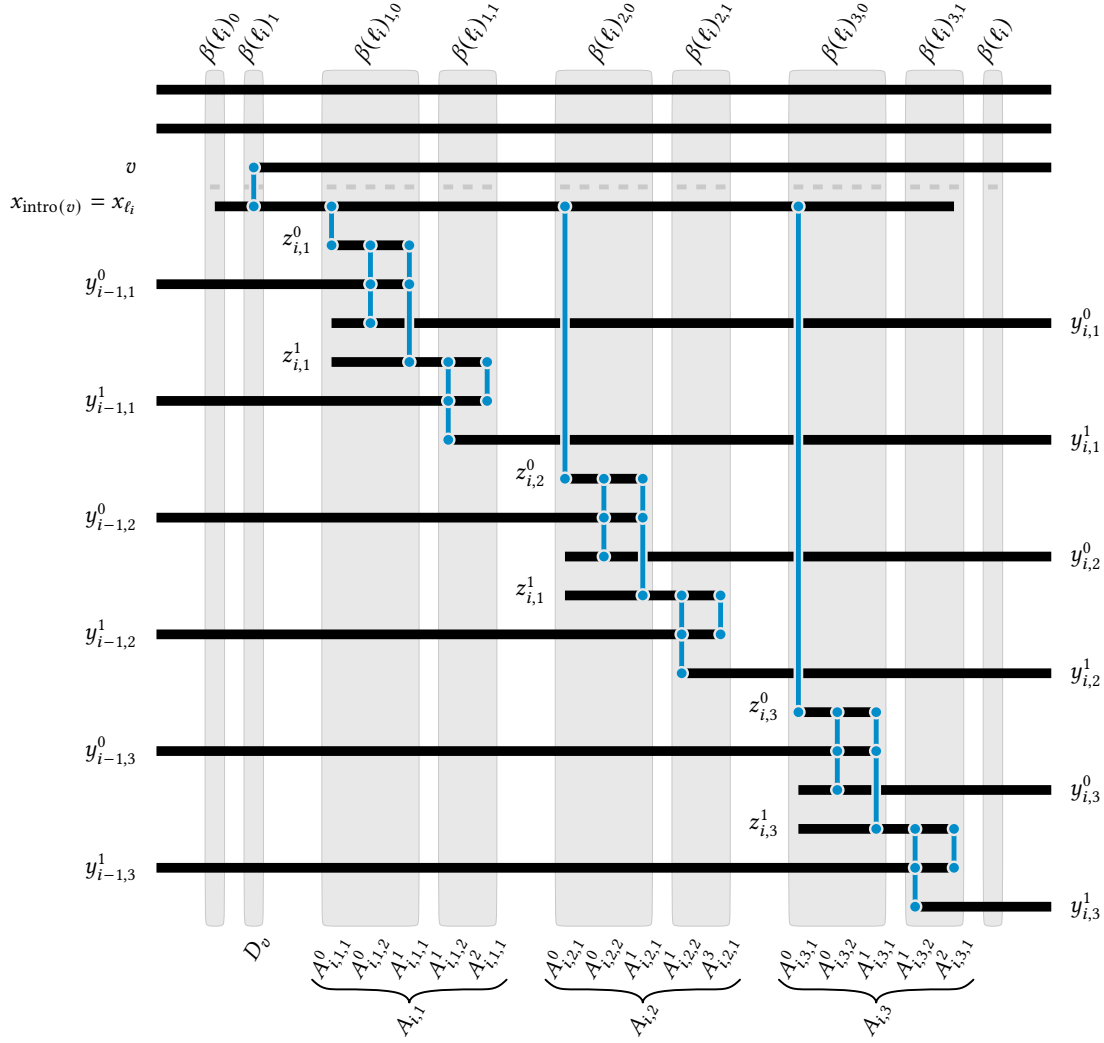


Figure 4.1: In the construction of Lemma 4.1.15 we add variables $y_{i,b}^v$ to count the original variables according to their value when they are introduced.

In this example we assume that variable v is introduced. The variables can have $B = 3$ states, and we encode the numbers with $N + 1 = 2$ bits.

The horizontal lines indicate the variables and the gray blocks indicate the different bags with their name on top. The vertical lines indicate the constraints, with their name at the bottom, that use the connected variables.

For each $b \in [1..B]$, we add a constraint $A_{i,b,1}^0$ which ensures that $(z_{i,b}^0 = 1)$ if and only if $(x_{\ell_i} = b)$. Otherwise, we have $(z_{i,b}^0 = 0)$.

Then, for all $b \in [1..B]$ and $v \in [0..N]$, we add two constraints $A_{i,b,1}^{v+1} := (z_{i,b}^{v+1} \leftrightarrow y_{i-1,b}^v \wedge z_{i,b}^v)$ and $A_{i,b,2}^v := (y_{i,b}^v \leftrightarrow y_{i-1,b}^v \oplus z_{i,b}^v)$ which ensure the addition. The constraint $A_{i,b,1}^{N+1}$ additionally ensures that $(z_{i,b}^{N+1} = 0)$, that is, there is no overflow for the

most significant bit.

To ensure that the parity of all constraints is defined, we introduce additional variables $p_{i,b}^\nu$ for all $i \in [1 \dots |V_1|]$, $b \in [1 \dots B]$, and $\nu \in [0 \dots N+1]$.

We add variable $p_{i,b}^\nu$ to the scope of constraint $A_{i,b,1}^\nu$ and require that this variable takes only value 1 or 2 such that the parity of the accepted inputs is always equal.

5. For all $i \in [0 \dots |V_1|]$, all $b \in [1 \dots B]$, and all $\nu \in [1 \dots N]$, we create a variable $\bar{y}_{i,b}^\nu$ and assume it takes values from $\{0, 1\}$ only.
6. By ordering $\text{forget}(V_1)$, we write $\text{forget}(V_1) = \{\bar{\ell}_1, \dots, \bar{\ell}_{|V_1|}\}$ such that $\bar{\ell}_i < \bar{\ell}_j$ if and only if $i < j$. Then, for all $i \in [1 \dots |V_1|]$, and all $b \in [1 \dots B]$, we create a set of constraints $\bar{A}_{i,b}$ which ensures that
 - if $\bar{x}_{\bar{\ell}_i} = b$, then $(\bar{y}_{i,b}^0, \dots, \bar{y}_{i,b}^N)$ represents the sum of 1 and the number represented by $(\bar{y}_{i-1,b}^0, \dots, \bar{y}_{i-1,b}^N)$, and
 - otherwise, that is, if $\bar{x}_{\bar{\ell}_i} \neq b$, then $\bar{y}_{i-1,b}^\nu = \bar{y}_{i,b}^\nu$ for all $\nu \in [0 \dots N]$.

Similar to before, we define the constraints formally as follows. For all $i \in [1 \dots |V_1|]$, all $b \in [1 \dots B]$, and all $\nu \in [0 \dots N+1]$, we create another variable $\bar{z}_{i,b}^\nu$ that takes values $\{0, 1\}$. As before, this variable encodes the carry-over for the addition with 1 or with 0 depending on $\bar{x}_{\bar{\ell}_i}$.

For each $b \in [1 \dots B]$, we add a constraint $\bar{A}_{i,b,1}^0$ which ensures that $(\bar{z}_{i,b}^0 = 1)$ if and only if $(\bar{x}_{\bar{\ell}_i} = b)$. Otherwise, we have $(\bar{z}_{i,b}^0 = 0)$.

Then, we add two constraints $\bar{A}_{i,b,1}^{\nu+1} := (\bar{z}_{i,b}^{\nu+1} \leftrightarrow \bar{y}_{i-1,b}^\nu \wedge \bar{z}_{i,b}^\nu)$ and $\bar{A}_{i,b,2}^\nu := (\bar{y}_{i-1,b}^\nu \leftrightarrow \bar{y}_{i,b}^\nu \oplus \bar{z}_{i,b}^\nu)$ which ensure the addition. The constraint $\bar{A}_{i,b,1}^{N+1}$ additionally ensures that $(\bar{z}_{i,b}^{N+1} = 0)$.

Moreover, we introduce a variable $\bar{p}_{i,b}^\nu$ and add it to the scope of the constraint $\bar{A}_{i,b,1}^\nu$. We modify this constraint to ensure that the variable $\bar{p}_{i,b}^\nu$ takes only two values (of different parity) such that the parities of all accepted inputs of the constraint $\bar{A}_{i,b,1}^\nu$ are equal.

7. For all $b \in [1 \dots B]$ and all $\nu \in [0 \dots N]$, we create a constraint Z_b^ν that ensures $y_{0,b}^\nu = \bar{y}_{0,b}^\nu = 0$.
8. For all $b \in [1 \dots B]$ and all $\nu \in [0 \dots N]$, we create a constraint E_b^ν that ensures $y_{|V_1|,b}^\nu = \bar{y}_{|V_1|,b}^\nu$.

This concludes the construction of the new instance ϕ .

Constructing the Path Decomposition. Before we prove the correctness, we first modify the path decomposition of ψ to obtain a path decomposition of the primal graph of ϕ . Simultaneously, we define the new mapping f from the constraints of ϕ to the bags of the new path decomposition. Both constructions are necessary to prove the correctness as the multi-assignments depend on these definitions.

To simplify notation, we first define two families of variable sets. For all $i \in [0 \dots |V_1|]$, all $b \in [1 \dots B]$, and all $v \in [0 \dots N]$, we set

$$Y_{i,b}^v := \bigcup_{b'=1}^{b-1} \{y_{i,b',v'} \mid v' \in [0 \dots N]\} \cup \{y_{i,b,v'} \mid v' \leq v\}$$

and

$$\bar{Y}_{i,b}^v := \bigcup_{b'=1}^{b-1} \{\bar{y}_{i,b',v'} \mid v' \in [0 \dots N]\} \cup \{\bar{y}_{i,b,v'} \mid v' \leq v\}.$$

We modify the path decomposition in four steps. For each step, we consider the modified decomposition that is obtained by the previous step.

1. We create $B \cdot (N + 1)$ new bags $\beta(0)_{b,v,1} := Y_{0,b}^v$ for all $b \in [1 \dots B]$ and $v \in [0 \dots N]$ and add them to the beginning of the path decomposition. Note that we first increase v and then b . Similarly, we add $B \cdot (N + 1)$ additional bags $\beta(0)_{b,v,2} := Y_{0,B}^N \cup Y_{0,b}^v$ for all $b \in [1 \dots B]$ and $v \in [0 \dots N]$ where we again first increase v and then b .

We define the mapping f such that the constraint Z_b^v is mapped to the bag $\beta(0)_{b,v,2}$.

2. For all $i \in [1 \dots |V_1|]$, we duplicate the bag $\beta(\ell_i)$ twice, and denote the first copy by $\beta(\ell_i)_0$, the second copy by $\beta(\ell_i)_1$, and the third copy by $\beta(\ell_i)$.

Say the variable v is introduced at $\beta(\ell_i)_0$. Then, we remove v from the bag $\beta(\ell_i)_0$ and add the variable $x_{\text{intro}(v)}$ to the bags $\beta(\ell_i)_0$ and $\beta(\ell_i)_1$. We assign the constraint D_v to bag $\beta(\ell_i)_1$.

Variable v is now introduced at bag $\beta(\ell_i)_1$.

Moreover, we add the vertices from $Y_{i-1,B}^N$ to all bags between $\beta(\ell_{i-1})$ and $\beta(\ell_i)_1$ including these two bags where we assume that $\beta(\ell_0) = \beta(0)_{B,N,2}$.

Now, we iterate over all $b \in [1 \dots B]$ and add the following sequence of bags between $\beta(\ell_i)_1$ and bag $\beta(\ell_i)$.

We create this sequence by iterating over all $v \in [0 \dots N]$, and add the following bag

$$\beta(\ell_i)_{b,v} := \beta(\ell_i) \cup \{x_{\text{intro}(v)}\} \cup (Y_{i-1,B}^N \setminus Y_{i-1,b}^{v-1}) \cup Y_{i,b}^v \cup \{z_{i,b}^v, z_{i,b}^{v+1}\}$$

to the decomposition. That is, we first increase the value of v . Once, we reach $v = N$, we increase the value of b .

Observe, that these bags cover the constraints $A_{i,b,1}^v$ and $A_{i,b,2}^v$. However, to achieve an injective mapping from the constraints to the bags, we duplicate each of these bags at most three times such that each constraint is assigned a unique bag.

3. Analogously, for all $i \in [1 \dots |V_1|]$, we duplicate the bag $\beta(\bar{\ell}_i)$ twice. We denote the first copy of this bag by $\beta(\bar{\ell}_i)_0$ the second copy by $\beta(\bar{\ell}_i)_1$, and the third copy by $\beta(\bar{\ell}_i)$.

Assume that variable v is forgotten at $\beta(\bar{\ell}_i)$, that is, $\text{forget}(v) = \bar{\ell}_i$. Then, we add the variable v to bag $\beta(\bar{\ell}_i)_0$, and add the variable $\bar{x}_{\text{forget}(v)}$ to the two bags $\beta(\bar{\ell}_i)_0$ and $\beta(\bar{\ell}_i)_1$.

Variable v is now forgotten at bag $\beta(\bar{\ell}_i)_1$.

We assign the relation $\overline{D}_v$ to bag $\beta(\bar{\ell}_i)_0$.

We add the variable set $\overline{Y}_{i-1,B}^N$ to all bags between $\beta(\bar{\ell}_{i-1})$ and $\beta(\bar{\ell}_i)_1$ including both endpoints.

Similarly to the previous step, we add the bags

$$\beta(\bar{\ell}_i)_{b,v} := \beta(\bar{\ell}_i) \cup \{\bar{x}_{\text{forget}(v)}\} \cup (\overline{Y}_{i-1,B}^N \setminus \overline{Y}_{i-1,b}^{v-1}) \cup \overline{Y}_{i,b}^v \cup \{\bar{z}_{i,b}^v, \bar{z}_{i,b}^{v+1}\}$$

between $\beta(\bar{\ell}_i)_1$ and $\beta(\bar{\ell}_i)$ where we first increase $v \in [0..N]$ and then increase $b \in [1..B]$.

Again, we can duplicate these bags sufficiently often to cover all constraints that we introduced to describe $\overline{A}_{i,b}$.

4. Assume that $\beta(r)$ denotes the last bag of the path decomposition. We add the variable set $Y_{|V_1|,B}^N$ to all bags from $\beta(\ell_{|V_1|})$ to $\beta(r)$ including these two bags. Similarly, we add the variable set $\overline{Y}_{|V_1|,B}^N$ to all bags from $\beta(\bar{\ell}_{|V_1|})$ to $\beta(r)$ including these two bags.

It remains to add the bags for the constraints E_b^v . To ensure this, we append $B \cdot (N+1)$ bags with the variables $Y_{|V_1|,b}^v \cup \overline{Y}_{|V_1|,b}^v$ to the last bag $\beta(r)$ where we start with $b = B$ and $v = N$ and first decrease v and then b .

Next, we prove that this modification does not change the width of the decomposition too much.

Claim 4.1.15.1. *The path decomposition can be constructed in time polynomial in the size of ψ . Every bag contains at most $\text{pw}(\psi) + 1$ vertices of V_1 and at most $2 \cdot (B+1) \cdot (N+1) + \mathcal{O}(1)$ vertices of V_2 .*

Proof of Claim. The bound on the width follows directly from our construction. Indeed, observe that the size of those bags added for the constraints E_b^v and Z_b^v are dominated by the other bags.

Hence, the change of the width is determined by the size of the bags $\beta(\ell_i)_{b,v}$ and $\beta(\bar{\ell}_i)_{b,v}$. From the construction we know that each of the second and third step above increases the size by at most $1 + B \cdot (N+1) + (N+1) + 2$. Note that in the second (resp. third) step we consider the vertices in the order in which they are introduced (resp. forgotten). Hence, the path decomposition is actually well-defined. Moreover, all these vertices are contained in the set V_2 and thus, the claim follows. $\triangleleft$

Correctness. Now we have everything ready to prove the correctness of the construction. We consider both directions individually and start with the reverse direction.

Claim 4.1.15.2. *Every satisfying monotone multi-assignment for ϕ that is consistent on V_2 is also consistent on V_1 and thus, satisfies ψ .*

Proof of Claim. Denote by τ a monotone multi-assignment for ϕ that is satisfying.

As τ satisfies ϕ , it especially satisfies all clauses of ψ . If we assume that τ is consistent on V_1 , we know that the restriction of τ to the variables of V_1 gives a well-defined assignment that satisfies ψ . Hence, in the remainder we show that τ is consistent on V_1 if it is consistent on V_2 .

By assumption, we know that the multi-assignment τ is monotone. That is, for every variable $v \in V_1$, it must hold that $\tau(\text{intro}(v), v) \leq \tau(\text{intro}(v) + 1, v) \leq \dots \leq \tau(\text{forget}(v) - 1, v)$.

Now, assume there is some variable where the multi-assignment is not consistent. We pick $v \in V_1$ as the variable with the smallest $b \in [1..B]$ such that $b = \tau(\text{intro}(v), v) \neq \tau(\text{forget}(v) - 1, v)$.

As the assignment τ is satisfying, we know that all constraints denoted by the sets $A_{i,b}$ and $\bar{A}_{i,b}$ are satisfied, that is, the constraints $A_{i,b,1}^v$, $A_{i,b,2}^v$, $\bar{A}_{i,b,1}^v$, and $\bar{A}_{i,b,2}^v$ are satisfied. By the assumption that the assignment τ is consistent on V_2 , we know that these constraints actually encode addition and thus, the variables $y_{|V_1|,b}^\star$ encode the number of variables from V_1 that receive value b when they are introduced. Especially, the variable v contributes to this count. For this to work it is crucial that we copy the value of v to the variable $x_{\text{intro}(v)}$ in the same bag where it is introduced. This is why we introduce the variable $x_{\text{intro}(v)}$ before the variable v .

Similarly, the variables $\bar{y}_{|V_1|,b}^\star$ encode the number of variables from V_1 that have value b when they are forgotten. By our choice of v , we know that the variable v *does not* contribute to this count.

However, by the constraints E_b^v , we know that both numbers must be equal. Thus, there must be some other variable $v' \in V_1$ where $\tau(\text{forget}(v'), v') = b$ but $\tau(\text{intro}(v'), v') \neq b$. Our choice of v yields that $\tau(\text{intro}(v'), v') > b$ which is an immediate contradiction to our assumption that τ is a monotone assignment. Hence, the assignment τ is actually consistent on the variables V_1 . $\triangleleft$

As a last step, we prove the forward direction.

Claim 4.1.15.3. *Every satisfying assignment for ψ can be extended to a unique satisfying assignment for ϕ that is consistent on V_2 .*

Proof of Claim. Denote by π the satisfying assignment for ψ . We define the assignment τ for ϕ in the natural way. That is, the assignment τ agrees with π on the variables from $V_1 = V(\psi)$. Then, we set the variables $y_{i,b}^\star$ such that they correctly encode the number of variables among the first i introduced variables that receive value b when they are introduced. Similarly, the variables $\bar{y}_{i,b}^\star$ encode the number of the variables among the first i forgotten variables that receive value b when they are forgotten. As the assignment is consistent for all variables by definition, these two numbers are equal for $i = |V_1|$.

Moreover, we choose the variables $z_{i,b}^v$, $\bar{z}_{i,b}^v$, and $x_\ell, \bar{x}_\ell$ such that the remaining constraints are satisfied. This is possible as the assignment is consistent for all variables and the remaining variables are chosen appropriately.

It remains to prove the uniqueness. As the assignment needs to be consistent on V_2 , it must also be consistent on V_1 by Claim 4.1.15.2. Hence, by the assumption that we extend the assignment for ψ , the variables in V_1 must receive the values defined by π . Moreover, the variables $x_{\text{intro}(v)}$ and $\bar{x}_{\text{forget}(v)}$ copy the state of the variable v in the bag where it is introduced and forgotten, respectively. Thus, these variables cannot change their value between their

introduce bag and the bag where they are used the first time. Similarly, this also holds for the case when the variable is forgotten.

As there is no choice for the values of the variables in V_2 , we conclude that there is a unique assignment for ϕ with the claimed properties. $\triangleleft$

Before we conclude the proof, we argue about the arity and the parity of the constraints of ϕ . Clearly, as we copy the constraints of ψ , their arity or parity do not change. Moreover, all new constraints have arity at most 4. Thus, it remains to argue about the parity of the new constraints. The constraints D_v and $\bar{D}_v$, as well as Z_b^v and E_b^v have even parity as they check for equality of two variables.

It is straightforward to check that the constraints $A_{i,b,2}^v$ have even parity because of the $\oplus$ operation. As the constraints $A_{i,b,1}^v$ include an additional variable to ensure that the parity is defined, we conclude that the parity of all constraints is defined.

This concludes the proof of Lemma 4.1.15 by the combination of Claims 4.1.15.1 to 4.1.15.3. $\square$

As a last step, we combine the lower bound for q -CSP- B under SETH (respectively the counting versions thereof) from Lemma 4.1.10 with the reduction to monotone q -CSP- B from Lemma 4.1.15, to prove Main Theorem I.3.

Main Theorem I.3. *If pw-SETH holds, then MONO-pw-CSPH holds.*

Moreover, if pw-#SETH holds, then MONO-pw-#CSPH holds.

Both results hold even if all constraints have a fixed parity.

Proof. We only focus on the result for the decision version, as the result for the counting version follows analogously.

Fix some $B \geq 2$ and $\varepsilon > 0$. Consider some algorithm $\mathcal{A}$ that takes as input a q -CSP- B instance ϕ with a partition $V_1 \dot{\cup} V_2$ of the variables and a path decomposition of the primal graph of width pw such that every bag contains at most $\mathcal{O}(B \cdot \log|V_1|)$ variables from V_2 . Now assume that this algorithm $\mathcal{A}$ decides, for all q , if the formula ϕ has a monotone satisfying assignment that is consistent on V_2 in time $(B - \varepsilon)^{\text{pw}} \cdot |\phi|^{\mathcal{O}(1)}$. That is, we assume that MONO-pw-CSPH is false.

Now consider some q -CSP- B instance ϕ where each constraint has a fixed parity. We apply the reduction from Lemma 4.1.15 to obtain another instance ψ of q -CSP- B along with a path decomposition of width $\text{pw}(\psi)$. Then, we apply the algorithm $\mathcal{A}$ to the instance ψ along with its path decomposition and return the result as the result for the given instance ϕ .

We observe that, by the properties of the reduction in Lemma 4.1.15, the instance ψ satisfies the preconditions of the algorithm $\mathcal{A}$ mentioned above. For the case of the counting version, we observe that this particular combination of the reduction and the algorithm $\mathcal{A}$ computes the number of solutions for ψ . As this proves that the returned result is correct, it remains to analyze the running time.

By Lemma 4.1.15, we know that $\text{pw}(\psi) \leq \text{pw}(\phi) + 2 \cdot B \cdot \log|V(\phi)| + \mathcal{O}(1)$ and $|V_2| \in \mathcal{O}(|V(\phi)| \cdot B \cdot \log|V(\phi)|)$. Thus, the algorithm operates in time

$$\begin{aligned} |\phi|^{\mathcal{O}(1)} + (B - \varepsilon)^{\text{pw}(\psi)} \cdot |\psi|^{\mathcal{O}(1)} &\leq (B - \varepsilon)^{\text{pw}(\phi) + 2 \cdot B \cdot \log|V(\phi)| + \mathcal{O}(1)} \cdot |\phi|^{\mathcal{O}(1)} \\ &\leq (B - \varepsilon)^{\text{pw}(\phi)} \cdot |V(\phi)|^{2 \cdot B \cdot \log B} \cdot |\phi|^{\mathcal{O}(1)}. \end{aligned}$$



(a) Some bipartite graph together with a labeling of the vertices.

(b) A reordering of the vertices reveals a half-induced matching of size $h = 3$ where the bold edges are included and the dashed edges forbidden.

Figure 4.2: An illustration of a bipartite graph together with a half-induced matching as defined in Definition 4.2.1.

As the value of B is fixed, the term $|V(\phi)|^{2 \cdot B \cdot \log B}$ can be bounded by a polynomial and thus, is subsumed by the term $|\phi|^{\mathcal{O}(1)}$. Hence, the algorithm operates in time

$$(B - \epsilon)^{\text{pw}(\phi)} \cdot |\phi|^{\mathcal{O}(1)}$$

which contradicts pw-CSPH (resp. pw-#CSPH for the counting version). By Lemma 4.1.10, this contradicts pw-SETH (resp. pw-#SETH) which concludes the proof. $\square$

4.2 Half-Induced Matchings

In the following, we introduce *half-induced matchings* and show relations to compatibility graphs and representative sets. Despite their relation to these techniques, which are used to show improved upper bounds, we use half-induced matching in Chapter 7 to prove lower bounds for variants of GENFAC.

We define half-induced matchings formally in Definition 4.2.1 and refer to Figure 4.2 for an illustration of this definition.

Definition 4.2.1 (Half-Induced Matching). *Consider a bipartite graph $G = (U \dot{\cup} V, E)$. We say that the graph G has a half-induced matching of size h if there are h pairwise different vertices $a_1, \dots, a_h \in U$ and h pairwise different vertices $b_1, \dots, b_h \in V$ such that*

1. *for all $i \in [1 \dots h]$, we have $(a_i, b_i) \in E$ and*
2. *for all $i, j \in [1 \dots h]$ with $j > i$, we have $(a_i, b_j) \notin E$.*

Erdős and Hajnal [44] defined a related concept known as *half-graph* which corresponds to a maximum half-induced matching, that is, all possible edges are present. Fabiański et al. [49] introduce other structures which they refer to as *ladders and semi-ladders*. With their definition, the complement of a half-graph defines a ladder which is again a half-graph but of smaller size and thus, also a maximum half-induced matching. The derived semi-ladder or rather its complement then precisely coincides with our notion of a half-induced matching.

From Sets to Graphs. Usually we do not consider arbitrary graphs in which we want to find a half-induced matching but consider special families of graphs. As the underlying graph we mainly use the compatibility graph as defined in Definition 3.2.11 which we restate here for convenience.

Definition 3.2.11 (Compatibility Graph). *Consider a list $\mathcal{F} = (F_1, \dots, F_k)$ of k sets $F_i \subseteq \mathbb{N}$, for all $i \in [1 \dots k]$, which are considered to be sets of forbidden numbers.*

We define the (infinite) compatibility graph $\mathcal{C}_{\mathcal{F}}$ of $\mathcal{F}$ with

- *vertex set $V(\mathcal{C}_{\mathcal{F}}) = L \dot{\cup} R$ where the two sets L and R are disjoint copies of $\mathbb{N}^k$, and*
- *edge set $E(\mathcal{C}_{\mathcal{F}}) = \{((a_1, \dots, a_k), (b_1, \dots, b_k)) \mid \forall i \in [1 \dots k] : a_i + b_i \notin F_i\}$.*

If we consider only one set $X \subseteq \mathbb{N}$ of forbidden degrees, we simplify notation and write $\mathcal{C}_X$ for the compatibility graph $\mathcal{C}_{(\mathcal{F})}$ where $\mathcal{F} = (X)$.

4.2.1 Relation to Arithmetic Progressions

We first relate the existence of large arithmetic progressions in the set X to the existence of large half-induced matchings in the corresponding compatibility graph.

We first show that arithmetic progressions are sufficient to obtain large half-induced matchings.

Lemma 4.2.2. *Consider a set $X \subseteq \mathbb{N}$. If X contains a maximal arithmetic progression of length h (especially, this sequence cannot be extended by adding larger values), then the compatibility graph $\mathcal{C}_X$ contains a half-induced matching of size $h + 1$.*

Proof. We denote by $a, a+d, a+2 \cdot d, \dots, a+(h-1) \cdot d \in X$ the elements of a maximal arithmetic progression of length h in X where $d \geq 1$ is chosen appropriately. Since this progression is maximal, we know that $a+h \cdot d \notin X$.

We construct the following half-induced matching in $\mathcal{C}_X$ where, for all $i \in [1 \dots h+1]$, we set

$$a_i := (i-1) \cdot d \quad \text{and} \quad b_i := a + (h+1-i) \cdot d.$$

It is straightforward to check these definitions provide a half-induced matching of size $h+1$. For all $i \in [1 \dots h+1]$, we have $a_i + b_i = (i-1) \cdot d + a + (h+1-i) \cdot d = a + h \cdot d \notin X$ and hence, by the definition of the compatibility graph, the vertices a_i and b_i form an edge in the compatibility graph, that is, we have $(a_i, b_i) \in E(\mathcal{C}_X)$.

Similarly, for all $i \in [1 \dots h]$ and all $j \in [1 \dots h+1]$ with $j > i$, we know that $a_i + b_j = (i-1) \cdot d + a + (h+1-j) \cdot d = a + (h+i-j) \cdot d \in X$. Hence, again by the definition of the compatibility graph, this implies that $(a_i, b_j) \notin E(\mathcal{C}_X)$. $\square$

Opposed to Lemma 4.2.2, we also prove that arithmetic progressions are necessary to obtain large half-induced matchings.

Lemma 4.2.3. *Consider a set $X \subseteq \mathbb{N}$ with $|X| = h \geq 2$. Suppose that $\mathcal{C}_X$ contains a half-induced matching of size $h+1$. Then, the set X is an arithmetic progression.*

Proof. We denote the $2(h + 1)$ vertices of the half-induced matching of size $h + 1$ in $\mathcal{C}_X$ by $a_1, \dots, a_{h+1}$ and $b_1, \dots, b_{h+1}$. From the definition of a half-induced matching, we directly conclude that

$$a_1 + b_2 \in X, \quad a_1 + b_3 \in X, \quad \dots, \quad a_1 + b_{h+1} \in X \quad (4.2.1)$$

$$a_2 + b_3 \in X, \quad \dots, \quad a_2 + b_{h+1} \in X. \quad (4.2.2)$$

By the definition of a half-induced matching, all h numbers listed in Equation (4.2.1) are distinct as $b_i \neq b_j$ for all $i \neq j \in [2 \dots h + 1]$. Hence, we immediately get that

$$\{a_1 + b_2, a_1 + b_3, \dots, a_1 + b_{h+1}\} = X.$$

Moreover, we sort the elements in X such that $X = \{x_1, x_2, \dots, x_h\}$ with $x_i < x_{i+1}$. Based on this, we can define a function $\pi: [2 \dots h + 1] \rightarrow [1 \dots h]$ such that $x_{\pi(i)} = a_1 + b_i$ for all $i \in [2 \dots h + 1]$.

For the next step, we consider Equation (4.2.2). We use again the fact that all b_i 's are different and conclude that there must be some index $j \in [1 \dots h]$ such that

$$\{a_2 + b_3, a_2 + b_4, \dots, a_2 + b_{h+1}\} = X \setminus \{x_j\}. \quad (4.2.3)$$

Now we consider two cases. For the first case we assume that $a_2 > a_1$ and set $d := a_2 - a_1$ as the period of the arithmetic sequence. The second case when $a_2 < a_1$ is handled below.

From Equation (4.2.3) we know that, for all $i \in [1 \dots h] \setminus \{j\}$, there is an index $i' \in [3 \dots h + 1]$ such that $x_i = a_2 + b_{i'}$. Thus, by the definition of d , we have that $x_i = a_2 + b_{i'} = a_1 + b_{i'} + d = x_{\pi(i')} + d$. Hence, for every element $x_i \in X \setminus \{x_j\}$ there exists a unique element $x_{i'} \in X$ such that $x_i = x_{i'} + d$.

We also observe that by the ordering of the elements of X (and the assumption that $d > 0$), we get that $x_h + d \notin X$.

With these two observations we can rewrite Equation (4.2.3) such that

$$\{x_1 + d, x_2 + d, x_3 + d, \dots, x_{h-1} + d\} = X \setminus \{x_j\}.$$

Since $d > 0$ and $x_1 = \min X$, we conclude that $x_1 \notin X \setminus \{x_j\}$ and hence, we get that $j = 1$. Moreover, as we assume that the set X has size h , every element (except for x_1) can be represented as a sum of some other element in X and d . Formally, we get from the ordering that $x_{i+1} = x_i + d$ for all $i \in [1 \dots h - 1]$ which shows that X is an arithmetic progression of length h .

It remains to consider the case when $a_1 < a_2$. With the same reasoning we get that in this case $j = h$ and also conclude that X is an arithmetic progression of length h . $\square$

4.2.2 Relation to Representative Sets

In the following we provide a relation between representative sets and half-induced matchings. For this we extend the notion of H -compatibility and H -representation from Definitions 3.2.1 and 3.2.2 to multiple dimensions in the straightforward way. That is, for an integer $k \geq 1$ and a graph H , we extend $\sim_H$ and $\subseteq_{H\text{-rep}}$ to k dimensions, denoted by $\sim_H^k$ and $\subseteq_{H\text{-rep}}^k$, such that the H -compatibility must hold for each of the k dimension.

We show that a large half-induced matching implies large representative sets.

Lemma 4.2.4. Consider a set $X \subseteq \mathbb{N}$, and denote by $\varepsilon > 0$ and $h \geq 2$ two constants.

There exists a constant k depending only on ε and h such that the following holds. Suppose the compatibility graph C_X contains a half-induced matching of size h . Then, there is a set $\mathcal{S} \subseteq \mathbb{N}^k$ such that every representative set $\mathcal{S}' \subseteq_{C_X\text{-rep}}^k \mathcal{S}$ has size $|\mathcal{S}'| \geq (h - \varepsilon)^k$.

Proof. We set the value of k later and proceed with some preliminaries.

Assume that the half-induced matching is between $A, B \subseteq \mathbb{N}$ with $A = \{a_1, \dots, a_h\}$ and $B = \{b_1, \dots, b_h\}$. We partition A^k into $h \cdot k$ different sets $\mathcal{S}_q$ with $q \in [1 \dots h \cdot k]$ such that

$$\mathcal{S}_q = \{(a_{i_1}, \dots, a_{i_k}) \in A^k \mid i_1 + \dots + i_k = q\}.$$

By pigeonhole-principle, there exists some $q' \in [1 \dots h \cdot k]$ such that

$$|\mathcal{S}_{q'}| \geq \frac{h^k}{h \cdot k}.$$

We define $\mathcal{S} := \mathcal{S}_{q'}$ as the set for which we want a lower bound on the size of its representative sets. To simplify notation, we define $q = q'$.

Now, consider some $s \in \mathcal{S}$ where $s = (a_{i_1}, \dots, a_{i_k})$ for appropriately chosen $i_1, \dots, i_k \in [1 \dots h]$. We define the vector $t \in B^k$ as $t = (b_{i_1}, \dots, b_{i_k})$ and prove that s is the unique compatible element for t . By our choice of the vectors s and t , it is clear that $s \sim_{C_X}^k t$.

Now consider some other $s' \in \mathcal{S}$ where $s' = (a_{j_1}, \dots, a_{j_k})$. Since

$$\sum_{p \in [1 \dots h]} i_p = q = \sum_{p \in [1 \dots h]} j_p$$

and $s \neq s'$, there is some index $p \in [1 \dots h]$ such that $i_p > j_p$. By the properties of the half-induced matching, there is no edge between a_{j_p} and b_{i_p} and hence, s' is not compatible with t . Thus, s is the only compatible partner of t and forced to belong to any representative set $\mathcal{S}' \subseteq_{C_X\text{-rep}}^k \mathcal{S}$.

Since the above argument holds for all $s \in \mathcal{S}$, we conclude that the only representative set for $\mathcal{S}$ is itself.

It remains to set k . We choose k large enough such that $k \log(h - \varepsilon) \leq k \log(h) - \log(h \cdot k)$. Then, we have

$$|\mathcal{S}| \geq \frac{h^k}{h \cdot k} \geq (h - \varepsilon)^k$$

which concludes the proof. $\square$

We conjecture that the converse of Lemma 4.2.4 is also true, that is, if the largest half-induced matching is small, then we can also compute a small representative set. For example, for $X = \{10, 100, 1000, \dots\}$ the largest half-induced matching in C_X is of size three, a constant, (even though X itself is infinite). Intuitively, the size of the representative set itself must be small because knowing any *two* forbidden degrees of a vertex in the future solution is enough for us to deduce the degree of the vertex in the partial solution.

Conjecture 4.2.5. *Consider a set $X \subseteq \mathbb{N}$ and an integer constant $h \geq 2$.*

There exists a constant k depending only on h such that the following holds. Suppose the largest half-induced matching in $\mathcal{C}_X$ has size h . Then, every $\mathcal{S} \subseteq \mathbb{N}^k$ has a representative set $\mathcal{S}' \subseteq_{\mathcal{C}_X\text{-rep}}^k \mathcal{S}$ with $|\mathcal{S}'| \leq h^{k+o(k)} \cdot \text{poly}(\max X, |X|)$.

We explain the consequences of Conjecture 4.2.5 in Chapter 5.

Part II

General Factor

Introduction To General Factor

In this part of the thesis we consider the GENERAL FACTOR problem which is a generalization of the MATCHING problem and the f -FACTOR problems. Formally, we define the problem as follows.

Definition 5.1.1 (GENERAL FACTOR (GENFAC)). *Let $G = (V, E)$ be an undirected vertex labeled graph where the label of a vertex v is a set $\Lambda(v) \subseteq \mathbb{N}$. We say that a set of edges $S \subseteq E$ is a solution if $\deg_S(v) \in \Lambda(v)$ for all $v \in V$.*

GENFAC is the problem of deciding whether G has a solution.

While GENFAC can be defined for all (computable) sets, we focus in the following on the case where the sets are finite or cofinite. Moreover, we define the two problems Λ -FACTOR and X -ANTI-FACTOR where all sets are identical and they are either all finite or all cofinite.

Definition 5.1.2 (Variants of GENFAC). *We define the following three variants of GENFAC.*

- *For a fixed set $\Lambda \subseteq \mathbb{N}$, we define Λ -FACTOR as the variant of GENFAC where for the input graph G , all vertices $v \in V(G)$ satisfy $\Lambda(v) = \Lambda$.*
- *For a fixed set $X \subseteq \mathbb{N}$, we define X -ANTI-FACTOR as the variant of GENFAC where for the input graph G , all vertices $v \in V(G)$ satisfy $\Lambda(v) = \mathbb{N} \setminus X$, that is, the set X specifies the complement of the set of allowed degrees.*
- *For a fixed integer $x \in \mathbb{N}$, we define ANTI-FACTOR_x as variant of GENFAC where all vertices $v \in V(G)$ are assigned a set such that $|\mathbb{N} \setminus \Lambda(v)| \leq x$.*

Phrased differently, for Λ -FACTOR the set Λ specifies the *allowed* degrees. For X -ANTI-FACTOR the set X specifies the forbidden or *excluded* degrees. For ANTI-FACTOR_x the number x specifies *how many* degrees are forbidden.

Based on these definitions, the problems Λ -FACTOR and X -ANTI-FACTOR are dual to each other. Indeed, Λ -FACTOR is identical to $(\mathbb{N} \setminus \Lambda)$ -ANTI-FACTOR, and X -ANTI-FACTOR is identical to $(\mathbb{N} \setminus X)$ -FACTOR. This allows us to switch between both problems seamlessly. Nevertheless, if we consider the problems separately, we usually choose the problem which fits the current definition better, that is, we choose the variant where the specified set is *finite*. More formally, when the set of allowed degrees is finite, we consider Λ -FACTOR, and when the set of allowed degrees is cofinite, we usually consider X -ANTI-FACTOR as in this case the set X is finite.

We naturally lift the definition of GENFAC and its variants mentioned in Definition 5.1.2 to the respective counting problems $\#\text{GENFAC}$, Λ - $\#\text{FACTOR}$, X - $\#\text{ANTI-FACTOR}$, and $\#\text{ANTI-FACTOR}_x$ where we want to *count the number of solutions* instead of just deciding if there is a solution.

Our Results. We study the three versions of GENFAC for the decision and counting version and prove an algorithmic upper bound and a lower bound based on pw-SETH for the decision problem and based on pw-#SETH for the counting problem (see Conjectures 4.1.4 and 4.1.5 for further details) for each version. Unfortunately, the lower bounds are not matching the upper bounds in certain cases for the decision version.

We first state the formal results and then analyze the tightness of the bounds.

Λ -FACTOR. The problem Λ -FACTOR with finite set Λ serves as an easy and well-structured starting point for the upper and lower bounds.

Intuitively, in the settings where Λ is finite, each vertex can have at most $\max \Lambda + 1$ “states” which are determined by the number of selected edges each vertex might be incident to in a solution. We show that the algorithms reflect this intuition and the bound captures the complexity of the problem.

Theorem 5.1.3. *Fix a non-empty and finite set $\Lambda \subseteq \mathbb{N}$. Assuming we are given a graph G and a tree decomposition of width w , the following holds.*

- We can count the number of solutions of size s for Λ -FACTOR in time $(\max \Lambda + 1)^w \cdot |G|^{\mathcal{O}(1)}$.
- If $0 \notin \Lambda$ and $\max\text{-gap } \Lambda \geq 2$, then no algorithm can solve Λ -FACTOR in time $(\max \Lambda + 1 - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$ for any $\varepsilon > 0$, unless pw-SETH fails.
- If $\Lambda \neq \{0\}$, then no algorithm can solve Λ -#FACTOR in time $(\max \Lambda + 1 - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$ for any $\varepsilon > 0$, unless pw-#SETH fails.

When we focus on the higher-level perspective, we can see that the lower bounds match the upper bounds and thus, our algorithm is optimal.

On the more detailed level, we can see that few cases are excluded from the lower bounds—and there is good reason why they should be. An empty set Λ results in a trivial no-instance, as no solution is possible. Similarly, if $0 \in \Lambda$, we obtain a trivial yes-instance for the decision version as selecting no edges gives a trivial solution. For the counting version, we additionally need that $\Lambda = \{0\}$ for the trivial cases as otherwise there might be more solutions to count.

It remains to address the restriction on the maximum gap of Λ , that is, the requirement that $\max\text{-gap } \Lambda \geq 2$. Indeed, Cornuéjols showed that when $\max\text{-gap } \Lambda \leq 1$, then Λ -FACTOR can be solved in polynomial-time [30].

Theorem 5.1.4 ([30, Section 3]). *The problem GENFAC can be solved in polynomial-time on arbitrary graphs if all vertices $v \in V(G)$ satisfy $\max\text{-gap } \Lambda(v) \leq 1$.*

Hence, it is natural to omit this case as well from the lower bound for the decision version.

We conclude that Theorems 5.1.3 and 5.1.4 together provide a complete classification of the hardness of Λ -FACTOR for the decision and counting version when the set Λ is finite.

We note that Dudycz and Paluch [41] recently extended the result by Cornuéjols [30] to the maximization and minimization version (see also [88]).

Surprisingly, the hardness of the *exact* decision version for Λ -FACTOR remains open even for the special case of the classical MATCHING problem [85, 100, 101]. We note that our algorithm

A lower bound for the maximization version of Λ -FACTOR similar to Theorem 5.1.3 is not included in this thesis though, it was proven in [103, 104].

solves this variant of the problem, but does not operate in polynomial-time for those cases where the decision version is polynomial-time solvable. The fact that the exact matching problem is still open is quite surprising as Mulmuley, Vazirani, and Vazirani [109] provide a *randomized* polynomial-time algorithm for the problem but no *deterministic* polynomial-time algorithm is known yet which makes it one of the few problems where currently no efficient derandomization is known.

X-ANTI FACTOR. When we consider cofinite sets of allowed degrees, that is, when we consider X-ANTI FACTOR, we prove a similar set of results as for Λ -FACTOR. Although the algorithmic complexity is similar to Λ -FACTOR, the precise complexity for the decision version is less clear. While the upper bounds solely depends on $\max X$, the lower bound makes use of half-induced matching (see Section 4.2 for an introduction to this concept). For the counting version, no such requirements are needed.

Theorem 5.1.5. *Fix a non-empty and finite set $X \subseteq \mathbb{N}$. Assuming we are given a graph G and a tree decomposition of width w , the following holds.*

- *We can count the number of solutions of size s for X-ANTI FACTOR in time $(\max X + 2)^w \cdot |G|^{\mathcal{O}(1)}$.*
- *If $0 \in X$ and $\max\text{-gap}(\mathbb{N} \setminus X) \geq 2$ such that X contains a half-induced matching of size h , then no algorithm can solve X-ANTI FACTOR in time $(h - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$ for any $\varepsilon > 0$, unless pw-SETH fails.*
- *If $X \neq \{0\}$, then no algorithm can solve X-#ANTI FACTOR in time $(\max X + 2 - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$ for any $\varepsilon > 0$, unless pw-#SETH fails.*

Again, we first observe that the restrictions of the set X in Theorem 5.1.5 are natural. The algorithmic result by Cornuéjols [30] from Theorem 5.1.4 does not require the sets to be finite and thus, also works for X-ANTI FACTOR when $\max\text{-gap}(\mathbb{N} \setminus X) \leq 1$. Similarly, if we drop the requirement that $0 \in X$, that is, if we allow 0 as a valid degree, the decision version is trivial.

For the counting version, we observe that the excluded case $X = \{0\}$ is *non-trivial*. Indeed, $\{0\}$ -#ANTI FACTOR precisely corresponds to COUNTING EDGE COVERS. We use a separate reduction to show that our algorithm is optimal in this case. See Theorem 7.9.6 in Section 7.9 for further details.

Before we consider the last variant ANTI FACTOR_X of GENFAC, we briefly comment on the gap between the upper bound and the lower bound for the decision version. At a first glance, and in light of the lower bound for the counting version, it seems reasonable that the lower bound can be strengthened to match the upper bound. However, the fact that the lower bound (for the decision version) is based on the half-induced matchings indicates that the complexity of the problem depends in a much more subtle way on the structure of the set X than possibly expected.

This strange behavior of GENFAC comes to an extreme when we consider the last variant of the problem. There we also obtain a more informative lower bound for the decision version.

ANTI FACTOR_x . In contrast to $X\text{-ANTI}\text{FACTOR}$, we can improve the algorithm and obtain almost matching upper and lower bounds for $\text{ANTI}\text{FACTOR}_x$. More precisely, the running time now only depends on the number of forbidden degrees.

Intuitively the idea of the algorithm is as follows. When only few degrees are forbidden (i.e., x in our case), it suffices to store only few partial solutions (i.e., $x + 1$ in our case) such that whenever a vertex might receive a forbidden degree, there is one partial solution such that this vertex does not obtain a forbidden degree. We formalize these concepts with representative sets as introduced in Section 3.2.

As the notion of representative sets is highly incompatible with the counting problem since we ignore certain solution, the algorithmic result does not apply for $\#\text{ANTI}\text{FACTOR}_x$. Instead, we show that this is not a shortcoming of our approach but an essential property of $\#\text{ANTI}\text{FACTOR}_x$.

Theorem 5.1.6. *Fix an integer $x \geq 1$. Assuming we are given a graph G and a tree decomposition of width w , the following holds.*

- We can solve $\text{ANTI}\text{FACTOR}_x$ in time $(x + 1)^{\mathcal{O}(w)} \cdot |G|^{\mathcal{O}(1)}$.
- If $x \geq 2$, then no algorithm can solve $\text{ANTI}\text{FACTOR}_x$ in time $(x + 1 - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$ for any $\varepsilon > 0$, unless pw-SETH fails.
- Even for $x = 1$ no algorithm can solve $\#\text{ANTI}\text{FACTOR}_x$ in time $|V(G)|^{w-\varepsilon}$ for any $\varepsilon > 0$, unless #SETH fails.

Before we discuss the implications of these results, we first show that we cover all relevant cases. For the case of $x = 0$ no degree is forbidden and there is always a solution. When $x = 1$, we can again use the result by Cornuéjols [30] as, for each vertex v , the maximum gap of $\Lambda(v)$ is at most 1.

We first observe that the counting version is significantly harder than the decision version. Even when for each vertex only one degree is forbidden, the problem does not admit an FPT-algorithm but is $\#W[2]$ -hard (see Main Theorem II.6 for the formal statement). This result adds $\#\text{ANTI}\text{FACTOR}_x$ to the (comparably) short list of problems that are not FPT parameterized by treewidth (or pathwidth).

For the decision version we observe that the upper bound and the lower bound are almost matching. The bottleneck to improve the algorithm such that we obtain the bound from the lower bound is two-fold.

- The algorithmic result from Theorem 5.1.6 relies on the computation of representative sets. For each node of the tree decomposition, we store a representative set of size $(x+1)^w$ assuming the bag has size w .

While this bound coincides with the lower bound, the computation of these sets is not fully understood yet. For example, one of the procedures from Section 3.2 relies on matrix multiplication, and thus, introduces a factor of ω in the exponent of the running time. In general, a better procedure to compute representative sets, which is in the best case just linear in the input size, would directly yield a faster algorithm.

When we parameterize by the pathwidth, such an improvement would already provide an optimal algorithm.

- For the parameterization by treewidth, we additionally have to handle the join nodes. Currently, the only known procedure for these nodes (in the context of representative sets) is to combine them in the naive way and then compute the representative set thereof.

We note that for classical algorithms based on tree decompositions, that is, algorithms that just compute and store the states relating to partial solutions, the situation was similar. For example, the first algorithms for DOMINATING SET were not optimal, but the optimal algorithm improved on this by using fast convolution techniques to handle the join nodes [132].

In light of these two obstacles, we conjecture that the lower bound for ANTIFACTOR_x is optimal, and thus, the algorithm can be improved only slightly to close the existing gap.

Conjecture 5.1.7. *Fix an integer $x \geq 1$. Assuming we are given a graph G and a tree decomposition of width w , we can solve ANTIFACTOR_x in time $(x + 1)^w \cdot |G|^{O(1)}$.*

Before we give the formal proofs of Theorems 5.1.3, 5.1.5 and 5.1.6, we first discuss one more relation between representative sets and half-induced matchings.

Relation to Half-Induced Matchings. As mentioned above, the running time of the algorithm for ANTIFACTOR_x from Theorem 5.1.6 depends on the size x of the list of *forbidden degrees*, which determines the size of the half-induced matching, but the lower bound in Theorem 5.1.5 depends on the size h of the *half-induced matching*.

From Lemma 4.2.4 we know that the existence of a half-induced matching of size h in the set X implies a lower bound on the size of the representative set.

Lemma 4.2.4. *Consider a set $X \subseteq \mathbb{N}$, and denote by $\varepsilon > 0$ and $h \geq 2$ two constants.*

There exists a constant k depending only on ε and h such that the following holds. Suppose the compatibility graph C_X contains a half-induced matching of size h . Then, there is a set $S \subseteq \mathbb{N}^k$ such that every representative set $S' \subseteq_{C_X\text{-rep}}^k S$ has size $|S'| \geq (h - \varepsilon)^k$.

The running time of the algorithm for ANTIFACTOR_x from Theorem 5.1.6 depends on the size of the representative sets computed. Moreover, this algorithm even works if we only consider a single set X and thus, transfers to $X\text{-ANTIFACTOR}$ with the same running time when $x = |X|$. Therefore, Lemma 4.2.4 implies that any algorithm which uses representative sets similarly takes time at least $(h - \varepsilon)^w$. This can be seen as an *unconditional* version of the lower bound shown in Theorem 5.1.5.

In contrast to this lower bound, we propose in Conjecture 4.2.5 that the size of half-induced matching provides an *upper bound* on the size of the representative sets as well.

Conjecture 4.2.5. *Consider a set $X \subseteq \mathbb{N}$ and an integer constant $h \geq 2$.*

There exists a constant k depending only on h such that the following holds. Suppose the largest half-induced matching in C_X has size h . Then, every $S \subseteq \mathbb{N}^k$ has a representative set $S' \subseteq_{C_X\text{-rep}}^k S$ with $|S'| \leq h^{k+o(k)} \cdot \text{poly}(\max X, |X|)$.

With Conjecture 4.2.5 it seems reasonable to obtain algorithms for X -ANTIFACTOR that are based on representative sets and thus, have a running time depending only on the size h of the half-induced matching. Such an algorithm would complement the lower bounds for the decision version and identify the size of the half-induced matching as the actual measure of the complexity of X -ANTIFACTOR.

Outline of the Proofs. Although, the upper and lower bounds differ for the individual problems, the conceptual ideas are similar and thus, we can simplify and unify several of the constructions.

Algorithmic Results. More precisely, we prove two algorithmic results in Chapter 6.

Main Theorem II.1. *Fix some set $\Lambda \subseteq \mathbb{N}$ that is finite or cofinite.*

Given a Λ -FACTOR instance G along with a tree decomposition of width w , we can count the solutions of size exactly s in time $(\text{top } \Lambda + 1)^w \cdot |G|^{\mathcal{O}(1)}$ for all s simultaneously.

For the second algorithm, we focus on ANTIFACTOR_x and exploit that only few numbers are forbidden.

Main Theorem II.2. *Fix some integer $x \geq 1$. Given an ANTIFACTOR_x instance G along with a tree decomposition of width w , we can decide if there is a solution of size exactly s in time $(x + 1)^{\mathcal{O}(w)} \cdot |G|^{\mathcal{O}(1)}$ for all s simultaneously.*

Lower Bounds. For the lower bound results in Chapter 7 the situation is more complicated. We start with the result for the decision version of Λ -FACTOR.

Main Theorem II.3. *Fix some non-empty set $\Lambda \subseteq \mathbb{N}$ that is finite or cofinite with $0 \notin \Lambda$ and $\max\text{-gap } \Lambda > 1$. If Λ is finite, we define $h := \max \Lambda + 1$, and if Λ is cofinite, then we define h as the size of the largest half-induced matching in the compatibility graph of $\mathbb{N} \setminus \Lambda$.*

If there is an $\varepsilon > 0$ and an algorithm that solves Λ -FACTOR on a given graph G with a given path decomposition of width w in time $(h - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-SETH is false.

While this result leaves a gap for the case of cofinite sets Λ , we close this gap for the counting version by proving a tight lower bound.

Main Theorem II.5 (Lower Bound for Counting Version). *Fix a non-empty set $\Lambda \subseteq \mathbb{N}$ that is finite with $\Lambda \neq \{0\}$ or cofinite with $\Lambda \neq \mathbb{N}$ and $\Lambda \neq \mathbb{N}_{\geq 1}$.*

If there is an $\varepsilon > 0$ and an algorithm that solves Λ -#FACTOR on a given graph G with a given path decomposition of width w in time $(\text{top } \Lambda + 1 - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-#SETH is false.

When we consider the problem ANTIFACTOR_x , we can extend the lower bound from Main Theorem II.3 into a more informative bound.

Main Theorem II.4. *Let $x \geq 2$ denote a fixed integer.*

If there is an $\varepsilon > 0$ and an algorithm that solves ANTIFACTOR_x on a given graph G with a given path decomposition of width w in time $(x + 1 - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-SETH is false.

For the counting version, the improved algorithm does not apply. We prove that this is not a shortcoming of our approach but inherent to the problem.

Main Theorem II.6 (Lower Bound for Counting Version II). *There exists a constant $c > 0$ such that if there is an algorithm that solves $\#ANTI\text{FACTOR}_1$ on a given graph G with n vertices with a given path decomposition of width p in time $\mathcal{O}(n^{p-c})$, then $\#SETH$ is false.*

Moreover, $\#ANTI\text{FACTOR}_1$ is $\#W[2]$ -hard parameterized by pathwidth.

Then, the proofs of Theorems 5.1.3 and 5.1.5 follow from Main Theorems II.1, II.3 and II.5, and the proof of Theorem 5.1.6 follows from Main Theorems II.2, II.4 and II.6.

Solving General Factor

In the following we provide algorithms parameterized by treewidth for the variants of GENFAC. More precisely, we consider the three variants Λ -FACTOR, X -ANTIFACTOR, and ANTIFACTOR _{X} and prove the algorithmic results from Main Theorems II.1 and II.2.

Instead of analyzing Λ -FACTOR and X -ANTIFACTOR separately, our first algorithm solves the variant of GENFAC where each vertex of the graph is assigned either a finite set or a cofinite set and all these sets might be different.

Intuitively the running time should depend on the largest element of each of the sets. As we are also dealing with cofinite sets of allowed degrees, this notion is not well-defined. Instead, we introduce some notation which, for the case of cofinite sets, return a value that can be treated as an equivalent to the largest element.

Definition 2.1.3 (The Top Element). *Consider a set $\tau \subseteq \mathbb{N}$ that is either finite or cofinite. Based on the set τ , we define the number $\text{top } \tau \in \tau$ as*

$$\text{top } \tau := \begin{cases} \max \tau & \text{if } \tau \text{ is finite,} \\ 1 + \max(\mathbb{Z} \setminus \tau) & \text{if } \tau \text{ is cofinite.} \end{cases}$$

With this useful definition, we can now state the main result for GENFAC. Henceforth, we assume that the reader is familiar with the notation related to tree decompositions from Section 2.2.

Main Lemma 6.1.1 (Algorithm for GENFAC). *Assume we are given an instance G of GENFAC where the set assigned to each vertex is either finite or cofinite. Moreover, assume we are given a tree decomposition (T, β) of the graph G of width w .*

We define

$$\eta := \max_{t \in V(T)} \prod_{v \in \beta(t)} 1 + \text{top}(\Lambda(v)) \quad \text{and} \quad \bar{H} := \max_{t \in V(T)} \prod_{h \in \{\text{top}(\Lambda(v)) \mid v \in \beta(t)\}} h.$$

Then, for all s , we can count the solutions of size exactly s in time $\eta \cdot \bar{H} \cdot |G|^{\mathcal{O}(1)} + 2^w \cdot |G|^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}(\bar{H})}$.

From this general algorithm, we directly obtain Main Theorem II.1 by using that all sets are equal.

For Λ -FACTOR we use that the set Λ is finite and thus, we get $\text{top}(\Lambda) = \max \Lambda$ which implies $\eta = (\max \Lambda + 1)^{w+1}$. Similarly, for X -ANTIFACTOR, we use that the problem is equivalent to $(\mathbb{N} \setminus X)$ -FACTOR and that $\text{top}(\mathbb{N} \setminus X) = \max X + 1$ which gives $\eta = (\max X + 2)^{w+1}$.

Main Theorem II.1. *Fix some set $\Lambda \subseteq \mathbb{N}$ that is finite or cofinite.*

Given a Λ -FACTOR instance G along with a tree decomposition of width w , we can count the solutions of size exactly s in time $(\text{top } \Lambda + 1)^w \cdot |G|^{\mathcal{O}(1)}$ for all s simultaneously.

Proof. By the problem definition, all vertices of the given Λ -FACTOR instance are assigned the same finite set Λ . Hence, from Main Lemma 6.1.1, we immediately get that $\eta = (\text{top } \Lambda + 1)^{w+1}$. Moreover, we have that $\bar{H} = \text{top}(\Lambda) = \text{top } \Lambda$.

Since the set Λ is fixed, the factor $2^{\text{top } \Lambda}$ contributes only a constant. As we can assume that $\text{top } \Lambda \geq 1$, the first part of the sum dominates the second part, which concludes the proof. Indeed, the case when $\text{top } \Lambda = 0$ is trivial. If $\Lambda = \{0\}$, then only the empty solution is allowed. Otherwise, Λ is cofinite and $\Lambda = \mathbb{N}$ which implies that every subset is a solution, that is there are $\binom{|E|}{s}$ solutions of size s . $\square$

The Size of the List. The running time of our algorithm from Main Lemma 6.1.1 is based on the maximum degree of the given sets. In contrast to this, our next result is not based on the maximum degree but on the number of forbidden degrees. That is, we focus on cofinite sets and consider the problem ANTIFACTOR_x where x is some (potentially small) constant. In this setting it is possible to achieve a significantly faster algorithm when very few degrees are forbidden. However, this faster algorithm does not solve the counting version of the problem but considers the decision version for a certain solution size.

Main Theorem II.2. *Fix some integer $x \geq 1$. Given an ANTIFACTOR_x instance G along with a tree decomposition of width w , we can decide if there is a solution of size exactly s in time $(x+1)^{\mathcal{O}(w)} \cdot |G|^{\mathcal{O}(1)}$ for all s simultaneously.*

Outline. We first present the algorithm for GENFAC in Section 6.1, that is, we prove Main Lemma 6.1.1. Afterward in Section 6.2, we present the algorithm for ANTIFACTOR_x , that is, we prove Main Theorem II.2.

6.1 Parameterizing by the Largest Degree

In this section we present the algorithm for GENFAC where all sets are finite or cofinite parameterized by treewidth and the largest reasonable degree of a vertex. We later complement this algorithm by a lower bound for Λ -FACTOR, which shows that this algorithm is essentially optimal under pw-SETH, even when all sets appearing in the graph are equal to one finite set of degrees.

Formally, we prove Main Lemma 6.1.1.

Main Lemma 6.1.1 (Algorithm for GENFAC). *Assume we are given an instance G of GENFAC where the set assigned to each vertex is either finite or cofinite. Moreover, assume we are given a tree decomposition (T, β) of the graph G of width w .*

We define

$$\eta := \max_{t \in V(T)} \prod_{v \in \beta(t)} 1 + \text{top}(\Lambda(v)) \quad \text{and} \quad \bar{H} := \max_{t \in V(T)} \prod_{h \in \{\text{top}(\Lambda(v)) \mid v \in \beta(t)\}} h.$$

Then, for all s , we can count the solutions of size exactly s in time $\eta \cdot \bar{H} \cdot |G|^{\mathcal{O}(1)} + 2^w \cdot |G|^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}(\bar{H})}$.

Arulselvan et al. implicitly showed that Λ -FACTOR (and also GENFAC for finite sets) can be solved in time $(\max \Lambda + 1)^{3\text{tw}} \cdot n^{\mathcal{O}(1)}$ [6]. We use a similar approach and combine it with a more precise analysis of the running time and fast convolution techniques to prove Main Lemma 6.1.1. Moreover, we extend the result in a way such that all vertices can have different sets.

The basic idea of the algorithm is a dynamic program which operates on the tree decomposition of the given graph. For each node t of the decomposition, the algorithm stores all possibilities in which a (partial) solution could interact with the graph corresponding to the node t . Depending on the type of the node (i.e., forget, introduce vertex, introduce edge, join), the algorithm computes the table entries based on the values for the children.

Notation. To formally state our algorithm, we first introduce some notation. Recall from Definition 5.1.1, that we denote by $\Lambda(v) \subseteq \mathbb{N}$ the set of allowed degrees associated with vertex $v \in V(G)$. We can clearly assume that $\Lambda(v) \subseteq [0 \dots |E(G)|]$ for every vertex v with a finite set $\Lambda(v)$ since no vertex can have more neighbors. Moreover, for the cofinite vertices we define a set $X(v) := \mathbb{N} \setminus \Lambda(v)$ which contains the forbidden degrees. Here we can also assume that $X(v) \subseteq [0 \dots |E(G)|]$ since a larger degree is not possible anyway.

To simplify notation, we write $\text{top}(v) := \text{top}(\Lambda(v))$ in the following whenever v denotes a vertex.

Recall that when given a node t of a tree decomposition, we denote by $\beta(t)$ the bag of t . Moreover, we denote by V_t the vertices introduced at the subtree rooted at t , and by E_t the edges introduced in the subtree rooted at t .

The basic idea is to associate with each partial solution a vector $x \in \mathbb{N}^{\beta(t)}$ such that (1) x represents the degree of the vertices in the bag in this partial solution and (2) all vertices below $\beta(t)$, i.e., the vertices in $V_t \setminus \beta(t)$, already have a valid number of incident edges in the solution.

Then, the idea is to store for each such function the number of partial solutions corresponding to this function. However, naively doing so results in a running time of $n^{\mathcal{O}(\text{tw})}$ which is far too slow. Also, it does not suffice to simply limit the function to $\max \Lambda(v)$ as this might not even be defined for a vertex v with a cofinite set.

In order to achieve a finite number of states for each vertex, we exploit that a vertex v with a cofinite set does not distinguish between a degree of $\max X(v) + 1$ and a degree of $\max X(v) + c$ for any other $c \geq 1$.

For the vertices with cofinite set, we use the notion of truncated addition from Definition 3.1.1 with the special symbol $\top$. Recall that the truncated addition $\oplus_{\text{top}(v)-1}$ is the extension of the standard addition where, for all $a, b \in [0 \dots \text{top}(v)] \cup \{\top\}$, we set

$$a \oplus_{\text{top}(v)-1} b := \begin{cases} \top & \text{if } \top \in \{a, b\} \vee a + b \geq \text{top}(v), \\ a + b & \text{otherwise.} \end{cases}$$

To simplify notation, we omit the value $\text{top}(v) - 1$ as subscript when it is clear from the context and extend the notion of $\oplus$ to vectors.

As a last ingredient, we define, for each vertex $v \in V(G)$, a set $\mathbb{A}_v$ of all possible states the vertex v could have. Formally, if the set $\Lambda(v)$ is finite, then we set $\mathbb{A}_v := [0 \dots \text{top}(v)]$ and otherwise, that is, if the set $\Lambda(v)$ is cofinite, then we set $\mathbb{A}_v := [0 \dots \text{top}(v)] \cup \{\top\}$.

Outline of the Algorithm. Now we can formally describe the definition of the dynamic program which we use to solve GENFAC.

As for other algorithms parameterized by treewidth, we also assume that a tree decomposition is given as input. This is reasonable as it allows us to study the problem independently of finding an optimal decomposition. Moreover, we assume that the given tree decomposition is *nice* with *introduce edge nodes*. Both properties can be easily achieved without increasing treewidth and the number of bags too much by adjusting a given tree decomposition appropriately (see Section 2.2 for further details).

The dynamic program fills a table NumbSol such that

1. for all nodes t of the tree decomposition,
2. for all state-vectors $x \in \mathbb{A}(t)$ where $\mathbb{A}(t) := \bigtimes_{v \in \beta(t)} \mathbb{A}_v$ and the positions for vectors in $\mathbb{A}(t)$ are indexed by vertices from $\beta(t)$ for notational convenience, and
3. for all possible size bounds $\kappa \in [0 \dots m]$ where $m := |E(G)|$ is the number of edges of G ,

it holds that $\text{NumbSol}[t, x, \kappa] = a$ if and only if there are a pair-wise different partial solutions $S \subseteq E_t$ with $|S| = \kappa$ such that

1. for all $v \in V_t \setminus \beta(t)$, the set S satisfies $\deg_S(v) \in \Lambda(v)$, and
2. for all $v \in \beta(t)$, the set S satisfies $\deg_S(v) = x[v]$ if $x(v) \neq \top$ and $\deg_S(v) \geq \text{top}(v)$ otherwise.

As the bag of the root r is empty, only the empty assignment is valid and hence, the value of $\text{NumbSol}[r, \emptyset, \kappa]$ corresponds to the number of solutions of size exactly κ .

The computation of the table entries is mostly straightforward. To achieve a fast convolution at the join nodes, we use the convolution techniques as introduced in Section 3.1. Formally, we use Main Theorem I.1.

Main Theorem I.1 (General Convolution). *Consider a domain $\mathbb{D}$ composed of the four subdomains $\mathbb{S}$, $\mathbb{A}$, $\mathbb{L}$, and $\mathbb{T}$ such that $\mathbb{D} := \mathbb{S} \times \mathbb{A} \times \mathbb{L} \times \mathbb{T}$*

- *with $\mathbb{S} = \mathbb{Z}_{z_1} \times \dots \times \mathbb{Z}_{z_\sigma}$ for the first domain,*
- *with $\mathbb{A} = [0 \dots q_1] \times \dots \times [0 \dots q_\alpha]$ for the second domain and define $A_+ := \sum_{i=1}^\alpha q_i$,*
- *with $\mathbb{L} = [0 \dots x_1] \times \dots \times [0 \dots x_\lambda]$ for the third domain, and*
- *with $\mathbb{T} = [0 \dots y_1] \cup \{\top\} \times \dots \times [0 \dots y_\tau] \cup \{\top\}$ for the fourth domain and define $T_+ := \sum_{i=1}^\tau y_i$.*

Denote by $f_1, f_2: \mathbb{D} \rightarrow \mathbb{N}$ two functions that are given as table representations.

We denote by $f: \mathbb{D} \rightarrow \mathbb{N}$ the convolution of f_1 and f_2 that satisfies, for all $(s, a, \ell, t) \in \mathbb{S} \times \mathbb{A} \times \mathbb{L} \times \mathbb{T} = \mathbb{D}$,

$$f(s, a, \ell, t) = \sum_{s_1+s_2=s} \sum_{a_1+a_2=a} \sum_{\ell_1+\ell_2=\ell} \sum_{t_1 \oplus t_2=t} f_1(s_1, a_1, \ell_1, t_1) \cdot f_2(s_2, a_2, \ell_2, t_2).$$

If U denotes a non-negative integer such that all images of f_1 , f_2 and f are contained in $[0 \dots U]$, then the function f can be computed in time

$$X \cdot \log X \cdot (\log U + \overline{M})^{\mathcal{O}(1)} + 2^\tau \cdot (\log U + \exp(\overline{M}))^{\mathcal{O}(1)}$$

with $X = |\mathbb{D}| \cdot |\mathbb{L}|^{\log(3/2)} \cdot (A_+ + T_+)^{\log(3)} \cdot 3^\lambda$ and $\overline{M} = \prod_{m \in M} m$ where $M = \{z_1, \dots, z_\sigma, q_1, \dots, q_\alpha, 3, y_1, \dots, y_\tau, 3\}$.

In the following we use Main Theorem 1.1 with $\sigma = \lambda = 0$, that is, we only use the domains corresponding to addition and truncated addition.

With this result we have everything ready to state our first algorithm.

Main Lemma 6.1.1 (Algorithm for GENFAC). *Assume we are given an instance G of GENFAC where the set assigned to each vertex is either finite or cofinite. Moreover, assume we are given a tree decomposition (T, β) of the graph G of width w .*

We define

$$\eta := \max_{t \in V(T)} \prod_{v \in \beta(t)} 1 + \text{top}(\Lambda(v)) \quad \text{and} \quad \overline{H} := \max_{t \in V(T)} \prod_{h \in \{\text{top}(\Lambda(v)) \mid v \in \beta(t)\}} h.$$

Then, for all s , we can count the solutions of size exactly s in time $\eta \cdot \overline{H} \cdot |G|^{\mathcal{O}(1)} + 2^w \cdot |G|^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}(\overline{H})}$.

Proof. As previously mentioned, the algorithm is essentially a dynamic program on the given tree decomposition. Then, the computation at every node t is determined by the type of the node t .

Formally, as a first step, the algorithm initializes the table NumbSol such that, for all nodes $t \in V(T)$, for all state-vectors $x \in \mathbb{A}(t)$, and for all integers $\kappa \in [0 \dots m]$, the corresponding table entry is set to 0, that is, we set $\text{NumbSol}[t, x, \kappa] := 0$.

As the second step, the algorithm traverses the given tree decomposition (which is, without loss of generality, assumed to be nice and has introduced edge nodes) in post-order and performs one of the following five operation based on the type of the current node t .

Leaf Node: The node t is a leaf node and thus, has an empty bag, that is, $\beta(t) = \emptyset$.

For the empty state-vector and the solution size of 0, we set $\text{NumbSol}[t, \emptyset, 0] := 1$. All other entries remain 0 as defined by the initialization.

Introduce Vertex Node: Assume that the node t' is the unique child of the current node t and denote by v the vertex that is introduced at the node t , that is, $\beta(t) = \beta(t') \cup \{v\}$.

We define the table entries to be

$$\text{NumbSol}[t, x, \kappa] := \begin{cases} \text{NumbSol}[t', x[\beta(t')], \kappa] & \text{if } x[v] = 0 \text{ or } \mathbb{A}_v = \{\top\}, \\ 0 & \text{otherwise.} \end{cases}$$

Introduce Edge Node: Assume that the node t' is the unique child of the current node t and denote by $\{u, v\}$ the (undirected) edge that is introduced at the node t .

We have to decide whether to pick the edge $\{u, v\}$ or not. Since choosing the edge results in a non-zero degree, the edge must not have been chosen when the degree is zero. As the edge can only be chosen if the degree of the vertices u and v is not 0 after the edge was added, we consider two cases.

If $x[v] = 0$ or $x[u] = 0$, then we define

$$\text{NumbSol}[t, x, \kappa] := \text{NumbSol}[t', x, \kappa].$$

Otherwise, we know that both values are not 0. However, because of the cofinite sets, they still might not be an integer but the symbol $\top$. In this latter case we can choose to decrease the value to $\text{top}(v) - 1$ or to keep it as $\top$. Therefore, with the definition of the operator $\sqcup$ to combine two vectors from Definition 2.1.4, we define

$$\begin{aligned} \text{NumbSol}[t, x, \kappa] := & \text{NumbSol}[t', x, \kappa] \\ & + \sum_{\substack{a \in \mathbb{A}_u \text{ s.t.} \\ a+1=x[u]}} \sum_{\substack{b \in \mathbb{A}_v \text{ s.t.} \\ b+1=x[v]}} \text{NumbSol}[t', x \sqcup (u \mapsto a, v \mapsto b), \kappa - 1] \end{aligned}$$

for the table entries. We observe that each of the sums considers at most two values. Indeed, if $x[u]$ is an integer, then only $a = x[u] - 1$ is possible and if $x[u]$ is equal to $\top$, then only the choices $a = \top$ and $a = \text{top}(v) - 1$ are possible. The same observation applies to v .

Forget Node: Assume that the node t' is the unique child of the current node t and denote by v the vertex that is forgotten at the node t , that is, $\beta(t) = \beta(t') \setminus \{v\}$.

If $\Lambda(v)$ is finite, then we set

$$\text{NumbSol}[t, x, \kappa] := \sum_{d \in \Lambda(v)} \text{NumbSol}[t', x \sqcup (v \mapsto d), \kappa].$$

Otherwise, when the set $\Lambda(v)$ is cofinite, we set

$$\begin{aligned} \text{NumbSol}[t, x, \kappa] := & \text{NumbSol}[t', x \sqcup (v \mapsto \top), \kappa] \\ & + \sum_{d \in \Lambda(v) \cap [0.. \text{top}(v)]} \text{NumbSol}[t', x \sqcup (v \mapsto d), \kappa]. \end{aligned}$$

Join Node: Assume that the two nodes t_1 and t_2 are the two children of the current node t and these three nodes satisfy $\beta(t) = \beta(t_1) = \beta(t_2)$.

We define two new functions $g_1, g_2: \mathbb{A}(t) \times [0..m] \rightarrow [0..2^m]$ where, for all state-vectors $x \in \mathbb{A}(t)$ and for all size bounds $s \in [0..m]$, we set

$$g_1(x, \kappa) := \text{NumbSol}[t_1, x, \kappa] \quad \text{and} \quad g_2(x, \kappa) := \text{NumbSol}[t_2, x, \kappa].$$

As a next step, we use the fast convolution algorithm from Main Theorem I.1, to compute the function $g: \mathbb{A}(t) \times [0..m] \rightarrow [0..2^m]$ as the convolution of the two functions g_1 and g_2 which is defined as

$$g(x, \kappa) = \sum_{\substack{x_1, x_2 \in \mathbb{A}(t) \\ \text{s.t. } x = x_1 \oplus x_2}} \sum_{\substack{\kappa_1, \kappa_2 \in [0..m] \\ \text{s.t. } \kappa = \kappa_1 + \kappa_2}} g_1(x_1, \kappa_1) \cdot g_2(x_2, \kappa_2)$$

for all x and κ .

As a last step for the join node, we define the table entries as

$$\text{NumbSol}[t, x, \kappa] := g(x, \kappa).$$

For the last step of the algorithm we assume that the node r denotes the root of the given tree decomposition. If an integer κ with $\kappa \in [0..m]$ is given as input of the instance, then the algorithm outputs the value $\text{NumbSol}[r, \emptyset, \kappa]$ as answer. Otherwise, that is, if $\kappa > m$ or $\kappa < 0$, then the algorithm outputs 0 as answer.

This concludes the description of the algorithm. It remains to analyze the correctness and the running time.

Claim 6.1.1.1. *The algorithm is correct, that is, for all choices of a node t , a state-vector x , and a size bound κ , the algorithm sets $\text{NumbSol}[t, x, \kappa] = c$ if and only if there are c partial solutions of size κ satisfying the degree constraints on the vertices from $\beta(t)$ as defined by the state-vector f .*

Proof of Claim. The correctness of the algorithm follows by induction. The case of leaf nodes, introduce vertex nodes, introduce edge nodes, and forget nodes follows directly from the definition of the algorithm.

The only non-trivial case is when the node t is a join node. Here we first crucially exploit the existence of a unique introduce edge node for each edge. With this we can assume that each edge (especially the ones connecting vertices in the bag $\beta(t)$) is introduced exactly once. Thus, such an edge cannot appear in the partial solution for t_1 and t_2 , but only in at most one of them.

Note that the edge might be introduced further up the decomposition tree.

By this assumption, the straightforward definition of the convolution is actually correct as we do not have to keep track of edges that might be counted twice. Indeed, it suffices to consider all possible ways of how the partial solutions for t of size κ could be formed based on the partial solutions for t_1 and t_2 . By the correctness of the convolution from Main Theorem I.1, we know that the function g is the convolution of g_1 and g_2 and thus, the correctness follows. $\triangleleft$

As a last ingredient before we finish the proof, we analyze the running time of the algorithm.

Claim 6.1.1.2. *The running time of the algorithm is*

$$\eta \cdot (n + \overline{H})^{\mathcal{O}(1)} + 2^{\text{tw}} \cdot n^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}(\overline{H})}.$$

Proof of Claim. If the current node t is a leaf node, introduce vertex node, introduce edge node, or forget node, with node t' as child, then the values $\text{NumbSol}[t, \star, \star]$ can be easily

computed from the values $\text{NumbSol}[t', \star, \star]$ by at most n operations each. Therefore, for each such node t , this computation takes time $|\mathbb{A}(t)| \cdot n^{\mathcal{O}(1)}$ as $m \in \mathcal{O}(n^2)$.

If the current node t is a join node, we use Main Theorem I.1 to compute the convolution. We write $\overline{H}(t) = \prod_{h \in H(t)} h$ with $H(t) := \{3\} \cup \{\text{top}(v) \mid v \in \beta(t)\}$ and $A_+(t) = \sum_{v \in \beta(t)} (1 + \text{top}(v))$ in the following. Then, the computation of the convolutions takes time

$$|\mathbb{A}(t)| \cdot A_+(t) \cdot (\log |\mathbb{A}(t)| + \log A_+(t)) \cdot (m + \overline{H}(t))^{\mathcal{O}(1)} + 2^{\text{tw}+1} \cdot (m + \exp(\overline{H}(t)))^{\mathcal{O}(1)}$$

in total.

Since the size of a bag is at most $\text{tw} + 1$ and by the assumption that $\text{top}(v) \leq m$ for all vertices v , we get that $A_+(t) \leq n^{\mathcal{O}(1)}$. Moreover, by definition of η , we know that $|\mathbb{A}(t)| \leq \eta$ and hence, we get $\log |\mathbb{A}(t)| \leq n^{\mathcal{O}(1)}$.

Based on all these observations, we conclude that the computation at a join node takes time

$$\eta \cdot (n + \overline{P})^{\mathcal{O}(1)} + 2^{\text{tw}+1} \cdot (n + 2^{\overline{P}})^{\mathcal{O}(1)}.$$

This concludes the proof of the running time of the algorithm since the number of nodes in a nice tree decomposition is polynomial (because of the introduced edge nodes) in the size of the underlying graph. $\triangleleft$

The combination of the algorithm with the analysis of the correctness from Claim 6.1.1.1 and the bound on the running time from Claim 6.1.1.2, finishes the proof of Main Lemma 6.1.1. $\square$

6.2 Parameterizing by the Number of Excluded Degrees

In this section we prove that ANTIFACTOR_x is FPT parameterized by treewidth and the *number* x of forbidden degrees. While the algorithm still follows the standard approach of a dynamic program on the tree decomposition as for the general algorithm from Section 6.1, we use representative sets to store the information more efficiently which results in a faster running time.

Recall that the general algorithm fills a table $\text{NumbSol}[t, x, \kappa]$ for all nodes t , all state-vectors x , and all solution sizes κ . Since each table entry comprises a single integer, we cannot simply compute a representative set of the table entries. Moreover, the size of the table NumbSol provides a lower bound on the running time. Thus, we need to change the definition of the table and hence, have to sacrifice the possibility to store all information available. The new algorithm can especially not solve the counting version but is limited to the decision version and the optimization version.

Now, to solve the exact decision version, it suffices to store a set of the possible state-vectors. That is, it suffices to store the set $\{\vec{u} \in \mathbb{A}(t) \mid \text{NumbSol}[t, x, \kappa] > 0\}$ for all nodes t and size bounds κ . With this observation it is straightforward how to use representative sets to obtain a faster algorithm; the algorithm does not store the previously mentioned set but a representative set thereof. We formalize these ideas in the proof of Main Theorem II.2.

Main Theorem II.2. *Fix some integer $x \geq 1$. Given an ANTIFACTOR_x instance G along with a tree decomposition of width w , we can decide if there is a solution of size exactly s in time $(x + 1)^{\mathcal{O}(w)} \cdot |G|^{\mathcal{O}(1)}$ for all s simultaneously.*

Later we show that this limitation to the exact decision version is not a shortcoming of our design but inherent to the problem. We refer to Main Theorem II.6 for the formal statement of the corresponding lower bound.

Idea of the Algorithm. As previously mentioned, the improved algorithm from Main Theorem II.2 follows the standard approach for dynamic programs on tree decompositions. In addition, we use a suitable version of representative sets to keep the size of those sets small which we store for each node of the tree decomposition.

More precisely, we use the definition of H -compatibility and H -representation as defined in Definitions 3.2.1 and 3.2.2 from Section 3.2. We instantiate the graph H with the compatibility graph as defined in Definition 3.2.11 where we use the sets $\mathbb{N} \setminus \Lambda(\star)$ of the vertices of the graph for the sets from the compatibility graph.

Definition 3.2.11 (Compatibility Graph). *Consider a list $\mathcal{F} = (F_1, \dots, F_k)$ of k sets $F_i \subseteq \mathbb{N}$, for all $i \in [1..k]$, which are considered to be sets of forbidden numbers.*

We define the (infinite) compatibility graph $\mathcal{C}_{\mathcal{F}}$ of $\mathcal{F}$ with

- *vertex set $V(\mathcal{C}_{\mathcal{F}}) = L \dot{\cup} R$ where the two sets L and R are disjoint copies of $\mathbb{N}^k$, and*
- *edge set $E(\mathcal{C}_{\mathcal{F}}) = \{((a_1, \dots, a_k), (b_1, \dots, b_k)) \mid \forall i \in [1..k] : a_i + b_i \notin F_i\}$.*

From now on, we use the following notation. We assume that the vertices of the given graph are ordered by some arbitrary but fixed ordering. Then, for a bag $\beta(t) = \{v_1, \dots, v_k\}$ of a node t of the tree decomposition (where we assume that the indices represent the ordering), we denote by $\mathcal{C}_t$ the graph $\mathcal{C}_{\mathcal{F}(t)}$ where we define $\mathcal{F}(t) = (\mathbb{N} \setminus \Lambda(v_1), \dots, \mathbb{N} \setminus \Lambda(v_k))$ as the list of the sets with the forbidden degrees.

With this notation, the goal of our improved algorithm is to fill a table $\text{ParSol}[\star, \star]$ as follows. For all nodes $t \in V(T)$ of the tree decomposition (T, β) and all size bounds $\kappa \in [0..m]$, we define $\text{ParSol}[t, \kappa] \subseteq \mathbb{A}(t)$ such that

$$\text{ParSol}[t, \kappa] \subseteq_{\mathcal{C}_t\text{-rep}} \{\vec{u} \in \mathbb{A}(t) \mid \text{NumbSol}[t, x, \kappa] > 0\}.$$

This especially implies that for all $x \in \text{ParSol}[t, \kappa]$, there is a partial solution $S \subseteq E_t$ with $|S| = \kappa$ such that $\deg_S(v) \in \Lambda(v)$ for all $v \in V_t \setminus \beta(t)$, and $\deg_S(v) = x[v]$ for all $v \in \beta(t)$.

With the previous idea, the algorithm decomposes into two tasks. First, we have to construct the table entries by a bottom-up approach given the table entries of the child nodes. Second, we have to compute a representative set of the resulting set which we then store in the table ParSol .

A Meta Algorithm. For the second step, we use Lemma 3.2.13 from Section 3.2 to compute the $\mathcal{F}$ -representative sets. Hence, we first focus on the construction of the table entries.

To this end, we do not explicitly use the results from Section 3.2 but assume that we have access to *some* procedure which computes these sets and provides some guarantee on the size of the resulting sets. Instead of designing the algorithm from scratch, we additionally use the observation that certain operations on the state-vectors preserve the representative set.

Lemma 3.2.16. Let $F_1, \dots, F_k, P_1, \dots, P_\ell \subseteq \mathbb{N}$ denote $k + \ell$ finite sets and write $\mathcal{F} = (F_1, \dots, F_k)$ and $\mathcal{P} = (P_1, \dots, P_\ell)$. Further, write $\mathcal{S} \subseteq \mathbb{N}^{k+\ell}$ for a finite set and let $\mathcal{S}'$ denote an $(\mathcal{F}; \mathcal{P})$ -representative set of $\mathcal{S}$.

- (a) For every vector $d \in \mathbb{Z}^{k+\ell}$, the set $\mathcal{S}' + d := \{s + d \mid s \in \mathcal{S}\}$ is an $(\mathcal{F}; \mathcal{P})$ -representative set of $\mathcal{S} + d$.
- (b) For every $i \in [1..k]$, the set $\mathcal{S}' \boxplus i := \{s[1..i-1] \ s[i+1..k+\ell] \mid s \in \mathcal{S}', s[i] \notin F_i\}$ is an $(\mathcal{F}'; \mathcal{P})$ -representative set of $\mathcal{S} \boxplus i$ where $\mathcal{F}' := (F_1, \dots, F_{i-1}, F_{i+1}, \dots, F_k)$.
- (c) For every $j \in [1..\ell]$, the set $\mathcal{S}' \boxplus (k+j) := \{s[1..k+j-1] \ s[k+j+1..k+\ell] \mid s \in \mathcal{S}', s[k+j] \in P_j\}$ is an $(\mathcal{F}; \mathcal{P}')$ -representative set of $\mathcal{S} \boxplus (k+j)$ where $\mathcal{P}' := (P_1, \dots, P_{j-1}, P_{j+1}, \dots, P_\ell)$.

For every finite set $Q \subseteq \mathbb{N}$, and

- (d) for every $i \in [1..k+1]$, the set $\mathcal{S}' \boxplus i := \{s[1..i-1] \ 0 \ s[i..k+\ell] \mid s \in \mathcal{S}'\}$ is an $(\mathcal{F}'; \mathcal{P})$ -representative set of $\mathcal{S} \boxplus i$ where $\mathcal{F}' := (F_1, \dots, F_{i-1}, Q, F_i, \dots, F_k)$, and
- (e) for every $j \in [1..\ell+1]$, the set $\mathcal{S}' \boxplus (k+j) := \{s[1..k+j-1] \ 0 \ s[k+j..k+\ell] \mid s \in \mathcal{S}'\}$ is an $(\mathcal{F}; \mathcal{P}')$ -representative set of $\mathcal{S} \boxplus (k+j)$ where $\mathcal{P}' := (P_1, \dots, P_{j-1}, Q, P_j, \dots, P_\ell)$.

For every finite set $\mathcal{R} \subseteq \mathbb{N}^{k+\ell}$ and every $(\mathcal{F}; \mathcal{P})$ -representative sets $\mathcal{R}'$ of $\mathcal{R}$,

- (f) the set $\mathcal{S}' \cup \mathcal{R}'$ is an $(\mathcal{F}; \mathcal{P})$ -representative set of $\mathcal{S} \cup \mathcal{R}$, and
- (g) the set $\mathcal{S}' + \mathcal{R}' := \{s + r \mid s \in \mathcal{S}', r \in \mathcal{R}'\}$ is an $(\mathcal{F}; \mathcal{P})$ -representative set of $\mathcal{S} + \mathcal{R}$.

As we assume that the vertices are ordered, we identify the vertices with their position. For example, when vertex v should appear at position i , we abuse notation and just write $\mathcal{S} \boxplus v$ instead of $\mathcal{S} \boxplus i$. Similar for the other modifications.

With these preliminaries we have everything ready to state the meta algorithm when parameterizing by treewidth and the size of the list of forbidden degrees.

Lemma 6.2.1. Assume, for a list $\mathcal{F}$ of k sets of at most f numbers each, we can compute, for a set $\mathcal{S} \subseteq \mathbb{N}^k$, a set $\mathcal{S}' \subseteq_{\mathcal{C}_{\mathcal{F}}\text{-rep}} \mathcal{S}$ of size $\text{Size}(k)$ in time $\text{Time}(k, |\mathcal{S}|)$, where Time and Size are allowed to depend on $\mathcal{F}$ and f .

Then, for a given $\text{ANTI} \text{FACTOR}_x$ instance G we can decide whether there is a solution of size exactly s in time

$$\text{Time}(w+1, (m+1) \cdot \text{Size}(w+1)^2) \cdot |G|^{\mathcal{O}(1)}$$

given a tree decomposition of width w and in time

$$\text{Time}(w+1, 2 \cdot \text{Size}(w+1)) \cdot |G|^{\mathcal{O}(1)}$$

given a path decomposition of width w .

Proof. We assume that the given tree decomposition is nice and contains introduce edge nodes. Moreover, we assume that each edge of the graph is introduced exactly once.

As the first step, we create a table $\text{ParSol}[\star, \star]$ and initialize each entry with $\emptyset$. That is, for all nodes t of the tree decomposition and all size bounds $\kappa \in [0..m]$, we set $\text{ParSol}[t, \kappa] := \emptyset$.

As a second step, we traverse the given tree decomposition in post-order and apply one of the following operations depending on the type of the current node t .

Leaf Node: The node t is a leaf node and thus, has an empty bag, that is, $\beta(t) = \emptyset$. We set $\text{ParSol}[t, 0] := \{\emptyset\}$. All other entries $\text{ParSol}[t, \kappa]$ where $\kappa \neq 0$ are not changed.

Introduce Vertex Node: Assume that the node t' is the child of the node t and that vertex v is introduced at the node t , that is, $\beta(t) = \beta(t') \cup \{v\}$. We define

$$\text{ParSol}[t, \kappa] := \text{ParSol}[t', \kappa] \boxplus v$$

for the table entries.

Introduce Edge Node: Assume that the node t' is the child of the current node t and that the edge uv is introduced at the node t . We need to consider the two cases where the edge uv is not selected and where the edge is selected.

We write $\chi_{uv} \in \{0, 1\}^{\beta(t)}$ for the vector where $\chi_{uv}[u] = \chi_{uv}[v] = 1$ and $\chi_{uv}[w] = 0$ for all $w \in \beta(t) \setminus \{u, v\}$. Now, for a size bound κ , we first set

$$R_\kappa := \text{ParSol}[t', \kappa] \cup (\text{ParSol}[t', \kappa - 1] + \chi_{uv}).$$

Then, we compute a $\mathcal{C}_t$ -representative set $\mathcal{S}_\kappa$ of R_κ and define $\text{ParSol}[t, \kappa] := \mathcal{S}_\kappa$ as the final table entry.

Forget Node: Assume that the node t' is the child of the current node t where vertex v is forgotten, that is, $\beta(t) = \beta(t') \setminus \{v\}$. We define an auxiliary set as

$$R_\kappa := \text{ParSol}[t', \kappa] \boxminus v.$$

We compute a $\mathcal{C}_t$ -representative set $\mathcal{S}_\kappa$ of R_κ and then, define $\text{ParSol}[t, \kappa] := \mathcal{S}_\kappa$ as the final table entry.

Join Node: Assume that the two nodes t_1 and t_2 are the children of the current node t with $\beta(t) = \beta(t_1) = \beta(t_2)$. For a given size bound κ , we define an auxiliary set R_κ as

$$R_\kappa := \bigcup_{\bar{\kappa} \in [0.. \kappa]} \text{ParSol}[t_1, \bar{\kappa}] + \text{ParSol}[t_2, \kappa - \bar{\kappa}].$$

We compute a $\mathcal{C}_t$ -representative set $\mathcal{S}_\kappa$ of R_κ and then, define $\text{ParSol}[t, \kappa] := \mathcal{S}_\kappa$ as the final table entry.

For the third (and last) step of the algorithm we assume that the node r is the root of the tree decomposition with $\beta(r) = \emptyset$. For a given size bound $\kappa \in [0..m]$, the algorithm checks if $\text{ParSol}[r, \kappa] \neq \emptyset$, that is, if $\text{ParSol}[r, \kappa]$ contains the empty vector, in this case there is a solution of size κ . Otherwise, no solution of size κ exists.

This concludes the description of the algorithm. In the following steps, we first prove the correctness and then analyze the running time.

Note that this dynamic program is somewhat dual to the previous algorithm. This comes from the fact that here the states correspond to the actual partial solutions where for the previous algorithm the vectors represent the solution we eventually want to build.

Claim 6.2.1.1. *The algorithm is correct, that is, for all nodes t and all size bounds $\kappa \in [0 .. m]$, the algorithm defines $\text{ParSol}[t, \kappa]$ such that*

$$\text{ParSol}[t, \kappa] \subseteq_{\mathcal{C}_t\text{-rep}} \{x \in \mathbb{A}(t) \mid \text{NumbSol}[t, x, \kappa] > 0\} = L_{t,\kappa}. \quad (6.2.1)$$

Proof of Claim. We first recall that by the definition of H -representation from Definition 3.2.2, it follows that H -representation is transitive. This holds independently of the chosen graph H .

The proof follows by structural induction on the tree decomposition.

Leaf Node: Since the bag is empty and the graph G_t contains no vertices, there is only the empty solution and hence, $\text{ParSol}[t, 0]$ only contains the empty vector.

Introduce Vertex Node: Let v denote the vertex introduced at the node t and let t' be the unique child of the node t . As the vertex v is isolated in G_t , we know that $L_{t,\kappa} = L_{t',\kappa} \boxplus v$. Hence, by the induction hypothesis and by Case (d) of Lemma 3.2.16, the set $\text{ParSol}[t, \kappa]$ is a $\mathcal{C}_t$ -representative set of $L_{t,\kappa}$.

Introduce Edges Node: Assume that the edge uv is introduced at the node t with unique child node t' . As the edge might be selected for some solutions and the edge might not be selected for some other solutions, we have that $L_{t,\kappa} = L_{t',\kappa} \cup (L_{t',\kappa-1} + \chi_{uv})$ where χ_{uv} is the 0-1-vector indicating which vertex in $\beta(t)$ is incident to the edge uv . Hence, by the induction hypothesis and Cases (a) and (f) of Lemma 3.2.16, we know that R_κ is a $\mathcal{C}_t$ -representative set of $L_{t,\kappa}$ and the claim follows by transitivity.

Forget Node: Let v denote the vertex that is forgotten at the current node t which has the node t' as unique child. As the vertices in $V_t \setminus \beta(t)$ must satisfy the degree constraints, we have $L_{t,\kappa} = L_{t',\kappa} \boxminus v$. Hence, by the induction hypothesis and Case (b) of Lemma 3.2.16, the set R_κ is a $\mathcal{C}_t$ -representative set of $L_{t,\kappa}$ and the claim follows by transitivity.

Join Node: Let t_1 and t_2 denote the children of the current node t . As all edges are introduced exactly once, we can partition each solution of size κ according to the nodes t_1 and t_2 such that, for some appropriate choice of $\bar{\kappa}$, the partial solution from t_1 is of size $\bar{\kappa}$ and the contribution from t_2 is of size $\kappa - \bar{\kappa}$. Hence, we know that $L_{t,\kappa} = \bigcup_{\bar{\kappa} \in [0 .. \kappa]} L_{t_1,\bar{\kappa}} + L_{t_2,\kappa-\bar{\kappa}}$. By the induction hypothesis and Cases (f) and (g) of Lemma 3.2.16, the set $\text{ParSol}[t, \kappa]$ is a $\mathcal{C}_t$ -representative set of $L_{t,\kappa}$ and the claim follows by transitivity. $\blacktriangleleft$

Next we consider the analysis of the running time of the algorithm.

Claim 6.2.1.2. *The algorithm operates in time*

$$\text{Time}(\text{pw} + 1, 2 \cdot \text{Size}(\text{pw} + 1)) \cdot |G|^{\mathcal{O}(1)}$$

if a path decomposition of width pw is given and in time

$$\text{Time}(\text{tw} + 1, (m + 1) \cdot \text{Size}(\text{tw} + 1)^2) \cdot |G|^{\mathcal{O}(1)}$$

if a tree decomposition of width tw is given.

Proof of Claim. We can assume that both functions Time and Size are non-decreasing. Moreover, by the design of the algorithm, we can assume that the size of the given table entries is bounded by $\text{Size}(\text{tw} + 1)$ (or by $\text{Size}(\text{pw} + 1)$ for the case of path decompositions).

The running time follows immediately from the bound on the size of the sets R_k and the computation of the representative set thereof. $\triangleleft$

This concludes the proof of our meta algorithm for ANTIFACTOR_x . $\square$

Finalizing the Algorithm. We combine our meta algorithm from Lemma 6.2.1 with the algorithm to compute the $\mathcal{F}$ -representative sets from Lemma 3.2.13 in Section 3.2.

Lemma 3.2.13. *Consider a list $\mathcal{F} = (F_1, \dots, F_k)$ of k sets each containing at most r numbers (which are considered to be forbidden), that is, for all $i \in [1..k]$, we have $F_i \subseteq \mathbb{N}$ and $|F_i| \leq r$.*

Further, let $S \subseteq \mathbb{N}^k$ denote a finite set. Then, one can compute a $\mathcal{C}_{\mathcal{F}}$ -representative set S' of S such that $|S'| \leq (r + 1)^k$ in time $\mathcal{O}(|S| \cdot (r + 1)^{k \cdot (\omega - 1)} \cdot k)$.

Then, the proof of Main Theorem II.2 follows immediately.

Corollary 6.2.2. *Given an instance G of ANTIFACTOR_x , we can decide if there is a solution of size exactly s in time $(x + 1)^{(\omega + 1) \cdot w} \cdot |G|^{\mathcal{O}(1)}$ if we are given a tree decomposition of width w and in time $(x + 1)^{\omega \cdot w} \cdot |G|^{\mathcal{O}(1)}$ if we are given a path decomposition of width w .*

Main Theorem II.2. *Fix some integer $x \geq 1$. Given an ANTIFACTOR_x instance G along with a tree decomposition of width w , we can decide if there is a solution of size exactly s in time $(x + 1)^{\mathcal{O}(w)} \cdot |G|^{\mathcal{O}(1)}$ for all s simultaneously.*

Proof of Main Theorem II.2 and Corollary 6.2.2. We use the meta algorithm from Lemma 6.2.1 and whenever a representative set of a set S is needed, we use Lemma 3.2.13 to compute a representative set of S .

To obtain the final size bound we use the size bounds provided by the meta algorithm and the fact that $x, w \leq n$. $\square$

Lower Bounds for General Factor

In the following, we prove the conditional lower bounds for the decision and counting versions of special cases of GENFAC. These lower bounds complement our algorithmic results in Chapter 6 and prove the optimality of the algorithm in many cases.

Decision Version. We first consider the decision version of GENFAC where all vertices are assigned the set Λ . That is, we consider Λ -FACTOR. We prove that our algorithm from Main Theorem II.1 is optimal when Λ is finite. For the case when Λ is cofinite, we prove a weaker lower bound.

Recall, that when the set Λ is cofinite, our algorithm from Main Theorem II.1 operates in time $(\max(\mathbb{N} \setminus \Lambda) + 2)^{tw} \cdot |G|$ (see Main Theorem II.1 for the precise statement). In contrast to this, we base the lower bound on the so-called half-induced matchings instead which depend on the set Λ (see Section 4.2 for further details).

Main Theorem II.3. *Fix some non-empty set $\Lambda \subseteq \mathbb{N}$ that is finite or cofinite with $0 \notin \Lambda$ and max-gap $\Lambda > 1$. If Λ is finite, we define $h = \max \Lambda + 1$, and if Λ is cofinite, then we define h as the size of the largest half-induced matching in the compatibility graph of $\mathbb{N} \setminus \Lambda$.*

If there is an $\varepsilon > 0$ and an algorithm that solves Λ -FACTOR on a given graph G with a given path decomposition of width w in time $(h - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-SETH is false.

For the case of cofinite sets, we also consider the variant where we parameterize by the number of forbidden degrees. By using representative sets we obtain a faster algorithm for the decision version where the running time depends on the number of forbidden degrees, say x (see Main Theorem II.2 for the formal statement). We prove that this algorithm is essentially optimal up to an improvement in the exponent.

Main Theorem II.4. *Let $x \geq 2$ denote a fixed integer.*

If there is an $\varepsilon > 0$ and an algorithm that solves ANTIFACTOR_x on a given graph G with a given path decomposition of width w in time $(x + 1 - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-SETH is false.

Note that the condition $x \geq 2$ in the statement of Main Theorem II.4 is actually needed to exclude the polynomial-time solvable cases.

Theorem 7.1.1. *ANTIFACTOR_x can be solved in polynomial time for $x \in \{0, 1\}$.*

Proof. Indeed, for $x = 0$ no degree is forbidden and all subsets of edges are solutions.

When $x = 1$, for each vertex v at most one degree is forbidden in $\Lambda(v)$ and thus, the set $\Lambda(v)$ contains a gap of size at most 1. We can use the algorithm by Cornuéjols to solve the decision version in polynomial time (see Theorem 5.1.4 and [30]).

We conclude that ANTIFACTOR_x is polynomial-time solvable when $x \in \{0, 1\}$. $\square$

Counting Version. We extend the lower bound from Main Theorem II.3 to the counting version and prove matching bounds in all cases. Indeed, for cofinite sets Λ the counting version becomes harder than the decision version. Thus, we do not have to rely on half-induced matchings to prove a lower bound. Instead, we can use the expressiveness of the counting problem to show a matching lower bound compared to the algorithm from Main Theorem II.1 for all cases.

Main Theorem II.5 (Lower Bound for Counting Version). *Fix a non-empty set $\Lambda \subseteq \mathbb{N}$ that is finite with $\Lambda \neq \{0\}$ or cofinite with $\Lambda \neq \mathbb{N}$ and $\Lambda \neq \mathbb{N}_{\geq 1}$.*

If there is an $\varepsilon > 0$ and an algorithm that solves Λ -#FACTOR on a given graph G with a given path decomposition of width w in time $(\text{top } \Lambda + 1 - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-#SETH is false.

Although the approach based on representative sets from Main Theorem II.2 does not extend to the counting version, there might be other approaches to obtain faster algorithms. However, we show that this is actually not the case and prove a lower bound even for the case when each vertex has only a single (individual) forbidden degree. Formally, we show that $\text{\#ANTI}(\text{FACTOR}_x)$ is $\text{\#W}[2]$ -hard and thus, most likely not fixed-parameter tractable.

Main Theorem II.6 (Lower Bound for Counting Version II). *There exists a constant $c > 0$ such that if there is an algorithm that solves $\text{\#ANTI}(\text{FACTOR}_1)$ on a given graph G with n vertices with a given path decomposition of width p in time $\mathcal{O}(n^{p-c})$, then #SETH is false.*

Moreover, $\text{\#ANTI}(\text{FACTOR}_1)$ is $\text{\#W}[2]$ -hard parameterized by pathwidth.

Outline. We do not prove each of these lower bounds individually. Instead, we modularize the proofs as much as possible to reuse partial results for multiple lower bounds. Therefore, we first outline the high-level structure of the proofs in Section 7.1 before we actually give the constructions.

7.1 Outline and Preliminaries

We show the lower bounds in several steps which can be grouped into three conceptual blocks. First, for each problem variant under consideration, we prove a lower bound for an intermediate problem where we allow additional relations. Then, in a second step, we reduce the number of different types of relations that can appear. Essentially, we narrow down the variety of the relations to two different types. Then, in a third step, we show how to remove these special relations and provide a reduction to the final problem variant without relations.

We refer to Figure 7.1 for an illustration of the individual steps of the reduction.

Relations. We first formalize what we mean by relations in the context of edge selection problems. Intuitively, we define relations as a family of edge subsets. Then, we can directly check whether a selection of edges satisfies the relation. However, formally this definition is based on the incident edges of a vertex and thus, does not immediately allow arguing about relations without a corresponding graph. To make this possible, we later additionally use the more standard notation for relations which is defined via sets of 0-1-vectors. By imposing an

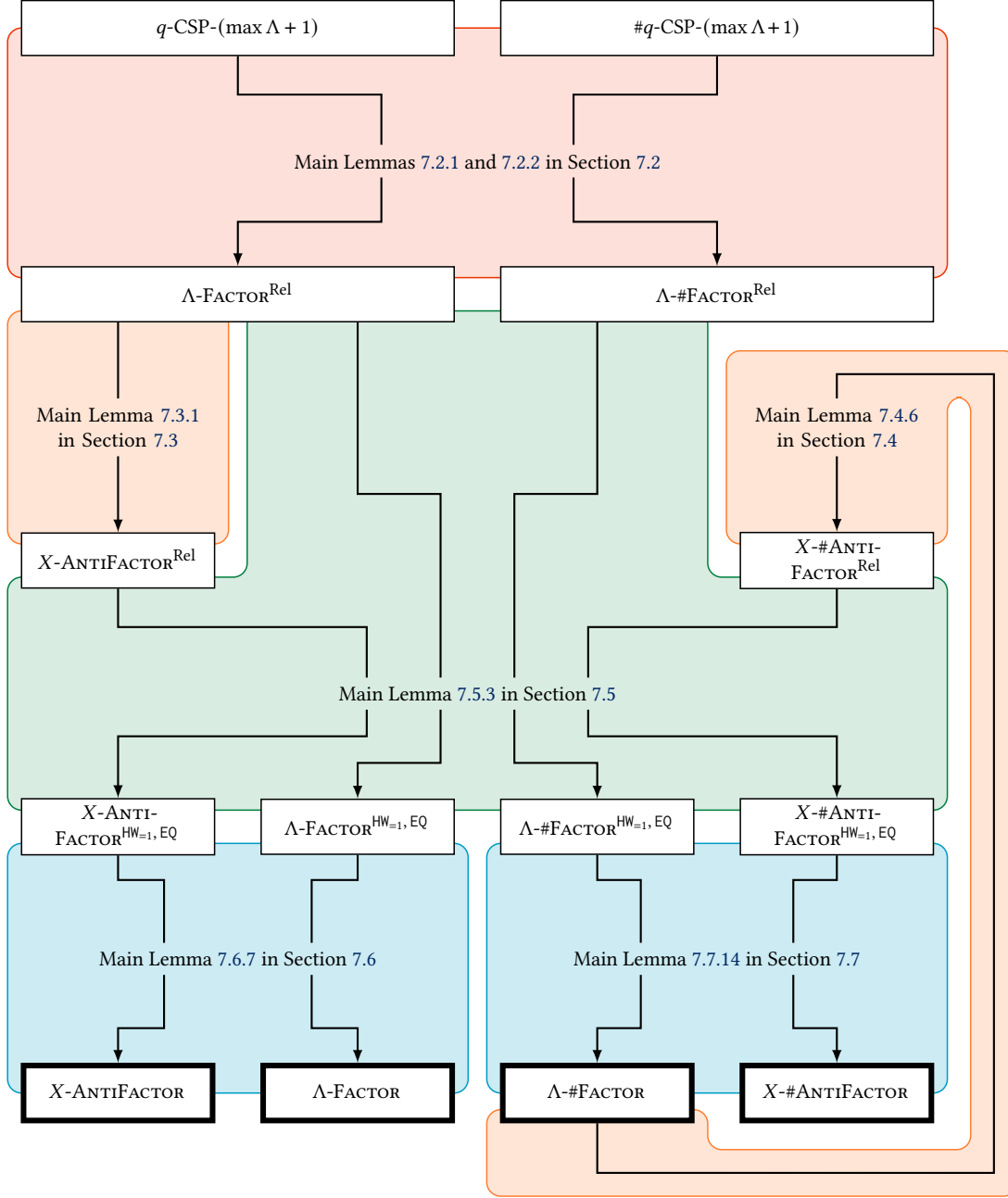


Figure 7.1: An overview of the main reductions from Chapter 7 which we use to prove the lower bounds for the variants of GENFAC.

The results for the decision version from Main Theorems II.3 and II.4 are shown on the left side. The results for the counting version from Main Theorems II.5 and II.6 are shown on the right side.

Each highlighted block indicates that we prove these results in one section. Blocks of the same color indicate the similarities between the results for the decision version and counting version.

We omit the restrictions for the sets Λ and X to improve readability.

ordering on the incident edges of a vertex, both definitions are interchangeable. Thus, we use the definition that is more convenient at the corresponding place.

To see this correspondence, consider some relation R_v from above and fix an arbitrary ordering of the incident edges $I(v)$ of v . We define a new relation $R'_v \subseteq \{0, 1\}^{|I(v)|}$ where, for each set $S \in R_v$, we add the corresponding characteristic vector of this set S with respect to the ordered set $I(v)$ to the new relation R'_v . With this equivalence in mind, the truth table representation of R_v is simply a list of all subsets of $I(v)$ (or some suitable encoding thereof) extended by a 0-1-value indicating whether the corresponding set is excluded or included in R_v .

With this observation in mind, we define the intermediate problem as follows.

Definition 7.1.2 (Λ -FACTOR WITH RELATIONS FROM Rel (Λ -FACTOR^{Rel})). *Consider a fixed set $\Lambda \subseteq \mathbb{N}$ and a set $\mathcal{R}$ of relations.*

A vertex labeled graph $G = (V_S \dot{\cup} V_C, E, \lambda: V_C \rightarrow \mathcal{R})$ is an instance of Λ -FACTOR WITH RELATIONS FROM $\mathcal{R}$, or Λ -FACTOR ^{$\mathcal{R}$} for short, if

- *all vertices in V_S , called simple vertices, are labeled with the set Λ ,*
- *all vertices $v \in V_C$, called complex vertices, are labeled with a relation $R_v = \lambda(v)$ that is given as a truth table such that $R_v \subseteq 2^{I(v)}$ where $I(v)$ is the set of edges incident to v in G .*

An edge set $S \subseteq E$ is a solution for G if

- *for all simple vertices $v \in V_S$, we have $\deg_S(v) \in \Lambda$ and*
- *for all complex vertices $v \in V_C$, we have $I(v) \cap S \in R_v$.*

For Λ -FACTOR ^{$\mathcal{R}$} the task is to decide if a given instance G has a solution.

If we do not restrict the set $\mathcal{R}$ to specific relations, we write $\mathcal{R} = \text{Rel}$ for the set of all computable relations.

Similarly to the problem X -ANTI Λ -FACTOR, we write X -ANTI Λ -FACTOR WITH RELATIONS FROM Rel , or short X -ANTI Λ -FACTOR^{Rel}, for the problem Λ -FACTOR^{Rel} where $\Lambda = \mathbb{N} \setminus X$ for a finite set X .

Definition 7.1.3 (Measures of a Graph with Relations). *We define the following measures for an instance G .*

- *The size of G is the size of the graph plus the size of the relations.*
- *We refer to $\max_{v \in V_C} \deg(v)$ as the arity of the instance.*
- *The pathwidth (resp. treewidth) of G is defined as the pathwidth (resp. treewidth) of the graph G' which is obtained from G where, for each $v \in V_C$, we introduce a clique on the vertices in $N(v)$.*
- *The total arity of G is defined as $\Delta^* := \max_{\text{bag } \beta} \sum_{v \in \beta \cap V_C} \deg(v)$, where the bags correspond to a given path (resp. tree) decomposition.*

In the following we outline the different intermediate steps to prove the lower bounds.

7.1.1 Lower Bound for the Intermediate Problem

Using this intermediate problem, we can formally state the first part of the reduction. We observe that the lower bound needs a careful formulation. Indeed, when we later reduce from $\Lambda\text{-FACTOR}^{\text{Rel}}$ to $\Lambda\text{-FACTOR}$, we insert gadgets realizing the relations at the complex vertices, and thus, the size and pathwidth of the graph can increase significantly.

Therefore, we state a stronger lower bound that can tolerate additional terms to take care of such increases. The key point is that this increase is mainly influenced by the total arity of the complex vertices in a bag of the path decomposition.

Main Lemma 7.2.1. *Denote by $\Lambda \subseteq \mathbb{N}$ a fixed, finite set with $\Lambda \neq \{0\}$. Let $f_\Lambda: \mathbb{N} \rightarrow \mathbb{R}^+$ be an arbitrary (computable) function that may depend on the set Λ .*

For all $\varepsilon > 0$, no algorithm can, for a $\Lambda\text{-FACTOR}^{\text{Rel}}$ instance G ,

- that is given with a path decomposition of G of width p ,*
- where the simple vertices of G form an independent set,*
- every simple vertex is adjacent to two complex vertices by $\max \Lambda$ edges each,*
- all simple vertices have degree exactly $\max \Lambda$ in every solution,*
- the total arity satisfies $\Delta^* \in \mathcal{O}(\max \Lambda)$,*

decide if there is a solution for G in time $(\max \Lambda + 1 - \varepsilon)^{p + f_\Lambda(\Delta^*)} \cdot |G|^{\mathcal{O}(1)}$ unless MONO-pw-CSPH fails.

As the lower bounds are stated for pathwidth, they immediately hold for treewidth as treewidth can be upper-bounded by pathwidth for all graphs.

By a careful design of the reduction, we obtain the corresponding result for the counting version of the problem.

Main Lemma 7.2.2. *Denote by $\Lambda \subseteq \mathbb{N}$ a fixed, finite set with $\Lambda \neq \{0\}$. Let $f_\Lambda: \mathbb{N} \rightarrow \mathbb{R}^+$ be an arbitrary (computable) function that may depend on the set Λ .*

For all $\varepsilon > 0$, no algorithm can, for a $\Lambda\text{-FACTOR}^{\text{Rel}}$ instance G ,

- that is given with a path decomposition of G of width p ,*
- where the simple vertices of G form an independent set,*
- every simple vertex is adjacent to two complex vertices by $\max \Lambda$ edges each,*
- all simple vertices have degree exactly $\max \Lambda$ in every solution,*
- the total arity is Δ^* ,*

count the number of solution of G in time $(\max \Lambda + 1 - \varepsilon)^{p + f_\Lambda(\Delta^*)} \cdot |G|^{\mathcal{O}(1)}$ unless MONO-pw-#CSPH fails.

We dedicate Section 7.2 to the proofs of Main Lemmas 7.2.1 and 7.2.2 which follow from *one* careful construction.

From Main Theorem I.3 we know that pw-SETH implies MONO-pw-CSPH and similarly for the counting variants.

Main Theorem I.3. *If pw-SETH holds, then MONO-pw-CSPH holds.*

Moreover, if pw-#SETH holds, then MONO-pw-#CSPH holds.

Both results hold even if all constraints have a fixed parity.

Hence, our lower bounds from Main Lemmas 7.2.1 and 7.2.2 also hold under the more established pw-SETH and pw-#SETH and thus, SETH and #SETH, respectively.

7.1.2 Extension to Cofinite Sets

As the previous intermediate lower bounds from Main Lemmas 7.2.1 and 7.2.2 hold for finite sets only, we extend these results to the case of cofinite sets. For this we provide the reduction from Main Lemma 7.3.1 for the decision version, which uses half-induced matchings to determine the bound.

Main Lemma 7.3.1. *Fix a finite, non-empty set $X \subseteq \mathbb{N}$ which contains a half-induced matching of size $h \geq 2$ with U as largest value.*

For every $\Lambda \subseteq \mathbb{N}$ with $\max \Lambda = h - 1$, there is a pathwidth-preserving many-one reduction from Λ -FACTOR^{Rel} with the restrictions from Main Lemma 7.2.1 to X -ANTIFACTOR^{Rel} such that the total arity increase by a factor of U , and the size by a factor of 2^U .

We prove Main Lemma 7.3.1 in Section 7.3.

Similarly, we also extend the result for the counting version to cofinite sets. We note that this reduction does not depend on the half-induced matchings and thus, can possibly yield a better bound. However, this reduction starts from Λ -#FACTOR, that is, the problem *without* relations.

Main Lemma 7.4.6. *Fix a finite, non-empty set $X \subseteq \mathbb{N}$.*

There is a pathwidth-preserving reduction starting with an instance H of $\{\max X + 1\}$ -#FACTOR and queries $|H|^{\mathcal{O}(\max X)}$ instances of X -#ANTIFACTOR^{Rel} where the size is bounded by $\mathcal{O}(|H| \cdot \log^2 |H|)$, and the total arity Δ^* is bounded by $\mathcal{O}(\max X)$.

We prove Main Lemma 7.4.6 in Section 7.4.

7.1.3 Reducing to Few Types of Relations

To obtain the lower bound for the intermediate problem, the reductions introduce arbitrary relations. In the next step, we replace these relations by suitable gadgets involving only two types of relations, namely $\text{HW}_{=1}$ and EQ. The observation that such a replacement is possible was first done by Curticapean and Marx [35] in a lower bound for COUNTING PERFECT MATCHINGS.

We give the formal definition of these relations in Definition 2.1.6 which we restate here for convenience.

Definition 2.1.6. For a k -dimensional vector $x \in \{0, 1\}^k$, we define $\text{hw}(x)$ as the number of 1s in x , that is, we define $\text{hw}(x)$ as the Hamming weight of x with $\text{hw}(x) := \sum_{i=1}^k x[i]$.

For two positive integers $i, k \in \mathbb{N}$ and a set $S \subseteq \mathbb{N}$, we define the relations

$$\text{HW}_{\in S}^{(k)} := \{x \in \{0, 1\}^k \mid \text{hw}(x) \in S\}, \quad \text{HW}_{=i}^{(k)} := \text{HW}_{\in \{i\}}^{(k)} \quad \text{and} \quad \text{EQ}_k := \text{HW}_{\in \{0, k\}}^{(k)}.$$

We say that EQ_k is the equality relation on k inputs.

We use $\text{HW}_{\in S}$ to denote the relation $\text{HW}_{\in S}^{(k)}$ when the arity k of the relation is clear from the context. Additionally, we also write $\text{HW}_{\in S}$ to denote the set of the relations $\text{HW}_{\in S}^{(k)}$ for all $k \in \mathbb{N}$.

We transfer this abuse of notation to $\text{HW}_{=i}$ and EQ .

With this notation, we can now formally state the main result of the second step.

Main Lemma 7.5.3. Let $Y \subseteq \mathbb{N}$ be an arbitrary set that might even be a part of the input. There is a parsimonious polynomial-time many-one reduction that transforms an instance G of $Y\text{-FACTOR}^{\text{Rel}}$ into an instance H of $Y\text{-FACTOR}^{\text{HW}=1, \text{EQ}}$ such that for some function f

- if given a path decomposition of G of width w_G , then we can simultaneously construct a path decomposition of H of width $w_H \leq w_G + f(\Delta^*(G))$,
- the size is bounded by $|H| \leq |G| \cdot f(\Delta^*(G))$,
- the total arity is bounded by $\Delta^*(H) \leq f(\Delta^*(G))$
- the number of $\text{HW}_{=1}$ -relations in H is even, and
- if the relations of G are even, then the arity of each EQ -relation in H is even.

Additionally, the reduction is oblivious to the set Y and does not introduce any simple vertices.

We prove Main Lemma 7.5.3 in Section 7.5.

7.1.4 Realizing Relations

In the last step we remove the remaining relations of type $\text{HW}_{=1}$ and EQ . We replace these relations by appropriate gadgets which only consist of simple vertices.

For the decision version of $\Lambda\text{-FACTOR}$ we observe that the problem is trivial whenever $0 \in \Lambda$. Moreover, if $\max\text{-gap } \Lambda \leq 1$, then the algorithm by Cornuéjols solves the problem in polynomial time (see Theorem 5.1.4 and [30] for further details). Hence, we only consider the case when $0 \notin \Lambda$ and $\max\text{-gap } \Lambda > 1$.

Our goal is to use the result for multiple cases. Therefore, we introduce a slightly more general version of the problem where we allow two sets of allowed degrees to be assigned to the vertices. For two sets $\Lambda_1, \Lambda_2 \subseteq \mathbb{N}$, we define $(\Lambda_1, \Lambda_2)\text{-FACTOR}$ as the generalization of $\Lambda_1\text{-FACTOR}$ where each vertex might be assigned the sets Λ_1 or the set Λ_2 . We extend the definition of $(\Lambda_1, \Lambda_2)\text{-FACTOR}$ in the natural way to the counting version $(\Lambda_1, \Lambda_2)\text{-\#FACTOR}$.

Main Lemma 7.6.7. *Fix a non-empty set $\Lambda \subseteq \mathbb{N}$ with $0 \notin \Lambda$ and $\max\text{-gap } \Lambda > 1$ that is either finite or cofinite.*

Let $Y \subseteq \mathbb{N}$ be an arbitrary set that might even be a part of the input. There is a parsimonious many-one reduction from $Y\text{-FACTOR}^{\text{HW}=1, \text{EQ}}$ with an even number of $\text{HW}=1$ -relations and where all EQ-relations have even arity to $(\Lambda, Y)\text{-FACTOR}$ such that the width of a given path decomposition increases by at most $f(\Delta^*)$, and size increases by a factor of $f(\Delta^*)$.

We prove Main Lemma 7.6.7 in Section 7.6.

For the counting version we observe that significantly more cases are non-trivial than for the decision version. Although there is a trivial solution when $0 \in \Lambda$, counting the number of solutions is still hard. Similarly, the restriction $\max\text{-gap } \Lambda > 1$ is no longer needed. Instead, the counting version is difficult for almost all finite or cofinite sets.

Indeed, the only trivial cases are $\Lambda = \emptyset$ (there is no solution at all), $\Lambda = \{0\}$ (the empty solution is the unique solution), and $\Lambda = \mathbb{N}$ (all subsets of edges are a solution).

In addition to the trivial cases, we exclude the case $\Lambda = \mathbb{N} \setminus \{0\}$ from the following reduction because of a limitation of our reduction. We consider this case separately.

Main Lemma 7.7.14. *Let $\Lambda \subseteq \mathbb{N}$ be a non-empty set with $\Lambda \notin \{\{0\}, \mathbb{N} \setminus \{0\}, \mathbb{N}\}$ that is finite or cofinite. Let $Y \subseteq \mathbb{N}$ be an arbitrary set that might even be given as input.*

There is a Turing-reduction from $Y\text{-}\#\text{FACTOR}^{\text{HW}=1, \text{EQ}}$ to $(Y, \Lambda)\text{-}\#\text{FACTOR}$ with running time $f(\text{top } \Lambda + \Delta^*) \cdot n^{\mathcal{O}(1)}$, increasing the size by a factor of $f(\text{top } \Lambda)$, decreasing Δ^* to zero, and increasing the width of the path decomposition by an additive term of $\Delta^* \cdot f(\text{top } \Lambda)$ for each oracle call.

We prove Main Lemma 7.7.14 in Section 7.7.

7.1.5 Combining the Results

As a last step, we can combine the previous results to prove the lower bounds for the variants of GENFAC. We refer to Figure 7.1 for a visual representation of most of the following proof outlines.

Decision Version. We start with the lower bounds under pw-SETH for the decision version. We combine Main Lemmas 7.2.1, 7.5.3 and 7.6.7 to prove the first result when Λ is finite. When Λ is cofinite, we additionally use Main Lemma 7.3.1 to obtain an instance of $X\text{-ANTIFACTOR}^{\text{Rel}}$.

Main Theorem II.3. *Fix some non-empty set $\Lambda \subseteq \mathbb{N}$ that is finite or cofinite with $0 \notin \Lambda$ and $\max\text{-gap } \Lambda > 1$. If Λ is finite, we define $h := \max \Lambda + 1$, and if Λ is cofinite, then we define h as the size of the largest half-induced matching in the compatibility graph of $\mathbb{N} \setminus \Lambda$.*

If there is an $\varepsilon > 0$ and an algorithm that solves $\Lambda\text{-FACTOR}$ on a given graph G with a given path decomposition of width w in time $(h - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-SETH is false.

Proof. We distinguish two cases depending on whether Λ is finite or cofinite.

Finite Sets. We start with the easier case when Λ is finite. Consider a Λ -FACTOR^{Rel} instance H that satisfies the restrictions from Main Lemma 7.2.1. Especially, we assume that H is given with a path decomposition of width w_H and total arity $\Delta_H^* \leq \mathcal{O}(\max \Lambda)$.

We apply Main Lemma 7.5.3 and Main Lemma 7.6.7 to instance H to obtain an equivalent instance of Λ -FACTOR, say G .

Now assume, for some $\varepsilon > 0$, that there exists a faster algorithm for Λ -FACTOR. When we apply this algorithm to the instance G , we can solve Λ -FACTOR^{Rel} for the original instance H_1 .

Before we bound the running time of this procedure, we first analyze the instance G . The size of G is bounded by $|G| \in \mathcal{O}(|H| \cdot f(\Delta_H^*))$ and the width of the path decomposition of G by $w_G \leq w_H + f(\Delta_H^*)$.

Therefore, the algorithm for Λ -FACTOR^{Rel} takes time

$$\begin{aligned} (\max \Lambda + 1 - \varepsilon)^{w_G} \cdot |G|^{\mathcal{O}(1)} &\leq (\max \Lambda + 1 - \varepsilon)^{w_H + f(\Delta^*)} \cdot (|H| \cdot f(\Delta^*))^{\mathcal{O}(1)} \\ &\leq (\max \Lambda + 1 - \varepsilon)^{w_H + f'(\Delta^*)} \cdot |H|^{\mathcal{O}(1)} \end{aligned}$$

for some appropriately chosen function f' .

By Main Lemma 7.2.1, this immediately contradicts pw-SETH and hence, concludes the proof.

Cofinite Set. When Λ is cofinite we write $X \coloneqq \mathbb{N} \setminus \Lambda$ to simplify notation. Assume the compatibility graph for X contains a half-induced matching of size h .

Let H be a given $\{h - 1\}$ -FACTOR^{Rel} instance that satisfies the constraints from Main Lemma 7.2.1. We first apply Main Lemma 7.3.1 to obtain an instance H_1 of X -ANTIFACTOR^{Rel}.

Then, similar to the previous case, we apply Main Lemma 7.5.3 and Main Lemma 7.6.7 with $X = Y$ to obtain a X -ANTIFACTOR instance G .

Then, when we apply the faster algorithm for X -ANTIFACTOR to this instance G , this contradicts pw-SETH by Main Lemma 7.2.1. Indeed, the change of the size and the width of the path decomposition is bounded by $f(\Delta^*)$ where we use that U only depends on X and thus, is fixed. $\square$

When we parameterize by the *number* of forbidden degrees, that is, when we consider ANTIFACTOR _{x} , we exploit that we can assign different sets to the vertices. We follow the general idea from the previous step and use a first set to satisfy the constraint for the half-induced matchings from Main Lemma 7.3.1 and two more sets to realize relations.

Main Theorem II.4. *Let $x \geq 2$ denote a fixed integer.*

If there is an $\varepsilon > 0$ and an algorithm that solves ANTIFACTOR _{x} on a given graph G with a given path decomposition of width w in time $(x + 1 - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-SETH is false.

Proof. For a fixed integer $x \geq 3$, we define $Y \coloneqq \{2, 4, \dots, 2x\}$. By Lemma 4.2.2, the set Y contains a half-induced matching of size $x + 1$.

Assume we are given an instance H of $\{x\}$ -FACTOR^{Rel} that satisfies the constraints from Main Lemma 7.2.1. We first apply Main Lemma 7.3.1 to obtain an instance H_1 of Y -ANTIFACTOR^{Rel} where we exploit the properties of Y .

We use Main Lemma 7.5.3 to transform the instance H_1 into an instance H_2 of Y -ANTI-FACTOR^{HW₌₁, EQ}.

To remove the relations, we define two sets as $X_1 := \{0\}$ and $X_2 := \{1, 2\}$. By a close observation of Main Lemma 7.6.7 and the proofs in Section 7.6, we conclude that we can realize HW₌₁ and EQ with these two types of vertices. To see this, we note that a vertex with set X_1 can be seen as HW_{≥1} and HW₌₁⁽¹⁾. Similarly, a vertex of degree at most 2 that is assigned the set X_2 can be seen as HW_{≤1}. Moreover, we can use these vertices with set X_2 to realize EQ₃ by exploiting the gap of size 2. Then, the remaining relations can be realized by combining Lemmas 7.6.3 and 7.6.6 with the observations above.

We denote the resulting instance, which fulfills the requirements from Main Lemma 7.6.7 by H_3 .

By our choice of the sets, we know that $|X_1|, |X_2|, |Y| \leq x$. Hence, a faster algorithm for ANTI-FACTOR_x would give us a faster algorithm for $\{x\}$ -FACTOR^{Rel} as the size and width of the path decomposition change only by factor depending on Δ^* and x . However, this would contradict pw-SETH by Main Lemma 7.2.1. $\square$

This concludes our lower bounds for the decision version. For the remaining results we move to the counting version of GENFAC.

Counting Version. We follow the basic outline for the decision version and combine the intermediate results to prove the lower bounds.

For the counting version, we observe that Main Lemma 7.7.14 excludes the case $\Lambda = \mathbb{N}_{\geq 1} = \mathbb{N} \setminus \{0\}$, that is, the problem corresponding to #EDGECOVER. Instead, we consider this case separately in Theorem 7.9.6 which we prove in Section 7.9.

Theorem 7.9.6 (Lower Bound for Counting Edge Covers). *If there is an $\varepsilon > 0$ and an algorithm that solves #EDGECOVER on a given graph G with a given path decomposition of width w in time $(2 - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then #SETH is false.*

We first combine Main Lemmas 7.2.2, 7.5.3 and 7.6.7 to prove the lower bound for the case when Λ is finite. Then we combine this lower bound with Main Lemma 7.4.6 and apply Main Lemmas 7.5.3 and 7.6.7 again to prove the lower bound for the counting version when Λ is cofinite.

Main Theorem II.5 (Lower Bound for Counting Version). *Fix a non-empty set $\Lambda \subseteq \mathbb{N}$ that is finite with $\Lambda \neq \{0\}$ or cofinite with $\Lambda \neq \mathbb{N}$ and $\Lambda \neq \mathbb{N}_{\geq 1}$.*

If there is an $\varepsilon > 0$ and an algorithm that solves Λ -#FACTOR on a given graph G with a given path decomposition of width w in time $(\text{top } \Lambda + 1 - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-#SETH is false.

Proof. We first consider the case when Λ is finite as we use this result for the case when Λ is cofinite.

Finite Sets. Consider a Λ -#FACTOR^{Rel} instance H that satisfies the properties from Main Lemma 7.2.2. Especially, we assume that H is given with a path decomposition of width w_H , and total arity $\Delta^* \in \mathcal{O}(\max \Lambda)$.

We use Main Lemmas 7.5.3 and 7.6.7 to construct a polynomial (in $|H|$) number of instances of Λ -#FACTOR; we denote them by G_i . By the properties of the reductions, we immediately get that the size is bounded by $|G_i| \leq |H| \cdot f(\Delta^*)$ and the width of the path decomposition is bounded by $w_{G_i} \leq w_H + f(\Delta^*)$ where the function f is chosen according to Main Lemmas 7.5.3 and 7.6.7.

Now, suppose that, for some $\varepsilon > 0$, we can solve Λ -#FACTOR in the claimed running time. Then, we apply this algorithm to all instances G_i and recover the solution for the original instance H of Λ -#FACTOR^{Rel}. This procedure takes time

$$\begin{aligned} |H|^{\mathcal{O}(1)} + \sum_i (\text{top } \Lambda + 1 - \varepsilon)^{w_{G_i}} \cdot |G_i|^{\mathcal{O}(1)} &\leq (\text{top } \Lambda + 1 - \varepsilon)^{w_H + f(\Delta^*)} \cdot (|H| \cdot f(\Delta^*))^{\mathcal{O}(1)} \\ &\leq (\text{top } \Lambda + 1 - \varepsilon)^{w_H + f'(\Delta^*)} \cdot |H|^{\mathcal{O}(1)} \end{aligned}$$

for an appropriately chosen function f' .

By Main Lemma 7.2.2 this immediately contradicts MONO-pw-#CSPH and thus, concludes the proof for the finite case.

Cofinite Sets. For ease of notation, We define $\Lambda := \{\max X + 1\}$ and consider an instance H of Λ -#FACTOR that is given with a path decomposition of width w_H —for example, the instance from the previous case. We apply Main Lemma 7.4.6 to obtain $|H|^{\mathcal{O}(\max X)}$ instances of X -#ANTIFACTOR^{Rel}; we denote them by H_i .

For each such instance H_j , we know that $|H_j| \leq \mathcal{O}(|H| \cdot \log^2 |H|)$, the total arity is bounded by $\Delta^*(H_j) \in \mathcal{O}(\max X)$, and the instance H_j is given with a path decomposition of width $w_{H_j} \leq w_H + \mathcal{O}(1)$.

For each instance H_j of X -#ANTIFACTOR^{Rel}, we apply Main Lemmas 7.5.3 and 7.7.14 with $Y = X$ to obtain $|H_j|^{\mathcal{O}(1)} = |H|^{\mathcal{O}(1)}$ instances of X -#ANTIFACTOR; we denote them by G_i .

We observe that $|G_i| \leq |H_j| \cdot f(\Delta^*(H_j) + \max X)$, and that instance G_i is given with a path decomposition of width $w_{G_i} \leq w_{H_j} + f(\Delta^*(H_j) + \max X)$. Together with the bounds for H_j , we know that $|G_i| \leq |H|^{\mathcal{O}(1)}$ and $w_{G_i} \leq w_{H_j} + \mathcal{O}(1)$ as the set X is fixed, and thus, we can treat $\max X$ as constant. Moreover, there are $|H|^{\mathcal{O}(1)}$ instances of X -#ANTIFACTOR.

Now, the remainder of the proof follows precisely as for the previous case when Λ is finite, except that we eventually use the lower bound from the previous case. $\square$

For the last result we parameterize by the size x of the set of forbidden degrees.

If we do not fix the set X but only the *size* of the set, the decision version of ANTIFACTOR _{x} is still fixed-parameter tractable parameterized by treewidth (and x). For #ANTIFACTOR _{x} we show that such algorithms are unlikely.

To prove Main Theorem II.6, that is, the #W[2]-hardness of #ANTIFACTOR _{x} , even if $x = 1$, when parameterizing by pathwidth, we design a new construction. Formally, we prove Main Lemma 7.8.4 in Section 7.8.

Main Lemma 7.8.4. *There exists a constant c such that there is no $\mathcal{O}(n^{w-c})$ algorithm for #ANTIFACTOR^{Rel}₁ on n -vertex graphs with $\Delta^* \in \mathcal{O}(1)$, even if only one set is used and we are given a path decomposition of width w , unless #SETH fails.*

Moreover, #ANTIFACTOR^{Rel}₁ is #W[2]-hard parameterized by pathwidth.

Then, the hardness of $\#ANTIFACTOR_X$ follows from the combination of Main Lemmas 7.5.3, 7.7.14 and 7.8.4.

Main Theorem II.6 (Lower Bound for Counting Version II). *There exists a constant $c > 0$ such that if there is an algorithm that solves $\#ANTIFACTOR_1$ on a given graph G with n vertices with a given path decomposition of width p in time $\mathcal{O}(n^{p-c})$, then $\#SETH$ is false.*

Moreover, $\#ANTIFACTOR_1$ is $\#W[2]$ -hard parameterized by pathwidth.

Proof. Consider an instance H of $\#ANTIFACTOR_1^{\text{Rel}}$ where only a single (possibly not fixed) set $Y \subseteq \mathbb{N}$ with $|Y| = 1$ is assigned to the vertices. Assume that H contains n_H vertices, is given with a path decomposition of width w_H and total arity $\Delta^* \in \mathcal{O}(1)$.

We use Main Lemmas 7.5.3 and 7.7.14 with $X := \{2\}$ to transform H into a polynomial (in n_H) number of instances of $\#ANTIFACTOR_1$, which we denote by G_i . By the choice of H and the properties of the reductions, we get $n_{G_i} \leq n_H \cdot \mathcal{O}(1) \cdot f(\max X)$ as bound on the number of vertices of G and $w_{G_i} \leq w_H + \mathcal{O}(1) \cdot f(\max X)$ as bound on the width of the path decomposition of G . By our choice of X these changes can be bounded by a constant.

Now assume that, for all $c > 0$, there is an algorithm for $\#ANTIFACTOR_1^{\text{Rel}}$ with running time $\mathcal{O}(n_{G_i}^{w_{G_i}-c})$. We apply this algorithm to all instances G_i and recover the solution for H . For some appropriately chosen constant c' this procedure takes time

$$\begin{aligned} n_H^{c'} \cdot \sum_i \mathcal{O}(n_{G_i}^{w_{G_i}-c}) &\leq \mathcal{O}(n_H^{c'+w_H+\mathcal{O}(1)-c}) \\ &\leq \mathcal{O}(n_H^{w_H-(c-\mathcal{O}(1))}) \\ &\leq \mathcal{O}(n_H^{w_H-\hat{c}}). \end{aligned}$$

Hence, for all $\hat{c}$, there is an algorithm solving $\#ANTIFACTOR_1^{\text{Rel}}$ in time $\mathcal{O}(n_H^{w_H-\hat{c}})$. By Main Lemma 7.8.4, this contradicts $\#SETH$ and concludes the proof as $\#ANTIFACTOR_1^{\text{Rel}}$ is also $\#W[2]$ -hard. $\square$

In the following we provide the proofs of the intermediate results.

7.2 Lower Bound for the Intermediate Problem

In this section we prove the lower bound for the intermediate problem $\Lambda\text{-FACTOR}^{\text{Rel}}$ and the corresponding counting version. Formally, we prove Main Lemma 7.2.1 for the decision version.

Main Lemma 7.2.1. *Denote by $\Lambda \subseteq \mathbb{N}$ a fixed, finite set with $\Lambda \neq \{0\}$. Let $f_\Lambda: \mathbb{N} \rightarrow \mathbb{R}^+$ be an arbitrary (computable) function that may depend on the set Λ .*

For all $\varepsilon > 0$, no algorithm can, for a $\Lambda\text{-FACTOR}^{\text{Rel}}$ instance G ,

- that is given with a path decomposition of G of width p ,*
- where the simple vertices of G form an independent set,*
- every simple vertex is adjacent to two complex vertices by $\max \Lambda$ edges each,*

- all simple vertices have degree exactly $\max \Lambda$ in every solution,
- the total arity satisfies $\Delta^* \in \mathcal{O}(\max \Lambda)$,

decide if there is a solution for G in time $(\max \Lambda + 1 - \varepsilon)^{p + f_\Lambda(\Delta^*)} \cdot |G|^{\mathcal{O}(1)}$ unless MONO-pw-CSPH fails.

We state the analogous result for the counting problem Λ -#FACTOR^{Rel} in Main Lemma 7.2.2.

Instead of starting from SAT, which would be similar to known lower bound reductions (see, for example [97]), we start from the q -CSP- B problem as introduced in Section 4.1.2. This allows us to present a simplified construction where most technicalities are already covered by the intermediate lower bound result for the CSP problem. More precisely, we base our results on MONO-pw-CSPH as formalized in Conjecture 4.1.11 and the corresponding version MONO-pw-#CSPH for the counting problem from Conjecture 4.1.12.

For convenience, we restate only Conjecture 4.1.11 here as Conjecture 4.1.12 for the counting version follows naturally.

Conjecture 4.1.11 (MONO-pw-CSPH). *For all $B \geq 2$ and $\varepsilon > 0$, there exists a q such that no algorithm can, for a q -CSP- B instance ϕ that is given with*

- a partition $V_1 \dot{\cup} V_2$ of the variables,
- a nice path decomposition of the primal graph of ϕ of width w
- where every bag contains at most $\mathcal{O}(B \cdot \log |V_1|)$ variables from V_2 ,
- an injective mapping c from the constraints of ϕ to the bags such that the bag contains the variables of the respective constraint,
- the promise that every satisfying multi-assignment that is consistent on V_2 is also consistent on V_1 ,

decide if there is a monotone satisfying multi-assignment that is consistent on V_2 in time

$$(B - \varepsilon)^w \cdot |\phi|^{\mathcal{O}(1)}.$$

For a given set Λ , we set from now on $M := \max \Lambda$. Then, the idea is to provide a parsimonious reduction from q -CSP- $(M + 1)$ to Λ -FACTOR^{Rel}.

Conceptual Idea. Assume we are given a q -CSP- $(M + 1)$ instance ϕ that consists of n variables and m constraints. Additionally, we assume that we are given a nice path decomposition of the primal graph of ϕ of pathwidth $\text{pw}(\phi)$ and a bijection between the constraints and the bags.

We construct the instance of Λ -FACTOR^{Rel} along the path decomposition of ϕ . For each variable $v \in V(\phi)$, we create a path of vertices alternating between complex vertices with a particular relation and (simple) vertices. The state of the simple vertices encodes the assignment of the variable and allows us to change the assignment between two consecutive bags. The complex vertices check whether the assignments encoded by the states of the simple vertices

Note the +1 as the number of “states” is counted differently for q -CSP- B and GENFAC.

satisfy the constraints of the CSP instance. Additionally, the complex vertices ensure that the assignment does not change in an uncontrolled way but only monotonically.

More formally, for each appearance of variable v in a bag $\beta(j)$, we create one complex vertex q_v^j . We connect the two consecutive vertices q_v^j and q_v^{j+1} to a simple vertex t_v^{j+1} by M parallel edges each. Then, the state of a (simple) vertex is determined by the number of selected incident edges to the complex vertex on the right.

In the formal construction, the vertex q_v^j corresponds to the two vertices r_v^j and $\bar{r}_v^j$.

For each constraint C_j of ϕ , we create a complex vertex c_j with relation R^j . We connect vertex c_j to q_v^j for all variables in the scope of the constraint, that is, for all $v \in \text{scp}(C_j)$

When we arrange the vertices in a path-like fashion, which follows naturally from the given path decomposition of ϕ , we create one row for each variable and one column for each bag of the decomposition. As we are given a mapping from the constraints to the bags, this also covers all constraints of ϕ .

We refer to Figure 7.2 for an example of this construction based on the path decomposition of a given q -CSP- $(M+1)$ instance.

Now we move to the formal proof of Main Lemma 7.2.1 and Main Lemma 7.2.2 which we prove simultaneously.

Main Lemma 7.2.1. Denote by $\Lambda \subseteq \mathbb{N}$ a fixed, finite set with $\Lambda \neq \{0\}$. Let $f_\Lambda: \mathbb{N} \rightarrow \mathbb{R}^+$ be an arbitrary (computable) function that may depend on the set Λ .

For all $\varepsilon > 0$, no algorithm can, for a Λ -FACTOR^{Rel} instance G ,

- that is given with a path decomposition of G of width p ,
- where the simple vertices of G form an independent set,
- every simple vertex is adjacent to two complex vertices by $\max \Lambda$ edges each,
- all simple vertices have degree exactly $\max \Lambda$ in every solution,
- the total arity satisfies $\Delta^* \in \mathcal{O}(\max \Lambda)$,

decide if there is a solution for G in time $(\max \Lambda + 1 - \varepsilon)^{p+f_\Lambda(\Delta^*)} \cdot |G|^{\mathcal{O}(1)}$ unless MONO-pw-CSPH fails.

Main Lemma 7.2.2. Denote by $\Lambda \subseteq \mathbb{N}$ a fixed, finite set with $\Lambda \neq \{0\}$. Let $f_\Lambda: \mathbb{N} \rightarrow \mathbb{R}^+$ be an arbitrary (computable) function that may depend on the set Λ .

For all $\varepsilon > 0$, no algorithm can, for a Λ -FACTOR^{Rel} instance G ,

- that is given with a path decomposition of G of width p ,
- where the simple vertices of G form an independent set,
- every simple vertex is adjacent to two complex vertices by $\max \Lambda$ edges each,
- all simple vertices have degree exactly $\max \Lambda$ in every solution,
- the total arity is Δ^* ,

count the number of solution of G in time $(\max \Lambda + 1 - \varepsilon)^{p + f_\Lambda(\Lambda^*)} \cdot |G|^{O(1)}$ unless MONO-pw-#CSPH fails.

Proof of Main Lemmas 7.2.1 and 7.2.2. For ease of notation, we again write $M := \max \Lambda$. We provide a reduction from q -CSP- $(M + 1)$ with the additional properties from Conjecture 4.1.11 to Λ -FACTOR^{Rel}.

Assume we are given an instance ϕ of q -CSP- $(M + 1)$ with n variables $V_1 \dot{\cup} V_2$ and m constraints $C_1, \dots, C_m$. Moreover, assume that we are given a nice path decomposition of the primal graph of ϕ of width $\text{pw}(\phi)$ such that there is a bijection between the bags and the constraints. This is possible by duplicating bags and introducing constraints that observe a single variable and always accept. Hence, we denote the bags of the decomposition by $\beta(1), \dots, \beta(m)$.

Recall, a nice path decomposition might contain dummy nodes; see Section 2.2.

Constructing the Λ -FACTOR^{Rel} Instance. We construct the graph G_ϕ as an instance of Λ -FACTOR^{Rel} as follows.

For a variable $v \in V(\phi)$, we denote by $\text{intro}(v)$ the index of the bag where v is introduced, that is, we write $\text{intro}(v)$ for the unique bag such that $\beta(\text{intro}(v)) \setminus \beta(\text{intro}(v) - 1) = \{v\}$. Similarly, we denote by $\text{forget}(v)$ the index of the bag where v is forgotten, that is, we write $\text{forget}(v)$ for the unique bag such that $\beta(\text{forget}(v) - 1) \setminus \beta(\text{forget}(v)) = \{v\}$.

For easier navigation in the graph, we define a left-hand neighbor and a right-hand neighbor for most of the following vertices. Naturally, when u is a left-hand neighbor of v , we also say that v is a right-hand neighbor of u , and vice versa.

We first define the vertices and edges of the graph. In a second step, we define the relations assigned to the vertices.

This notation formalizes the arrangement of the vertices as depicted in Figure 7.2.

1. For all $j \in [1 .. m]$, we create a complex vertex c_j with relation R^j .
2. For all $v \in V(\phi)$ and $j \in [\text{intro}(v) .. \text{forget}(v)]$, we create two complex vertices $\bar{r}_v^j$ and r_v^j . We assign relation $\bar{R}$ to vertex $\bar{r}_v^j$ and relation R to vertex r_v^j .

If variable v appears in constraint C_j , then we connect c_j by M parallel edges to $\bar{r}_v^j$ and by M parallel edges to r_v^j . We say that c_j is the right-hand neighbor of $\bar{r}_v^j$ and the left-hand neighbor of r_v^j .

Otherwise, we connect $\bar{r}_v^j$ by M parallel edges to r_v^j as the right-hand neighbor.

3. For all $v \in V(\phi)$ and $j \in [\text{intro}(v) .. \text{forget}(v) - 1]$, we create a simple vertex t_v^j . We connect vertex t_v^j by M parallel edges to r_v^j and by M parallel edges to $\bar{r}_v^{j+1}$. We say that t_v^j is the right-hand neighbor of r_v^j and the left-hand neighbor of $\bar{r}_v^{j+1}$.

Note that this does neither create a vertex $t_v^{\text{forget}(v)-1}$ nor $t_v^{\text{forget}(v)}$.

4. For all $v \in V_1$, we create two vertices p_v and $\bar{p}_v$ and assign relation P to both of them. We connect vertex p_v by M parallel edges to $\bar{r}_v^{\text{intro}(v)}$, and refer to it as the right-hand neighbor of p_v . Similarly, we connect vertex $\bar{p}_v$ by M parallel edges to $r_v^{\text{forget}(v)-1}$, and refer to it as the left-hand neighbor of $\bar{p}_v$.

5. Denote by $w_1, \dots, w_{|V_1|}$ an ordering of the vertices in V_1 according to their *introduce* node such that $\text{intro}(w_i)$ comes before $\text{intro}(w_{i'})$ in the given path decomposition if and only if $i < i'$.

This ordering exists as we assume that the decomposition is nice.

Then, for all $i \in [1 \dots |V_1|]$, we add an edge between p_{w_i} and $p_{w_{i+1}}$.

6. Similarly, denote by $\bar{w}, \dots, \bar{w}_{|V_1|}$ an ordering of the vertices in V_1 according to their *forget* node such that $\text{bag forget}(w_i)$ comes before $\text{forget}(w_{i'})$ in the given path decomposition if and only if $i < i'$.

Then, for all $i \in [1 \dots |V_1|]$, we add an edge between $\bar{p}_{\bar{w}_i}$ and $\bar{p}_{\bar{w}_{i+1}}$.

We add an edge between $p_{w_{|V_1|}}$ and $\bar{p}_{\bar{w}_{|V_1|}}$.

7. For all $v \in V_2$, we create a vertex $t_v^{\text{forget}(v)-1}$.

We connect vertex $t_v^{\text{forget}(v)-1}$ by M parallel edges to $\bar{r}_v^{\text{intro}(v)}$, and refer to it as the right-hand neighbor of $t_v^{\text{forget}(v)-1}$. Similarly, we connect vertex $t_v^{\text{forget}(v)-1}$ by M parallel edges to $r_v^{\text{forget}(v)-1}$, and refer to it as the left-hand neighbor of $t_v^{\text{forget}(v)-1}$.

Note, this creates
a cycle for each
variable $v \in V_2$.

8. For all $j \in [1 \dots m]$, if the number of vertices $\bar{r}_\star^j$ is odd, that is, if the number of variables in $\beta(j)$ is odd, then we add a new vertex $\bar{r}_0^j$ with relation $\bar{R}$ and a vertex t_0^j . We make these two vertices adjacent by M parallel edges.

For all $j \in [1 \dots m]$, we connect pairs of the vertices in $\{\bar{r}_0^j\} \cup \{\bar{r}_v^j\}_v$ by an edge. For ease of notation, we assume that we order the vertices in $\{\bar{r}_0^j\} \cup \{\bar{r}_v^j\}_v$ according to their introduce bag and then, connect the vertices based on this order.

Before we define the relations, we first provide some intuition about the construction. The assignment for variable v in bag $\beta(j)$ is determined by the number of selected edges between r_v^j and its right-hand neighbor. More precisely, if k edges between r_v^j and its right-hand neighbor q are selected, then the multi-assignment gives value $k + 1$ to variable v in bag $\beta(j)$.

We have $q = t_v^j$ unless
 $j = \text{forget}(v) - 1$, in
which case $q = \bar{p}_v$.

With this idea we can move to the formal definition of the relations.

- Relation R forwards the information to the right-hand neighbor. Formally, the relation R ensures that k edges to the left-hand neighbor are selected if and only if k edges to the right-hand neighbor are selected.

Moreover, for each k only one possible selection of the $2k$ edges is allowed, that is, for each k we fix an arbitrary set of k edges to the left-hand neighbor and a set of k edges to the right-hand neighbor that are accepted when selected together.

We can think of the
“first” k edges to be
selected. Without
this restriction, the
number of solutions
would increase.

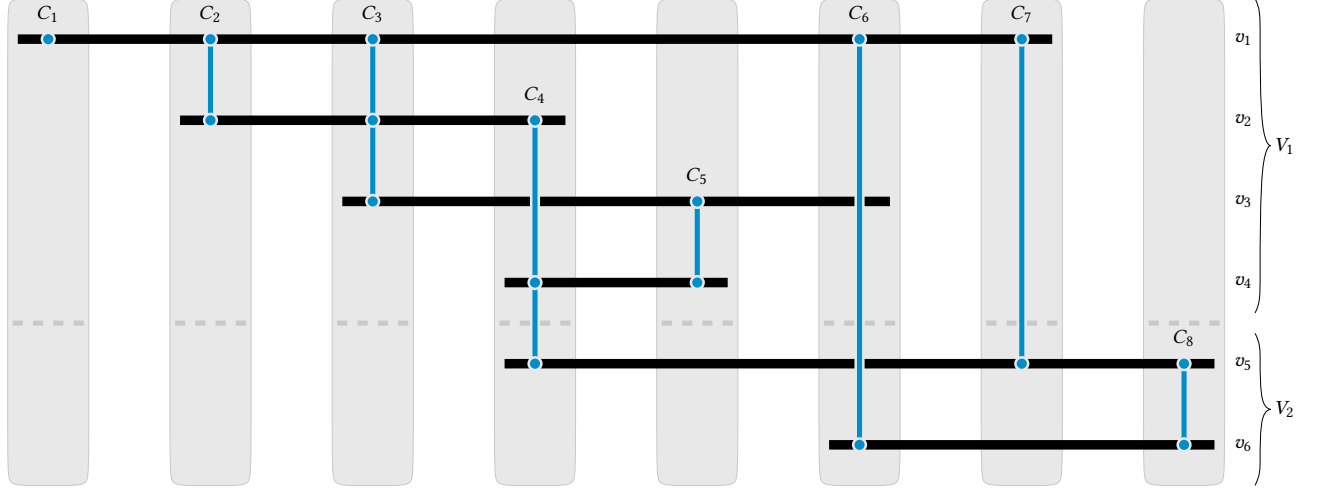
- Relation $\bar{R}$ complements the number of selected edges with respect to M . Formally, relation $\bar{R}$ ensures that exactly k edges to the left-hand neighbor are selected if and only if exactly $M - k$ edges to the right-hand neighbor are selected. That is, exactly M incident edges are selected in total.

Similarly to relation R , for each k , we additionally fix a set of M edges subject to the above constraint that need to be selected.

This ensures that
the parity of the
relation $\bar{R}$ is even.

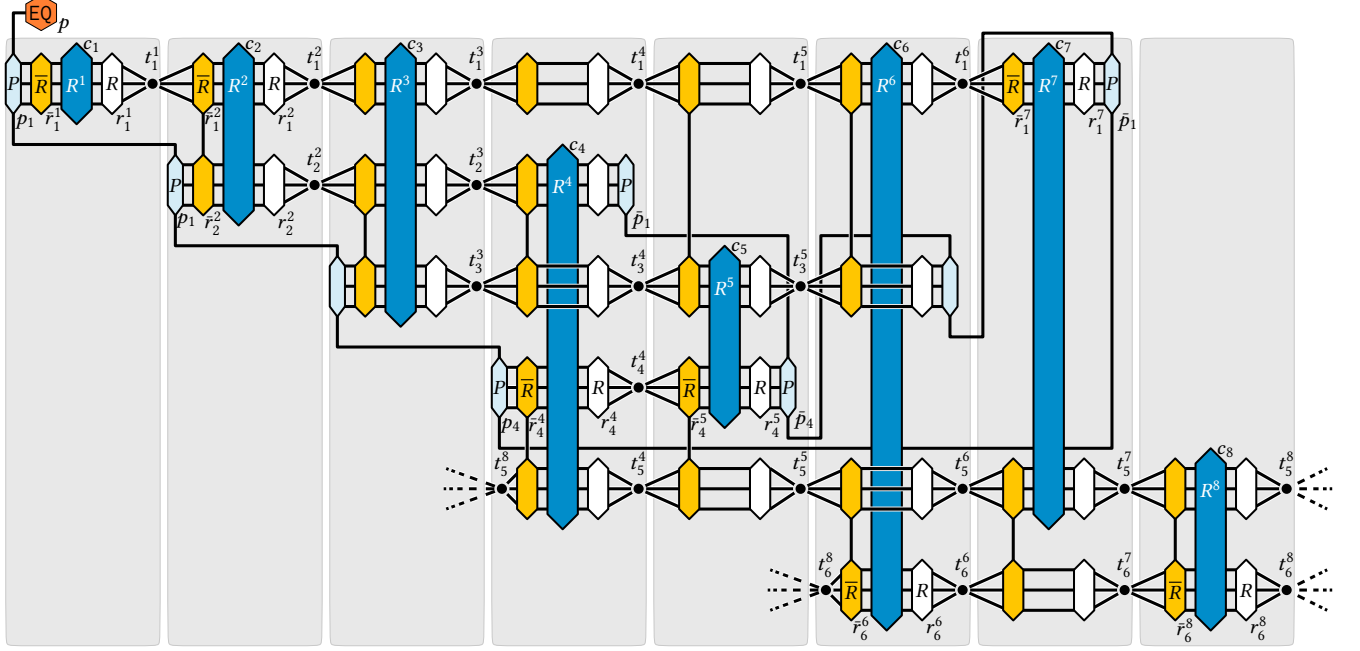
It remains to define the behavior with regard to the edge to another vertex with relation $\bar{R}$. If M is even, then we require that this edge is unselected. Otherwise, when M is odd, the relation $\bar{R}$ requires that this edge is selected.

7.2 Lower Bound for the Intermediate Problem



(a) An illustration of a path decomposition of the CSP-instance ϕ which serves as a starting point of the construction.

The variables are indicated by bold horizontal lines. The constraints are illustrated by vertical lines. A dot indicates that the variable appears in the constraint. The boxes represent the bags of the path decomposition.



(b) An illustration of the constructed graph G_ϕ . The hexagons represent the complex vertices. The gray boxes correspond to the bags of the formula ϕ .

Figure 7.2: An illustration of the reduction from an instance ϕ of q -CSP-($\max \Lambda + 1$) (displayed on the top) to an instance G_ϕ of Λ -FACTOR^{Rel} (displayed below) according to the proof of Main Lemma 7.2.1.

Here we assume that $\max \Lambda + 1 = 4$ and the formula ϕ has 4 variables in V_1 , 2 variables in V_2 , and $m = 8$ constraints.

- The relation R^j mimics the constraint C_j .

For all $v \in \text{scp}(C_j)$, the relation R^j ensures that the number of selected edges going to r_v^j is equal to the number of selected edges going to $\bar{r}_v^j$. We denote the number of selected edges to r_v^j by $\pi_j(v) \in [0 .. M]$ such that $\pi_j(v) + 1$ corresponds to an assignment for variable v in bag $\beta(j)$.

Moreover, the relation R^j ensures that only those edge selections are accepted where the encoded assignment $\pi_j + 1$ satisfies the constraint C_j .

- The relation P ensures that the number of selected incident edges is even. Apart from this constraint, the relation accepts all possible combinations of edge selections.

Finally, we add one more complex vertex p with relation EQ_1 to the graph and make it adjacent to vertex p_{w_1} . As the relation P only allows solutions with an even number of selected edges, this vertex ensures that the number of selected edges to the vertices $\bar{r}_v^{\text{intro}(v)}$ and $r_v^{\text{forget}(v)-1}$ might also be odd. We denote the resulting graph by G_ϕ .

Correctness. In the following, we first show the correctness of the construction. Formally, we prove Claims 7.2.2.1 and 7.2.2.2. Then, in Claim 7.2.2.3, we analyze the remaining properties of the graph (that is, especially the size and pathwidth) and provide a path decomposition of the instance. Finally, we combine the results and conclude the proof.

We start with the forward direction of the correctness.

Claim 7.2.2.1. *Every satisfying multi-assignment for ϕ that is consistent on V_2 can be extended to a solution for G_ϕ .*

Proof of Claim. Let σ be a satisfying multi-assignment for ϕ that is consistent on V_2 . By assumption about V_2 , the multi-assignment σ is also consistent on V_1 and thus, is an assignment.

For fixed node $j \in [1 .. m]$, we denote by $\sigma_j: \beta(j) \rightarrow [1 .. M + 1]$ the restriction of σ to the vertices in $\beta(j)$.

For the solution for G_ϕ , we select the edges as follows.

- For all $v \in V(\phi)$ and all $j \in [\text{intro}(v) .. \text{forget}(v)]$, we select $\sigma_j(v) - 1$ edges to the left-hand neighbor of r_v^j and $\sigma_j(v) - 1$ edges to the right-hand neighbor of r_v^j .

Moreover, we select $\sigma_j(v) - 1$ edges to the right-hand neighbor of $\bar{r}_v^j$ and $(M + 1) - (\sigma_j(v) - 1) = M - \sigma_j(v)$ edges to the left-hand neighbor of $\bar{r}_v^j$.

- We select the edges between the vertices with relation $\bar{R}$ such that each vertex $\bar{r}_v^j$ is incident to an even number of selected edges. That is, if M is odd, then we select the edge, and otherwise we do not select the edge.
- We select the edges between the vertices in $\{p_v, \bar{p}_v\}$ such that each vertex is incident to an even number of edges. Hence, we might also select the edge to the vertex p with relation EQ_1 .

We first observe that each simple vertex t_v^j is incident to M selected edges. This also holds for the vertices corresponding to the variables in $v \in V_2$ as the edge selection does not change for the different bags. Moreover, the relations R , $\bar{R}$, and P are satisfied by our choice of the edges according to their definition and because of vertex p .

Now, for all $j \in [1..m]$, consider the relation R^j that is associated to constraint C_j . In this case, vertex c_j is adjacent to the vertices r_v^j and $\bar{r}_v^j$ for all $v \in \text{scp}(C_j)$.

Our edge selection directly corresponds to the assignment σ_j . Since σ satisfies ϕ , the assignment especially satisfies constraint C_j and thus, the relation R^j is satisfied as well.

We conclude that all relations are satisfied and the claim follows. $\triangleleft$

As a next step, we prove the reverse direction of the correctness.

Claim 7.2.2.2. *For every solution for G_ϕ , there is a unique monotone satisfying multi-assignment for ϕ that is consistent on V_2 .*

Proof of Claim. Consider a solution for G_ϕ . For every variable $v \in V(\phi)$, and every node $j \in [\text{intro}(v) .. \text{forget}(v)]$, we denote the number of selected edges from r_v^j to the right-hand neighbor by x_v^j , and denote the number of selected edges from $\bar{r}_v^j$ to the left-hand neighbor by $\bar{x}_v^j$.

With this notation, we can define the multi-assignment σ for ϕ . For each node $j \in [1..m]$, we define the assignment $\sigma_j: \beta(j) \rightarrow [1..M+1]$ with $\sigma_j(v) := x_v^j + 1$ for all $v \in \beta(j)$. When we combine these assignments σ_j for the different bags, we obtain the multi-assignment σ for ϕ .

We claim that σ is a satisfying multi-assignment for ϕ that is monotone on the variables in V_1 and consistent on V_2 . We start with the monotonicity and consistency.

By the construction of the graph, the choice of $M = \max \Lambda$, and the definition of the relations R and $\bar{R}$, we get

$$\begin{aligned} \bar{x}_v^j + x_v^j &= M & \text{for all } j \in [\text{intro}(v) .. \text{forget}(v)] \text{ and} \\ x_v^j + \bar{x}_v^{j+1} &\leq M & \text{for all } j \in [\text{intro}(v) .. \text{forget}(v) - 1]. \end{aligned}$$

Combining both observations, we get

$$x_v^j \leq x_v^{j+1} \quad \text{for all } j \in [\text{intro}(v) .. \text{forget}(v) - 1] \quad (7.2.1)$$

which proves that the assignment is monotone on all variables.

For the consistency on V_2 , we fix a variable $v \in V_2$. We observe that the vertex $t_v^{\text{forget}(v)-1}$ is connected to $\bar{r}_v^{\text{intro}(v)}$ and $r_v^{\text{forget}(v)-1}$. Thus, we additionally have $x_v^{\text{forget}(v)-1} + \bar{x}_v^{\text{intro}(v)} \leq M$. Combining this with the monotonicity from Equation (7.2.1), we obtain

$$x_v^{\text{intro}(v)} \leq \dots \leq x_v^{\text{forget}(v)-1} \leq M - \bar{x}_v^{\text{intro}(v)} = x_v^{\text{intro}(v)}.$$

This proves that the multi-assignment σ is consistent for all variables in V_2 by the definition of σ based on σ_j .

It remains to argue that σ satisfies ϕ . For this, we fix some arbitrary constraint C_j of ϕ . As we are given a solution for G_ϕ , the relation R^j must be satisfied. By the definition of R^j , the

values $\{x_v^j + 1\}_{v \in \text{scp}(C_j)}$ correspond to an assignment that satisfies the constraint C_j . As we choose precisely this assignment for σ_j , we conclude that σ satisfies ϕ . $\triangleleft$

As a next step, we analyze the pathwidth of our construction and bound the arity of the relations.

Claim 7.2.2.3. *The size of G_ϕ is bounded by $M \cdot |\phi|^{\mathcal{O}(1)}$.*

The degree of the simple vertices is exactly $2M$ and the arity of the complex vertices is bounded by $2M \cdot q$.

We can efficiently construct a path decomposition of width $\text{pw}(\phi) + \mathcal{O}(M \cdot \log|V_1|)$ and total arity $\Delta^* \in \mathcal{O}(M \cdot q)$.

Proof of Claim. For each variable v and each bag j where this variable appears in, we create at most 3 vertices (that is, the vertices t_v^j , $\bar{r}_v^j$ and r_v^j). Additionally, for the vertices in V_1 , we create p_v and $\bar{p}_v$. Hence, the instance consists of $|V(\phi)| \cdot r \cdot \mathcal{O}(1)$ vertices where we observe that the number of bags r is bounded by a function polynomial in the size of ϕ because of the definition of the primal graph.

We observe that the simple vertices are incident to exactly $2M$ edges. The complex vertices p_v , $\bar{p}_v$, $\bar{r}_v^j$, and r_v^j have arity at most $2M + 2$. The m vertices c_j corresponding to the relations have arity $2M \cdot q$. This concludes the size bound on ϕ .

We provide a mixed search strategy which then immediately yields a path decomposition of the graph G_ϕ , as no vertex gets “recontaminated”. See Section 2.2 and [36] for further details.

For all $j \in [1..m]$, we define $w^j \in V_1$ as the variable introduced at node j or as the last variable from V_1 that got introduced. That is, we define w^j as the variable v that maximizes $\text{intro}(v) \leq j$. Similarly, we define $\bar{w}^j$ as the last variable forgotten among the bags $\beta(1), \dots, \beta(j)$. That is, we define $\bar{w}^j$ as the variable $v \in V_1$ that maximizes $\text{forget}(v) - 1 \leq j$. If such a variable does not exist, we define $\bar{w}^j$ to store no variable.

We start by placing searchers on the vertices $\{t_v^1\}_{v \in \beta(1)}$ and on w^1 and $\bar{w}^1$.

We place a searcher on p and remove it again. Then, we place one searcher on the four vertices c_1 , p_{w^1} , $r_{w^1}^1$, and $\bar{r}_{w^1}^1$ and remove them again.

Then, we clean the rest of the graph in stages where we start with stage 2. Stage j starts if the searchers are placed on the vertices $\{t_v^{j-1}\}_{v \in \beta(j-1)} \cup \{p_{w^{j-1}}, \bar{p}_{\bar{w}^{j-1}}\}$. We place two searchers on p_{w^j} and $\bar{p}_{\bar{w}^j}$. For all $v \in V_2$, we place a searcher on $t_v^{\text{forget}(v)-1}$, and we place one searcher on c_j . Now we iterate over the vertices $v \in \beta(j)$ given by the ordering in which we connected the vertices $\bar{r}_v^j$ by an edge. That is, we always consider two vertices $u, v \in \beta(j)$ together. We place two searchers on t_v^j and t_u^j and two searchers on r_v^j and on $\bar{r}_v^j$. Then, we remove the searcher from r_v^j and place it on $\bar{r}_u^j$. Then, we remove the searcher from $\bar{r}_v^j$ and place it on r_u^j . Then, we remove the two searchers from r_v^j and $\bar{r}_v^j$. We remove the searcher from t_v^{j-1} . We repeat this process for the variable u .

By assumption, each bag of the primal graph of ϕ contains at most $\mathcal{O}((M+1) \cdot \log|V_1|)$ vertices from V_2 . Hence, the pathwidth of the instance is bounded by

$$(\text{pw}(\phi) + 1) + \mathcal{O}(1) + \mathcal{O}(M \cdot \log|V_1|).$$

We observe that each bag contains at most one vertex c_j and at most 6 other complex vertices. Hence, the total arity of a bag is bounded by $\mathcal{O}(M \cdot q)$. $\triangleleft$

This “extends”
the orderings
 $w_1, \dots, w_{|V_1|}$ and
 $\bar{w}_1, \dots, \bar{w}_{|V_1|}$ by
introducing duplicates.

Note that we keep
the searcher on
 $t_v^{\text{forget}(v)-1}$ until we
forget the variable v .
This is needed to
avoid that vertices
get recontaminated
as the vertices
associated to the
variables in V_2 form
a cycle in the graph.

Before we conclude the proof, we need to ensure that the parity of all relations is even.

Adjusting the Parity of the Relations. By definition, the relation P has even parity. Similarly, the relations R and R^j have even parity as the number of selected edges on the left and the right side are equal, that is, the vertices with relation R or R^j are always incident to exactly $2k$ selected edges for some k . For the vertices with relation $\bar{R}$, we recall that we connect two such vertices to ensure that they have even parity in case M is odd. Hence, only the vertex p with relation EQ_1 has an undefined parity.

We denote by $\#\text{Sol}_0(G_\phi)$ the number of solutions of G_ϕ where vertex p is *not incident* to a selected edge. Similarly, we denote by $\#\text{Sol}_1(G_\phi)$ the number of solutions of G_ϕ where vertex p is *incident* to a selected edge. It directly follows that $\#\text{Sol}(G_\phi) = \#\text{Sol}_0(G_\phi) + \#\text{Sol}_1(G_\phi)$. Therefore, it suffices to compute these values independently and then combine them as the final result.

To compute the value $\#\text{Sol}_0(G_\phi)$, we can simply delete the vertex p from G_ϕ to obtain the instance G'_ϕ . By our previous arguments, all relations in G'_ϕ now have even parity. Moreover, as the remaining graph does not change, we have $\#\text{Sol}(G'_\phi) = \#\text{Sol}_0(G_\phi)$.

To compute the value $\#\text{Sol}_1(G_\phi)$, we use a different approach. Note that even if we force the edge incident to p to be selected, the parity of the corresponding relation would be odd. Instead, we duplicate the entire graph G_ϕ , merge the two copies of the vertex p into a vertex denoted by q , and assign relation $\text{HW}_{=2}^{(2)}$ to q . We denote the resulting instance by G''_ϕ . We observe that all relations have even parity and that the graph G''_ϕ satisfies the bounds from Claim 7.2.2.3. By the relation $\text{HW}_{=2}^{(2)}$, the behavior of vertex q is fixed. Hence, we obtain $\#\text{Sol}(G''_\phi) = \#\text{Sol}_1(G_\phi)^2$ as the solutions for the two copies are independent of each other.

We conclude that once we can compute $\#\text{Sol}(G'_\phi)$ and $\#\text{Sol}(G''_\phi)$, we can compute

$$\#\text{Sol}(G_\phi) = \#\text{Sol}(G'_\phi) + \sqrt{\#\text{Sol}(G''_\phi)}$$

as the number of solutions for the original instance G_ϕ .

Concluding the Proof. Now we have everything ready to prove the lower bound for the intermediate problem $\Lambda\text{-FACTOR}^{\text{Rel}}$ based on the previous construction. Recall, we define the total arity Δ^* as the maximum sum over the arities of the complex vertices appearing in one bag, that is, we define $\Delta^* := \max_{\text{bag } \beta} \sum_{v \in \beta \cap V_C} \deg(v)$.

Assume that for some $\varepsilon > 0$, there is an algorithm $\mathcal{A}$ that, given an instance G along with a path decomposition of width $\text{pw}(G)$, solves $\Lambda\text{-FACTOR}^{\text{Rel}}$ (resp. $\Lambda\text{-\#FACTOR}^{\text{Rel}}$ for the counting version) in time $(M+1-\varepsilon)^{\text{pw}(G)+f(\Delta^*)} \cdot |G|^{\mathcal{O}(1)}$ for some function f .

We denote by q the value from Conjecture 4.1.11 (resp. Conjecture 4.1.12 for the counting version) such that $q\text{-CSP-}(M+1)$ cannot be solved in time $(M+1-\varepsilon)^{\text{pw}} \cdot |\psi|^{\mathcal{O}(1)}$ for a formula ψ .

Now, consider a $q\text{-CSP-}(M+1)$ instance ϕ with the additional properties as stated above, that is, the properties specified in Conjecture 4.1.11 (resp. in Conjecture 4.1.12 for the counting version) we construct the instance G_ϕ and derive the two instances G'_ϕ and G''_ϕ from it.

For example, we can revert the path decomposition of the first copy of G_ϕ , or add q to all bags corresponding to the first copy of G_ϕ and append the decomposition of the second copy.

When we apply the algorithm $\mathcal{A}$ on these two instances, we can compute the result for G_ϕ and thus, also for ϕ by Claims 7.2.2.1 and 7.2.2.2. Including the construction of G_ϕ , the entire algorithm operates in time

$$|\phi|^{\mathcal{O}(1)} + 2 \cdot (M + 1 - \varepsilon)^{\text{pw}(G_\phi) + f(\Delta^*(G_\phi))} \cdot |G_\phi|^{\mathcal{O}(1)}.$$

With Claim 7.2.2.3, we can bound the running time by

$$\begin{aligned} & (M + 1 - \varepsilon)^{\text{pw}(\phi) + \mathcal{O}(M \cdot \log |V_1|) + f(M \cdot q)} \cdot |G_\phi|^{\mathcal{O}(1)} \\ & \leq (M + 1 - \varepsilon)^{\text{pw}(\phi)} \cdot (M + 1)^{\mathcal{O}(M \cdot \log |V_1|) + f(M \cdot q)} \cdot |G_\phi|^{\mathcal{O}(1)} \\ & \leq (M + 1 - \varepsilon)^{\text{pw}(\phi)} \cdot |V_1|^{\mathcal{O}(M \cdot \log M)} \cdot (M + 1)^{f(M \cdot q)} \cdot |G_\phi|^{\mathcal{O}(1)}. \end{aligned}$$

As q only depends on M and ε and M only depends on Λ which is fixed by assumption, we can hide the corresponding terms by the polynomial term $|\phi|^{\mathcal{O}(1)}$. Thus, the running time is bounded by

$$(M + 1 - \varepsilon)^{\text{pw}(\phi)} \cdot |G_\phi|^{\mathcal{O}(1)}.$$

Now we observe that this running time violates MONO-pw-CSPH (resp. MONO-pw-#CSPH for the counting version) which concludes the proof. $\square$

7.3 Decision Version: Extension to Cofinite Sets

In this section, we consider the intermediate problem Λ -FACTOR^{Rel} when the set Λ is cofinite, that is, we consider X -ANTIFACTOR^{Rel} for a finite set X . To eventually establish the conditional lower bound for X -ANTIFACTOR, we prove, for some appropriately chosen set Λ , a reduction from Λ -FACTOR^{Rel} with the restrictions from Main Lemma 7.2.1 to X -ANTIFACTOR^{Rel}. Formally, we prove the reduction from Main Lemma 7.3.1.

Main Lemma 7.3.1. *Fix a finite, non-empty set $X \subseteq \mathbb{N}$ which contains a half-induced matching of size $h \geq 2$ with U as largest value.*

For every $\Lambda \subseteq \mathbb{N}$ with $\max \Lambda = h - 1$, there is a pathwidth-preserving many-one reduction from Λ -FACTOR^{Rel} with the restrictions from Main Lemma 7.2.1 to X -ANTIFACTOR^{Rel} such that the total arity increase by a factor of U , and the size by a factor of 2^U .

To prove the reduction from Λ -FACTOR^{Rel} to X -ANTIFACTOR^{Rel}, it suffices to replace the simple vertices with set Λ by appropriate gadgets consisting of simple vertices with set X and complex vertices.

However, we design the reduction not in a black-box way but use the information about the instance which we constructed to prove the lower bound for Λ -FACTOR^{Rel}. Phrased differently, we use the additional assumptions provided in Main Lemma 7.2.1.

Main Lemma 7.2.1. *Denote by $\Lambda \subseteq \mathbb{N}$ a fixed, finite set with $\Lambda \neq \{0\}$. Let $f_\Lambda: \mathbb{N} \rightarrow \mathbb{R}^+$ be an arbitrary (computable) function that may depend on the set Λ .*

For all $\varepsilon > 0$, no algorithm can, for a Λ -FACTOR^{Rel} instance G ,

- *that is given with a path decomposition of G of width p ,*

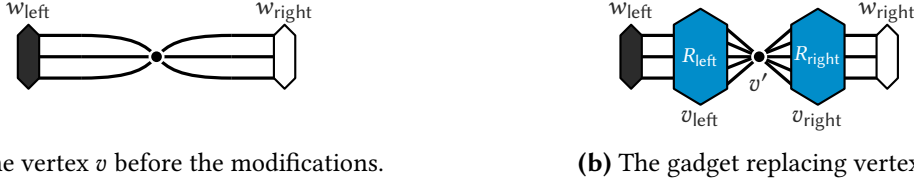


Figure 7.3: In Main Lemma 7.3.1 we replace each simple vertex v with set Λ by a gadget consisting of two complex vertices v_{left} and v_{right} and the simple vertex v' with set X based on a half-induced matching of size h and largest value U . Here shown for $h - 1 = \max \Lambda = 3$ and $U = 5$.

- where the simple vertices of G form an independent set,
- every simple vertex is adjacent to two complex vertices by $\max \Lambda$ edges each,
- all simple vertices have degree exactly $\max \Lambda$ in every solution,
- the total arity satisfies $\Delta^* \in \mathcal{O}(\max \Lambda)$,

decide if there is a solution for G in time $(\max \Lambda + 1 - \varepsilon)^{p + f_{\Lambda}(\Delta^*)} \cdot |G|^{\mathcal{O}(1)}$ unless MONO-pw-CSPH fails.

Our goal is to exploit the structure of how the information vertices propagate information from one side to the other side and relate this to the half-induced matching in the compatibility graph as introduced in Section 4.2.

More formally, we replace each simple vertex by a small gadget consisting of three parts; one simple vertex and two complex vertices. The simple vertex is connected to both complex vertices by sufficiently many edges, and each complex vertex is connected to one half of the original edges, depending on their other end point.

The two complex vertices “translate” between the encoded information and the half-induced matching while the simple vertex ensures the properties of the half-induced matching.

With this intuition in mind, we move to the formal proof of the reduction. See Figure 7.3 for an illustration of the construction.

Main Lemma 7.3.1. Fix a finite, non-empty set $X \subseteq \mathbb{N}$ which contains a half-induced matching of size $h \geq 2$ with U as largest value.

For every $\Lambda \subseteq \mathbb{N}$ with $\max \Lambda = h - 1$, there is a pathwidth-preserving many-one reduction from $\Lambda\text{-FACTOR}^{\text{Rel}}$ with the restrictions from Main Lemma 7.2.1 to $X\text{-ANTIFACTOR}^{\text{Rel}}$ such that the total arity increase by a factor of U , and the size by a factor of 2^U .

Proof. Assume that the compatibility graph $\mathcal{C}_X$ contains a half-induced matching of size h on the h vertices $a_0, \dots, a_{h-1}$ on one side and the h vertices $b_0, \dots, b_{h-1}$ on the other side. We denote by U the maximum value appearing in this half-induced matching.

We denote by $\Lambda \subseteq \mathbb{N}$ an arbitrary but fixed, finite set with $\max \Lambda = h - 1$. Consider an instance H of $\Lambda\text{-FACTOR}^{\text{Rel}}$ with the properties stated in Main Lemma 7.2.1. It suffices to handle the simple vertices of H as the complex vertices do not need to change. Still, we later adjust the complex vertices as the degree of the adjacent simple vertices changes.

Fix a simple vertex $v \in V_S(H)$. By assumption, this vertex v is incident to exactly $2 \max \Lambda = 2(h - 1)$ edges. We can partition these edges into two sets of size exactly $h - 1$ depending on their endpoints. We denote these sets of edges by $E_{v,\text{left}}$ and $E_{v,\text{right}}$ and refer to them as *left-external* and *right-external* edges.

We remove the vertex v and its incident edges and create the following gadget.

- We create a simple vertex v' which is assigned set X .
Moreover, we create two new complex vertices v_{left} and v_{right} with relations R_{left} and R_{right} , respectively, which are defined shortly.
- We connect the left-external edges to vertex v_{left} and the right-external edges to vertex v_{right} . That is, for all edges $uv \in E_{\text{left}}$, we create the edge uv_{left} , and, for all edges $uv \in E_{\text{right}}$, we create the edge uv_{right} .
- We connect vertex v' by U (parallel) edges to v_{left} and call these edges the left-internal edges. Additionally, we connect vertex v' by U parallel edges to v_{right} and call these edges the right-internal edges.
- The relation R_{left} accepts if and only if, there exists an $i \in [0 .. h)$ such that exactly i left-external edges and exactly a_{h-1-i} left-internal edges are selected.
- Similarly, the relation R_{right} accepts if and only if there exists a $j \in [0 .. h)$ such that exactly b_j right-internal edges and exactly j right-external edges are selected.

As a last step, we merge the newly created complex vertices with the existing complex vertices. For this, we observe that, by Main Lemma 7.2.1, every simple vertex v is only connected to two complex vertices in H , say w_{left} and w_{right} .

For each simple vertex of H , we merge the two complex vertices v_{left} and w_{left} into a new vertex as follows. We create the new complex vertex $\bar{w}_{\text{left}}$ and connect it to all vertices (potentially by multiple parallel edges according to the original adjacency) in $N(\{w_{\text{left}}, v_{\text{left}}\})$. The relation of this new vertex $\bar{w}_{\text{left}}$ is precisely the combination of the relation R_{left} with the relation of the vertex w_{left} . Note that we can exploit that R_{left} allows at most one selection of left-internal edges based on a given selection of left-external edges. Thus, the relation can be adjusted by a linear pass over all accepted inputs. Finally, we delete the two complex vertices w_{left} and v_{left} .

Analogously we can merge the two complex vertices w_{right} and v_{right} into a new vertex $\bar{w}_{\text{right}}$.

We repeat this process for all other simple vertices and denote the resulting instance of $X\text{-ANTI}\text{FACTOR}^{\text{Rel}}$ by G .

It remains to show that the original instance H has a solution if and only if the new instance G has a solution. As a first step we prove the forward direction.

Claim 7.3.1.1. *If H has a solution, then G has a solution.*

Proof of Claim. Consider an arbitrary solution $S \subseteq E(H)$ for H . We consider each simple vertex v of H separately and modify the solution S to obtain a solution S' for G .

By Main Lemma 7.2.1, vertex v is incident to exactly $h - 1$ selected edges in the solution S . Hence, assume that vertex v is incident to i selected left-external edges and hence, vertex v is

incident to $h - 1 - i$ selected right-external edges. For the new solution S' we select a_{h-1-i} left-internal edges and b_{h-1-i} right-internal edges.

By this selection, the new simple vertex v' is incident to $a_{h-1-i} + b_{h-1-i}$ selected edges. By the properties of the half-induced matching (see Definition 4.2.1 for further details), this value is not contained in the set X . Hence, vertex v' is incident to a valid number of selected edges in the new solution. $\triangleleft$

As a next step, we prove the reverse direction of the correctness.

Claim 7.3.1.2. *If G has a solution, then H has a solution.*

Proof of Claim. Consider an arbitrary solution $S' \subseteq E(G)$ for the instance G . We first analyze the solution S' . We fix a simple vertex v' of G and its corresponding vertex v in H .

By the definition of the relation R_{left} , vertex v' must be incident to a_{h-1-i} selected left-internal edges for some $i \in [0 .. h)$. Similarly, by the definition of the relation R_{right} , vertex v' must be incident to b_j selected right-internal edges for some $j \in [0 .. h)$.

By the definition of the half-induced matching for the compatibility graph $\mathcal{C}(X)$, we know that $a_{h-1-i} + b_j \in X$ whenever $j > h - 1 - i$. Thus, the number of selected right-internal edges corresponds to b_j with $h - 1 - i \geq j \geq 0$ and hence, we have $i + j \leq h - 1$.

By the modifications of the relations above, that is, the combination of the complex vertices v_{left} and w_{left} into one new vertex, we know that the number of selected left-internal edges in S' must correspond to some accepted input of the original relation R_{left} of H . Similarly, this holds for the right-internal edges of vertex v' .

Moreover, by the properties of the instance H from Main Lemma 7.2.1, we know that in every solution for H the simple vertices are incident to exactly $h - 1$ selected edges. Hence, the selected internal edges correspond to exactly $i + j = h - 1$ selected external edges.

Thus, we can define a new solution S where we select, for vertex v , exactly i left-external edges and exactly j right-external edges. It immediately follows from the previous observations that S is a solution for H . $\triangleleft$

As a last step we analyze the pathwidth and size of the instance G .

Claim 7.3.1.3. *The new instance G satisfies the following three conditions.*

1. *The pathwidth of G is equal to the pathwidth of H .*
2. *The total arity of G is bounded by $\Delta^*(G) \leq \Delta^*(H) \cdot U$.*
3. *The size of G is bounded by $|G| \leq |H| \cdot 2^U$.*

Proof of Claim. As we merge the complex vertices in a final step, the pathwidth of the graph does not change.

However, merging the relations increases the degree of the complex vertices. Hence, the total arity increases from $\Delta^*(G')$ to at most $\Delta^*(H) \cdot U$ where U denotes the largest value from the half-induced matching. Additionally, this also increases the size of the instance, as the arity of the new relations increases by at most U and thus, the truth-table increases by a factor of at most 2^U . $\triangleleft$

This concludes the reduction from $\Lambda\text{-FACTOR}^{\text{Rel}}$ to $X\text{-ANTI}\text{FACTOR}^{\text{Rel}}$ by the combination of Claims 7.3.1.1 to 7.3.1.3 and the observation that U only depends on the set X and hence, can be bounded by $f(\max X)$ for some function f . $\square$

7.4 Counting Version: Extension to Cofinite Sets

In this section we consider the counting problem $X\text{-}\#\text{ANTI}\text{FACTOR}^{\text{Rel}}$. Recall from Section 7.3 that we extend the lower bound for $\Lambda\text{-FACTOR}^{\text{Rel}}$ to $X\text{-ANTI}\text{FACTOR}^{\text{Rel}}$, for some appropriately chosen set Λ . For the counting version we present a similar reduction.

However, we do not start from $\Lambda\text{-}\#\text{FACTOR}^{\text{Rel}}$ but from $\Lambda\text{-}\#\text{FACTOR}$, that is, we start from the version *without* relations. Formally, we prove Main Lemma 7.4.6.

Main Lemma 7.4.6. *Fix a finite, non-empty set $X \subseteq \mathbb{N}$.*

There is a pathwidth-preserving reduction starting with an instance H of $\{\max X + 1\}\text{-}\#\text{FACTOR}$ and queries $|H|^{\mathcal{O}(\max X)}$ instances of $X\text{-}\#\text{ANTI}\text{FACTOR}^{\text{Rel}}$ where the size is bounded by $\mathcal{O}(|H| \cdot \log^2 |H|)$, and the total arity Δ^* is bounded by $\mathcal{O}(\max X)$.

Before we prove Main Lemma 7.4.6 in the remainder of this section, we first remind ourselves of the ideas for the lower bound of the *decision* version. In that setting we exploit that every simple vertex has only complex vertices as neighbors and the incident edges can be partitioned accordingly. Moreover, we use half-induced matchings to prove the bound (see Section 7.3 for further details). Phrased differently, the construction heavily depends on the construction for the intermediate lower bound for $\Lambda\text{-FACTOR}^{\text{Rel}}$.

Now, for the counting version, we use a completely different approach that is independent of the result for the decision version.

Idea of the Proof. The main observation is that we start from $\Lambda\text{-}\#\text{FACTOR}$ as the version *without relations* and do not require any specific properties of the input instance. Moreover, by using interpolation techniques, we can avoid the dependence on half-induced matchings and can prove a tight lower bound compared to the running time of the algorithm from Main Theorem II.1.

The main idea of this construction is to attach gadgets to the simple vertices without altering the remaining graph structure. As these gadgets use edge-weights and vertex-weights, we gradually modify these gadgets to finally arrive at instances of $X\text{-}\#\text{ANTI}\text{FACTOR}^{\text{Rel}}$.

We split the reduction into multiple steps. We refer to Figure 7.4 for an overview of the general outline.

1. In Lemma 7.4.1 we transition from a finite set of allowed degrees to a cofinite set of allowed degrees.

For a set Y , we define $Y\text{-}\#\text{MIN-ANTI}\text{FACTOR}$ as the variant of $Y\text{-}\#\text{ANTI}\text{FACTOR}$ where we only count solutions of *minimum* size.

Then, formally, this step is a reduction from $\{x + 1\}\text{-}\#\text{FACTOR}$ to $[0 \dots x]\text{-}\#\text{MIN-ANTI}\text{FACTOR}$ with $x = \max X \geq 0$.

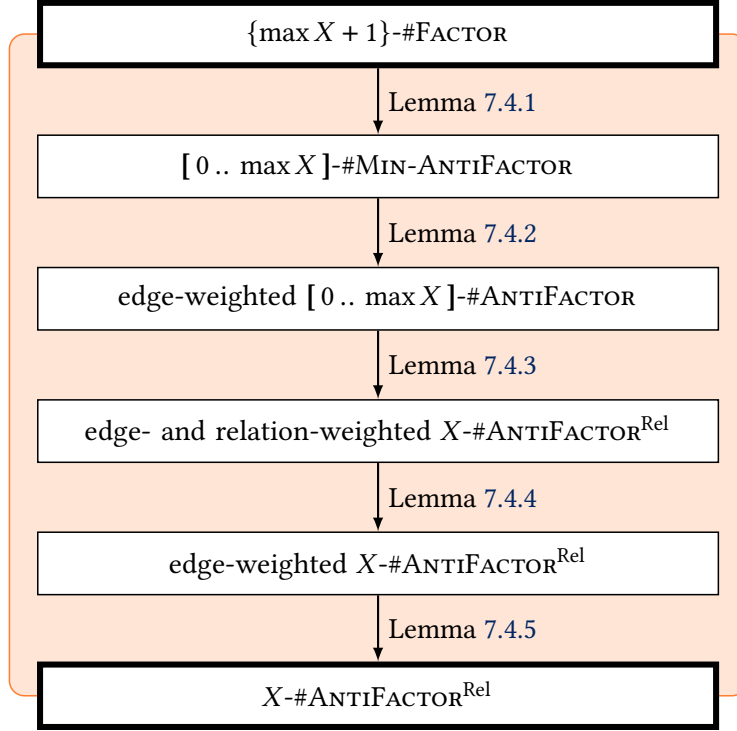


Figure 7.4: The chain of reductions used in the proof of Main Lemma 7.4.6, that is, the reduction from $\{\max X + 1\}$ -#FACTOR to X -#ANTIFACTOR^{Rel} for a finite, non-empty set $X \subseteq \mathbb{N}$.

2. In Lemma 7.4.2 we remove the requirement that we only count solutions of minimum size.

For a set Y , the edge-weighted version of Y -#MIN-ANTIFACTOR is the variant of Y -#MIN-ANTIFACTOR where the edges are assigned weights. Then, the weight of a solution is the *product* of the weights of the selected edges. The output of the problem is the *sum* of all weighted solutions. The unweighted problem can be seen as assigning weight 1 to all edges.

Then, formally, we use interpolation techniques to reduce from $[0 .. x]$ -#MIN-ANTIFACTOR to the *edge-weighted* $[0 .. x]$ -#ANTIFACTOR problem.

3. In Lemma 7.4.3 we transition from *simple* cofinite sets of allowed degrees to *arbitrary* cofinite sets of allowed degrees by introducing weighted relations.

For *edge- and relation-weighted* X -#ANTIFACTOR^{Rel} each accepted input of a relation is assigned a (rational) weight. Then, an accepted input contributes by this weight to the solution. The weight of the solution is the *product* of the weights of the relations multiplied by the weights of the selected edges. The problem without relation-weights can be obtained by assigning weight 1 to all accepted inputs and weight 0 to all rejected inputs.

Then, formally, this step is a reduction from edge-weighted $[0 \dots x]$ -#ANTIFACTOR to edge- and relation-weighted X -#ANTIFACTOR^{Rel}.

4. In Lemma 7.4.4 we remove the weights on the relations by adding additional edge-weights.

Formally, this step is a reduction from edge- and relation-weighted X -#ANTIFACTOR^{Rel} to edge-weighted X -#ANTIFACTOR^{Rel}.

5. The last step in Lemma 7.4.5 removes all edge-weights by an appropriate Turing-reduction combined with known techniques for the interpolation of multivariate polynomials.

Formally, this step is a Turing-reduction from edge-weighted X -#ANTIFACTOR^{Rel} to X -#ANTIFACTOR^{Rel}.

In the following we prove all five steps individually and then combine the results to prove Main Lemma 7.4.6.

Step 1. We first consider the reduction starting at Λ -#FACTOR where we set $\Lambda := \{\max X + 1\}$. Note that picking the set Λ based on the set X is sufficient as we know a lower bound for the counting version whenever $\Lambda \neq \{0\}$.

Ideally we would like to state a direct reduction to X -#ANTIFACTOR. However, although we know that $\Lambda \subseteq \mathbb{N} \setminus X$, we do not obtain a trivial reduction. To see this, denote by H an instance of Λ -#FACTOR. When we treat H as a X -#ANTIFACTOR instance G , every solution of H is also a solution for G because of our choice of Λ . However, the converse is not true. Indeed, since the set X is finite and thus, the set of allowed degrees is cofinite, the degree of the solutions for G can be larger than $\max X + 1$. This is now precisely the place where we consider solutions of *minimum* size only. By some preprocessing, we can ensure that no vertex has degree larger than $\max X + 1$ in the solution of minimum size and hence, the solution for G is also a solution for the Λ -#FACTOR^{Rel} instance H .

Lemma 7.4.1. *For every fixed integer $x \geq 0$, there is a pathwidth-preserving many-one reduction from $\{x + 1\}$ -#FACTOR to $[0 \dots x]$ -#MIN-ANTIFACTOR that operates in time polynomial in the size of the input.*

Proof. Denote by H an instance of Λ -#FACTOR with $\Lambda := \{x + 1\}$. Since Λ contains only one element, the set has no gap. Therefore, the algorithm by Cornuéjols [30] (see also Theorem 5.1.4) can be used to check in polynomial time if H has a solution, that is, H contains an $(x + 1)$ -factor. If H has no such solution, then we return a trivial no-instance, for example, a graph with only a single vertex.

Otherwise, we know that H has at least one solution. We replace all vertices in H (these are simple vertices and are assigned the set Λ of allowed degrees) by vertices with the set X (of forbidden degrees). This provides an instance G of $[0 \dots x]$ -#MIN-ANTIFACTOR. We output G as the resulting instance of the reduction.

We first argue about the correctness of the reduction.

Here we exploit that the instance H has no relations.

Claim 7.4.1.1. *The reduction is correct, that is, we have $\#Sol(H) = \#Sol(G)$.*

Proof of Claim. By our choice of Λ , we know that $\{x+1\} = \Lambda \subseteq \mathbb{N} \setminus [0..x]$. Thus, every solution for H is also a solution for G . Hence, we directly get that

$$\#Sol(H) \leq \#Sol(G).$$

To show the converse of this inequality, we can assume that H (and thus, also G) has a solution $S_0 \subseteq E(H)$ as the algorithm checks this initially. Moreover, since S_0 is an $(x+1)$ -factor, that is, all vertices in H are incident to exactly x selected edges in S_0 , we have

$$2 \cdot |S_0| = \sum_{v \in V(H)} \deg_{S_0}(v) = |V(H)| \cdot (x+1).$$

Now consider a *minimum* solution S of G . We claim that S is also a solution for H . By the problem definition, we know that every vertex $v \in V(G)$ is incident to at least $x+1$ selected edges in S , that is, the solution S satisfies $\deg_S(v) \geq x+1$. Hence, we have that

$$2 \cdot |S| = \sum_{v \in V(G)} \deg_S(v) \geq |V(G)| \cdot (x+1) = 2 \cdot |S_0|.$$

By the minimality of S , and the observation that S_0 is also a solution for G , the inequality must be an equality. Hence, every vertex has degree exactly $x+1$ in S which implies that S is an $(x+1)$ -factor. Therefore, the set S is a solution for H , and we get

$$\#Sol(G) \leq \#Sol(H)$$

which concludes the proof. ◀

It remains to analyze the running time of the reduction. With the algorithm by Cornuéjols (see Theorem 5.1.4 and [30]), we can check in time polynomial in the size of H whether H has a solution. Given H , the construction of the instance G takes time linear in the size of H , as the construction is just a change of the assigned set. Hence, the reduction operates in time polynomial in the size of H .

Technically no change is needed, as the problem definition implies this modification.

Since the reduction does not change the pathwidth (for the no-instance the pathwidth might even decrease), this concludes the proof. □

Step 2. As a next step we remove the restriction of counting only minimum solutions. For this we introduce edge-weights and compute the weighted number of solutions. By repeating this process for multiple weights, we can separate the solutions with few edges from the solutions with many edges. Then, we can recover the number of solutions of the original instance by using interpolation techniques.

Fact 2.1.7 ([33, Lemma 2.5]). *Let $P \in \mathbb{Q}[x_1, \dots, x_q]$ be a multivariate polynomial such that, for all $i \in [1..q]$, the degree of x_i in P is bounded by $d_i \in \mathbb{N}$. Furthermore, assume we are given sets $\Xi_i \subseteq \mathbb{Q}$ for $i \in [1..q]$ such that $|\Xi_i| = d_i + 1$ for all $i \in [1..q]$. Consider the Cartesian product of these sets, that is,*

$$\Xi := \Xi_1 \times \dots \times \Xi_q.$$

Then, we can compute the coefficients of P with $\mathcal{O}(|\Xi|^3)$ arithmetic operations when given as input the set $\{(\xi, P(\xi)) \mid \xi \in \Xi\}$.

With this result, we can now formally state and prove the next step in the chain of reductions.

Lemma 7.4.2. *Consider a fixed integer $x \geq 0$. There is a pathwidth-preserving reduction from $[0..x]$ -#MIN-ANTIFACTOR to edge-weighted $[0..x]$ -#ANTIFACTOR with a single edge-weight without changing the size of the graph for each of the oracle calls.*

Proof. Before we state the actual reduction, we first start with some observations. For ease of notation we set $X := [0..x]$ in the following.

Consider an instance G of X -#MIN-ANTIFACTOR. For all $i \in [0..m]$, where m denotes the number of edges of G , we denote by a_i the number of solutions of G with respect to X -#ANTIFACTOR that have size exactly i , that is, the solutions that select exactly i edges. Note that the a_i s also cover solutions that are *not* minimum solutions.

Our goal is to find the smallest i^* such that $a_{i^*} > 0$ and then return a_{i^*} as the number of minimum solutions.

It remains to find these values a_i . For a positive integer w , we define the instance G_w of edge-weighted X -#ANTIFACTOR as the modification of G , where we assign weight w to every edge of G . Although, assigning the same weight to all edges might feel like a scalar, the effect is not that straightforward as we count all solutions.

Then, we can write the (weighted) number of solutions of G_w as

$$\#Sol(G_w) = \sum_{i=0}^m a_i \cdot w^i.$$

Now, if we treat w as a variable, we can see that $\#Sol(G_w)$ behaves as a polynomial in the variable w with degree at most m . Hence, once we evaluate this polynomial for sufficiently many values, we can recover the coefficients a_i by Fact 2.1.7.

With these observations we have everything ready to formally state the reduction.

For all integers $w \in [1..m+1]$, we construct the graph G_w and compute $\#Sol(G_w)$ by the oracle-access to X -#ANTIFACTOR.

Then, we interpret these values as the evaluation of the univariate polynomial $p(w) := \sum_{i=0}^m a_i \cdot w^i$ for all choices of $w \in [1..m+1]$. Next, we use Fact 2.1.7 to recover all coefficients a_i of the polynomial p . With $i^* := \min\{i \in [0..m] \mid a_i \neq 0\}$ we get that G has a_{i^*} solutions (of minimum size) with respect to X -#MIN-ANTIFACTOR.

The correctness of the algorithm follows directly from our observations and the correctness of Fact 2.1.7.

As the new instances of X -#ANTIFACTOR agree with the original instance except that all edges are assigned a specific weight, it remains to bound the running time.

Since the construction of G_w can be done in polynomial time and recovering the a_i s also takes polynomial time by Fact 2.1.7, this concludes the proof. $\square$

Step 3. The next reduction drops the requirement that X must be a continuous interval, that is, we drop the requirement that $X = [0..x]$ for some x . Instead, we now allow arbitrary



(a) The simple vertex before the modification. (b) The complex vertex $\bar{v}$ that is added to vertex v . Here shown for $\max X + 1 = 3$. We denote by α and β the number of selected outer and inner edges, respectively.

Figure 7.5: The modification of the vertices in the reduction from edge-weighted $[0 .. \max X]$ -#ANTIFACTOR to edge- and relation-weighted X -#ANTIFACTOR^{Rel} formalized in Lemma 7.4.3.

finite sets of forbidden degrees, which results in arbitrary cofinite sets of allowed degrees. To adjust for this change, we introduce weighted relations in this reduction.

For the reduction we construct a relation R and attach a complex vertex with a copy of this relation to every single vertex v of the graph by sufficiently many edges. The idea of this gadget is as follows. Whenever vertex v is incident to more than $\max X$ selected edges in the original graph, then any number of edges to this new complex vertex can be selected (assuming the assigned relation R permits). This makes the states with at least $\max X + 1$ selected edges somewhat equivalent (with respect to X).

On the other side, when the vertex v is incident to less than $\max X$ selected edges of the original graph, certain extensions are not possible as this would violate the degree constraints imposed by the set X . The list of possible extensions can be seen as a fingerprint that uniquely characterizes the number of selected edges in the original graph. We use this property to define the weighted relation R such that all those solutions cancel out where a vertex is not incident to at least $\max X + 1$ selected edges from the original graph. Hence, the number of solutions agrees with the number of solutions for the original instance of $[0 .. \max X]$ -#ANTIFACTOR.

We refer to Figure 7.5 for an illustration of the construction from the following reduction.

Lemma 7.4.3. *Let $X \subseteq \mathbb{N}$ be a fixed, non-empty and finite set. There is a pathwidth-preserving reduction from edge-weighted $[0 .. \max X]$ -#ANTIFACTOR with r weights to edge- and relation-weighted X -#ANTIFACTOR^{Rel} with at most $r + \max X + 2$ weights such that the size increases by a constant factor, and the total arity Δ^* of the resulting instances is $\max X + 1$.*

Proof. Consider an arbitrary instance H of edge-weighted $[0 .. \max X]$ -#ANTIFACTOR.

For each (simple) vertex v of H , we apply the following transformation and refer to the resulting instance of X -#ANTIFACTOR^{Rel} by G .

We assign the set X to the vertex v (note that vertex v was assigned the set $[0 .. \max X]$ before) and create a new complex vertex $\bar{v}$. Then, we connect the two vertices v and $\bar{v}$ by $\max X + 1$ parallel edges which we refer to as *inner* edges from now on. Similarly, we refer to the edges from vertex v to its neighbors in H as the *outer* edges.

It remains to define the relation R which we assign to the complex vertex $\bar{v}$. We note that this relation R is identical for all vertices.

For all $\beta \in [0 .. \max X + 1]$, we select one (arbitrary) set of exactly β inner edges and define the relation R such that when $\bar{v}$ is incident to exactly β selected inner edges, then precisely

Though these parallel edges disappear later, one could place a complex vertex with relation EQ₂ on them to obtain a simple graph.

the edges from the predefined set must be selected. Moreover, relation R accepts this set of β selected inner edges with weight w_β . All other sets of selected edges are accepted with weight 0, that is, they are rejected.

Claim 7.4.3.1. *In time polynomial in $\max X$, we can find, for all $\beta \in [0 .. \max X + 1]$, values for the weights w_β such that $\#Sol(H) = \#Sol(G)$.*

Proof of Claim. Our goal is to rule out “bad” solutions in G where we say that a solution is *bad* if a vertex v is incident to k selected outer edges for some $k \leq \max X$. As the set X can be quite general, this is not possible in general. However, every solution in H can be extended in several ways to a solution in G depending on the selection of the inner edges.

We define the weights for the relation R such that the extensions of a bad solution cancel out, that is, they contribute weight 0. Moreover, the weights are chosen such that the extensions of a “good” solution contribute with total weight 1 to the final number of solutions. Phrased differently, we can group the solutions in G according to the edges in H . Then, only those solutions in G are relevant where the corresponding solution in H is good and thus, every vertex v is always incident to k selected outer edges, for some $k \geq \max X + 1$. Note that in order to have solutions that cancel out, we crucially need that the weights w_β might also be negative.

In the following we denote by α the number of selected outer edges and by β the number of selected inner edges. Based on the previous observations, for all $\alpha \in \mathbb{N}$, the weights must satisfy

$$\sum_{\substack{\beta=0 \\ \alpha+\beta \notin X}}^{\max X+1} w_\beta = \begin{cases} 1 & \text{if } \alpha \geq \max X + 1, \\ 0 & \text{if } \alpha \leq \max X. \end{cases} \quad (7.4.1)$$

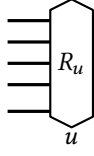
If we consider each w_β as a variable, then the constraints from Equation (7.4.1) form a system of linear equations with $\max X + 2$ variables. Observe further that all constraints with $\alpha \geq \max X + 1$ are identical. Hence, we only have to consider the cases when $\alpha \in [0 .. \max X + 1]$ which leads to $\max X + 2$ constraints.

We claim that this system of linear equations has a solution. For this consider the matrix M formed by the coefficients of Equation (7.4.1) where we index rows by $\alpha \in [0 .. \max X + 1]$ and columns by $\beta \in [0 .. \max X + 1]$. Observe that the anti-diagonal of M and all entries below are 1s as $\max X + 1, \max X + 2, \dots \notin X$. Moreover, every entry one row above the anti-diagonal in M is zero since $\max X \in X$.

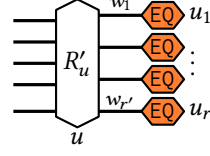
This observation directly implies that the sums for $\alpha + 1$ and α (i.e., the rows $\alpha + 1$ and α) differ by at least one summand, namely by $w_{\max X - \alpha}$. Hence, the matrix M is invertible and there is a solution to Equation (7.4.1).

From the previous observations, we also get a direct method to compute the value of the weights. Indeed, when starting with the sums for $\max X + 1$ and $\max X$, we can find the value for w_0 and eliminate it from the system of linear equations. Then, we repeat this process to iteratively find and eliminate w_1 up to $w_{\max X+1}$.

We conclude that there are values for each w_β such that Equation (7.4.1) holds. Moreover, the above procedure already finds these values in time polynomial in the number of variables which is $\max X + 2$. $\triangleleft$



(a) The complex vertex u with the weighted relation R_u before the modification.



(b) The resulting gadget with an unweighted relation R'_u and edges of weight w_i between u and u_i . The vertices u_i are assigned relation EQ_1 .

Figure 7.6: The modification of the vertices in the reduction from edge- and relation-weighted $X\text{-}\#\text{ANTIFACTOR}^{\text{Rel}}$ to edge-weighted $X\text{-}\#\text{ANTIFACTOR}$ formalized in Lemma 7.4.4.

As a last step, we analyze the new instance G . Recall that H is the initial instance of edge-weighted $X\text{-}\#\text{ANTIFACTOR}$. The above construction doubles the number of vertices, that is, we have $n_G \leq 2n_H$, and increases the pathwidth by 1, that is, we have $\text{pw}(G) \leq \text{pw}(H) + 1$. As the relation R has arity $\max X + 1$ we get $\Delta_G^* = \max X + 1$ which concludes the proof. $\square$

Step 4. The next step of our reduction removes the weights from the relations by using additional edge-weights. For this we add additional edges to the relations or rather complex vertices; for each weight used by the relation we add one edge with the same weight. Then, the new relation agrees with the old one except that now additionally the edge with the corresponding weight must be selected. We refer to Figure 7.6 for an illustration of this modification.

Lemma 7.4.4. *Let $X \subseteq \mathbb{N}$ be an arbitrary set (which may not be fixed but given as input).*

There is a pathwidth-preserving reduction from edge- and relation-weighted $X\text{-}\#\text{ANTIFACTOR}^{\text{Rel}}$ with r different weights to edge-weighted $X\text{-}\#\text{ANTIFACTOR}^{\text{Rel}}$ with at most r weights such that

- *the size increases by a multiplicative factor of $\mathcal{O}(r)$, and*
- *the total arity Δ^* increases to $\Delta^* + r$.*

Proof. We apply the following procedure to each complex vertex u whose corresponding relation R_u uses weights, that is, where the weights of the relation R_u are different from 0 and 1. For some $r' \leq r$, we denote by $w_1, \dots, w_{r'}$ the r' different weights used by R_u .

For all $i \in [1 .. r']$, we create a new complex vertex u_i with relation EQ_1 and make vertex u_i adjacent to vertex u by an edge of weight w_i .

Based on the relation R_u , we design a new unweighted relation R'_u as follows. Whenever relation R_u accepts the input y with weight w_i , for some $i \in [1 .. r']$, then relation R'_u accepts y but additionally requires that the edge to vertex u_i is selected while the edges to the other vertices $u_{i'}$ with $i' \neq i$ remain unselected.

It directly follows that this modification does not change the solution. Moreover, the pathwidth increases by at most 1 and the degree of the complex vertices by at most r . $\square$

Step 5. In the next step, which is also the last step of our chain of reductions, we remove the edge-weights from the graph. First observe that we do not have to change edges of weight 1. Furthermore, we can simply remove all edges with weight 0.

The main idea is to treat each edge-weight as a variable. Then, the number of solutions can be seen as a (unknown) multivariate polynomial in these variables. If we can evaluate this polynomial at sufficiently many points, then we can recover the polynomial. Hence, we can compute the solution when we evaluate the polynomial with the actual weights of the edges.

Therefore, it remains to simulate certain edge weights. Since the output of the counting version is the sum of the weighted solutions, it suffices to create a gadget with multiple extensions which then simulate the weight.

We use the following result to recover the original polynomial from its evaluations.

Fact 2.1.7 ([33, Lemma 2.5]). *Let $P \in \mathbb{Q}[x_1, \dots, x_q]$ be a multivariate polynomial such that, for all $i \in [1..q]$, the degree of x_i in P is bounded by $d_i \in \mathbb{N}$. Furthermore, assume we are given sets $\Xi_i \subseteq \mathbb{Q}$ for $i \in [1..q]$ such that $|\Xi_i| = d_i + 1$ for all $i \in [1..q]$. Consider the Cartesian product of these sets, that is,*

$$\Xi := \Xi_1 \times \dots \times \Xi_q.$$

Then, we can compute the coefficients of P with $\mathcal{O}(|\Xi|^3)$ arithmetic operations when given as input the set $\{(\xi, P(\xi)) \mid \xi \in \Xi\}$.

Formally, we prove the reduction in Lemma 7.4.5.

Lemma 7.4.5. *Let $X \subseteq \mathbb{N}$ be a non-empty set (which may not be fixed but given as input). There is a pathwidth-preserving reduction from edge-weighted X -#ANTIFACTOR^{Rel} on instances of size n and with r different weights to unweighted X -#ANTIFACTOR^{Rel} operating in time $n^{\mathcal{O}(r)}$. The reduction is such that for each oracle call*

- *the size increases by a multiplicative factor of $\mathcal{O}(\log^2(n))$,*
- *the degree of the simple vertices stays the same, and*
- *the total arity increases to at most $\Delta^* + \mathcal{O}(1)$.*

Proof. Denote by H an instance of edge-weighted X -#ANTIFACTOR^{Rel} with n vertices and m edges.

To remove the edge weights we use an interpolation technique based on the one used by Curticapean and Marx [35] to prove a lower bound for COUNTING PERFECT MATCHINGS.

We replace the r edge weights $w_1, \dots, w_r$ by variables $x_1, \dots, x_r$ and treat the value of the solution as an r -variate polynomial $P_H(x_1, \dots, x_r)$. Observe that, for each $i \in [1..r]$, the polynomial P_H has degree at most $m < n^2$ in variable x_i . Moreover, the total degree, that is, the sum of the degrees in a monomial, is at most n^2 , too.

For each variable x_i we pick n^2 values and evaluate the polynomial P_H on all $(n^2)^r$ combinations of these values. Then, we use Fact 2.1.7 to recover the coefficients of the polynomial P and thus, recover the polynomial P_H itself in time $n^{\mathcal{O}(r)}$. Finally, we output $P_H(w_1, \dots, w_r)$.

To evaluate the polynomial P_H on values $x_1, \dots, x_r$, we modify the given instance H and replace each edge with weight w_i by some gadget which simulates a weight equal to the value

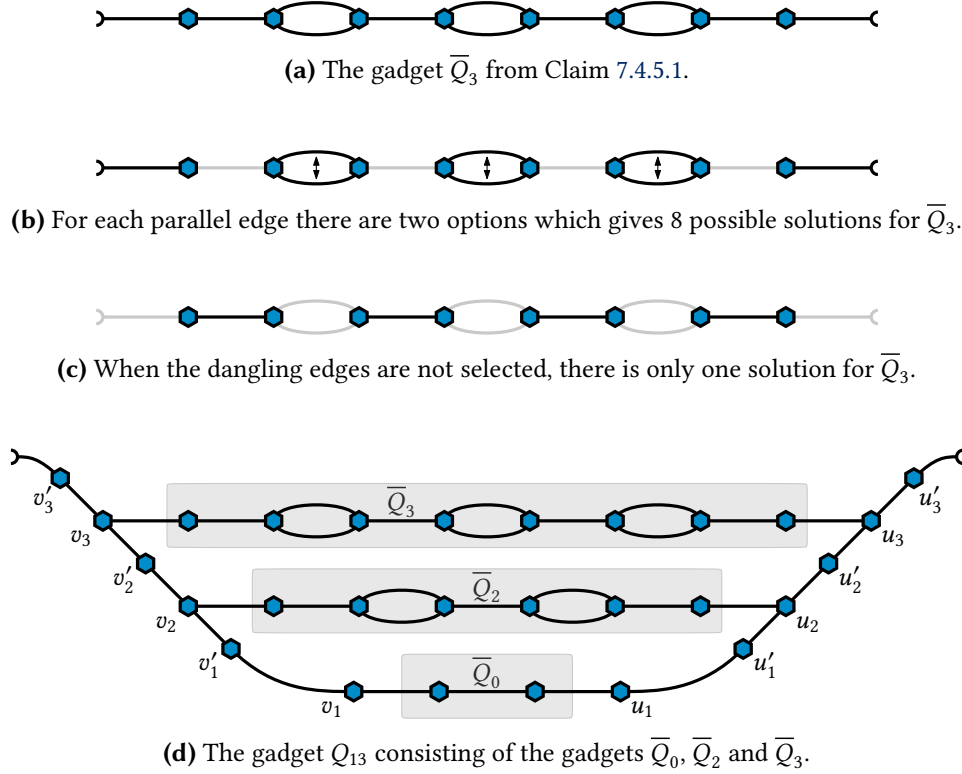


Figure 7.7: An illustration of the gadgets from Lemma 7.4.5 to realize weighted edges. The hexagons represent vertices with relation $\text{HW}_{=1}$.

of x_i . Hence, we first have to show how to achieve n^2 different values for each x_i . See Figure 7.7 for an illustration of the following constructions.

As a first step, we show how to realize edge weights of the form 2^i .

Claim 7.4.5.1. *For all $i \in \mathbb{N}$, there is a gadget $\overline{Q}_i$ with two dangling edges*

- *with 2^i solutions if both dangling edges are selected,*
- *with a unique solution if none of the dangling edges is selected,*
- *and with no solution if exactly one dangling edge is selected.*

The gadget has size $\mathcal{O}(i)$ and pathwidth $\mathcal{O}(1)$.

Proof of Claim. We replace the edge of weight 2^i by a chain of $2i + 3$ edges where, for all $j \in [1 \dots i]$, the $2j + 1$ th edge is a parallel edge. We assign the relation $\text{HW}_{=1}^{(3)}$ to all vertices that are introduced by this gadget. The first and last edge are the dangling edges of the gadget.

When the original edge was selected, we have, for each $j \in [1 \dots i]$, two choices for selecting one of the $2j + 1$ th parallel edges. Moreover, these choices are independent of each other. This contributes a weight of 2^i to the solution as the first and last edge must be selected as well.

When the original edge is not selected, then we only have a unique solution; we select the 2nd, 4th, $\dots$, edge, that is, the edges that are not parallel edges.

When exactly one of the dangling edges is selected, we need to propagate this selection through the chain of edges by following the two previous approaches accordingly. However, this results in one complex vertex that is adjacent to two selected edges which violates the relation assigned to it. Therefore, there is no solution. $\triangleleft$

We use the gadget from Claim 7.4.5.1 to realize arbitrary positive weights $w \in \mathbb{N}$ based on the binary representation of w .

Claim 7.4.5.2. *For all $w \in \mathbb{N}$, there is a gadget Q_w with two dangling edges*

- *with w solutions if both dangling edges are selected,*
- *with a unique solution if none of the dangling edges is selected,*
- *and with no solution if exactly one dangling edge is selected.*

The gadget has size $\mathcal{O}(\log^2 w)$ and pathwidth $\mathcal{O}(1)$.

Proof of Claim. The binary representation of w yields a finite set of integers $S = \{s_1, \dots, s_\ell\} \subseteq \mathbb{N}$ such that $w = \sum_{i=1}^{\ell} 2^{s_i}$.

We create two paths with $2\ell + 1$ vertices each. The first path consists of the complex vertices $v_1, v'_1, \dots, v_\ell, v'_\ell$ and the second path of the complex vertices $u_1, u'_1, \dots, u_\ell, u'_\ell$, and both paths are in this order. We assign the relation $\text{HW}_{=1}$ to each of these vertices.

For all $i \in [1 \dots \ell]$, we add an edge of weight 2^{s_i} between v_i and u_i . Then, we use Claim 7.4.5.1 to replace this edge by the gadget $\overline{Q}_{s_i}$.

This concludes the construction, when we add one dangling edge to the vertices v'_ℓ and u'_ℓ each.

To avoid parallel edges, we can place a complex vertex with relation EQ_2 on each parallel edge.

Consider the case that none of the two dangling edges is selected. As the vertex v'_ℓ must be incident to a selected edge, the edge from v'_ℓ to v_ℓ must be selected. Hence, neither the edge from v_ℓ to u_ℓ nor the edge to $v'_{\ell-1}$ can be selected. Repeating this argument we get that none of the edges from v_i to u_i can be selected and there is only a unique solution.

Using a similar argument we can conclude that there is no solution when exactly one dangling edge is selected.

As a last case we assume that both dangling edges are selected. We claim that in this case exactly one edge from v_i to u_i can be selected for an arbitrarily chosen $i \in [1.. \ell]$. As we can select this edge arbitrarily, the total weight of this solution (or rather the number of solutions) sums up to w because of the binary representation of w .

Assume the edge from v_i to u_i is selected. Then, the edge from u_i to u'_i is not selected. By the relation of u'_i , the edge from u'_i to u_{i+1} must be selected. Therefore, the weighted edge from u_{i+1} to v_{i+1} cannot be selected. As the edge u_{i+1} to u'_{i+1} also cannot be selected and the argument applies recursively to the vertices $u_{i'}$ and $u'_{i'}$ with $i' < i$, no other weighted edge can be selected.

It remains to bound the size and the pathwidth of this construction. Recall that the gadget $\bar{Q}_i$ which simulates an edge of weight 2^i has size $\mathcal{O}(i)$ and pathwidth $\mathcal{O}(1)$. Since the size of the set S is $\mathcal{O}(\log w)$ and the largest value in S is $\mathcal{O}(\log w)$ (which generates weight $2^{\mathcal{O}(\log w)}$), the size of the entire gadget is bounded by $\mathcal{O}(\log^2 w)$.

For the bound on the pathwidth we observe that the earlier gadget has a constant pathwidth which especially implies that the pathwidth is independent of the weight simulated. To see that the entire gadget also has constant pathwidth, we consider a node search strategy. We search the two paths of length $2\ell + 1$ simultaneously starting from vertices v'_ℓ and u'_ℓ . Whenever, we have placed searchers on the vertices v_i and u_i , we clean the path simulating the edge of weight 2^i . As all of these steps only need a constant number of searchers, that is, the number is independent of the weight w , the pathwidth is constant. $\triangleleft$

Now we have everything ready to state the final reduction.

For all $a_1, \dots, a_r \in [1.. n^2]$, we construct an instance $G_{a_1, \dots, a_r}$ of $X\text{-}\#\text{ANTIFACTOR}^{\text{Rel}}$ where we replace each edge of weight w_i in H by the gadget Q_{a_i} from Claim 7.4.5.2 which simulates weight a_i . Then, we query the oracle for $X\text{-}\#\text{ANTIFACTOR}^{\text{Rel}}$ to compute $\#\text{Sol}(G_{a_1, \dots, a_r})$ and define $P_H(a_1, \dots, a_r) = \#\text{Sol}(G_{a_1, \dots, a_r})$ as the evaluation of the unknown polynomial P_H .

Finally, we use Fact 2.1.7 to recover the polynomial P_H from the previous evaluations and output $P_H(w_1, \dots, w_n)$.

Claim 7.4.5.3. *The reduction operates in time $n^{\mathcal{O}(r)}$.*

Proof of Claim. In total, the reduction constructs n^{2r} instances, each takes time polynomial in the input size, and queries the oracle on them. Then, we use Fact 2.1.7 to recover the polynomial P in time $(n^{2r})^3$. A final evaluation of the polynomial completes the proof. $\triangleleft$

As a last step we bound the pathwidth and the size of the new instances.

Claim 7.4.5.4. *The size of the new instances is bounded by $\mathcal{O}(|H| \cdot \log^2 |H|)$.*

The total arity of the new instances is bounded by $\Delta^*(H) + \mathcal{O}(1)$.

The pathwidth of the new instances is bounded by $\text{pw}(H) + \mathcal{O}(1)$.

Proof of Claim. Each of the gadgets Q_w we introduce, simulates weight at most n^2 and thus, has size at most $\mathcal{O}(\log^2(n))$. Hence, the size increases by a factor of $\mathcal{O}(\log^2(n))$ as we might have to replace each edge by such a gadget.

As the degree of each complex vertex that we introduce is at most three, we directly obtain the bound on the total arity Δ^* .

To bound the pathwidth, we assume that we are given a node search strategy for the instance H . For each (weighted) edge uv , searchers are placed on u and v simultaneously at some time step.

When this is the case, we use the node search strategy for the gadget Q_w to clean the vertices of the gadget. As the pathwidth of the gadget G_w is constant, the pathwidth increases only by a constant. $\blacktriangleleft$

This concludes the reduction by Claims 7.4.5.3 and 7.4.5.4 $\square$

We remark that the previous construction uses weights between 1 and n^2 . By this choice, the size increases only by a logarithmic factor (in the number of vertices n). In principle, we could also use weights from 2^1 to 2^{n^2} which would increase size by a polynomial factor. Though this is not an issue for the lower bound for X -#ANTIFACTOR, this reduction could not be applied for the proof of the #W[1]-hardness which we give later.

Putting the Pieces Together. To prove Main Lemma 7.4.6, that is, the reduction from Λ -#FACTOR to X -#ANTIFACTOR^{Rel}, we combine Lemmas 7.4.1 to 7.4.5 in the straightforward way.

Main Lemma 7.4.6. *Fix a finite, non-empty set $X \subseteq \mathbb{N}$.*

There is a pathwidth-preserving reduction starting with an instance H of $\{\max X + 1\}$ -#FACTOR and queries $|H|^{\mathcal{O}(\max X)}$ instances of X -#ANTIFACTOR^{Rel} where the size is bounded by $\mathcal{O}(|H| \cdot \log^2 |H|)$, and the total arity Δ^* is bounded by $\mathcal{O}(\max X)$.

Proof. We define $\Lambda := \{\max X + 1\}$ and consider an instance H of Λ -#FACTOR of size n_H .

We proceed as follows.

1. We apply Lemma 7.4.1 to H to obtain one instance of $[0 .. \max X]$ -#MIN-ANTIFACTOR.
2. We apply Lemma 7.4.2 to the instance from the previous step and obtain a polynomial number (in n_H) of edge-weighted $[0 .. \max X]$ -#ANTIFACTOR instances with a single edge-weight each.
3. We apply Lemma 7.4.3 to each of these instances and obtain a polynomial number (in n_H) of edge- and relation-weighted X -#ANTIFACTOR^{Rel} instances with at most $\max X + 3$ weights.
4. We apply Lemma 7.4.4 to each of these instances and obtain a polynomial number (in n_H) of edge-weighted X -#ANTIFACTOR^{Rel} instances with at most $\max X + 3$ weights.
5. Lastly, we apply Lemma 7.4.5 to each of these instances and obtain $n_H^{\mathcal{O}(\max X)}$ instances G_i of X -#ANTIFACTOR^{Rel}.

For each such instance G_i , we get from the properties of the previous reductions that the number of vertices n_{G_i} is bounded by $\mathcal{O}(n_H \cdot \log^2(n_H))$, the total arity is bounded by $\Delta^*(G_i) \in \mathcal{O}(\max X)$, and the pathwidth is bounded by $\text{pw}(G_i) \leq \text{pw}(H) + \mathcal{O}(1)$. This concludes the proof of Main Lemma 7.4.6. $\square$

7.5 Reducing to Few Types of Relations

As the next main step toward proving the lower bounds for the various variants of GENFAC, we replace the arbitrary relations by few types of well-structured relations. More formally, we replace all relations by gadgets that only use simple vertices or vertices with $\text{HW}_{=1}$ or EQ as relation.

Definition 2.1.6. For a k -dimensional vector $x \in \{0, 1\}^k$, we define $\text{hw}(x)$ as the number of 1s in x , that is, we define $\text{hw}(x)$ as the Hamming weight of x with $\text{hw}(x) := \sum_{i=1}^k x[i]$.

For two positive integers $i, k \in \mathbb{N}$ and a set $S \subseteq \mathbb{N}$, we define the relations

$$\text{HW}_{\in S}^{(k)} := \{x \in \{0, 1\}^k \mid \text{hw}(x) \in S\}, \quad \text{HW}_{=i}^{(k)} := \text{HW}_{\{i\}}^{(k)} \quad \text{and} \quad \text{EQ}_k := \text{HW}_{\{0, k\}}^{(k)}.$$

We say that EQ_k is the equality relation on k inputs.

We use $\text{HW}_{\in S}$ to denote the relation $\text{HW}_{\in S}^{(k)}$ when the arity k of the relation is clear from the context. Additionally, we also write $\text{HW}_{\in S}$ to denote the set of the relations $\text{HW}_{\in S}^{(k)}$ for all $k \in \mathbb{N}$.

We transfer this abuse of notation to $\text{HW}_{=i}$ and EQ .

To formally describe the gadgets, we make use of dangling edges. Recall that a dangling edge is an edge with only one endpoint, that is, their second endpoint is not known yet. We say that a graph with dangling edges realizes a relation if only those selections of the dangling edges are allowed that correspond to an element of the relation. We formalize this intuition in Definition 7.5.1.

Definition 7.5.1 (Realization). Let $R \subseteq 2^{[1..k]}$ be a binary relation. Let G be a graph that might use relations. Denote by $D = \{d_1, \dots, d_k\} \subseteq E(G)$ the dangling edges of G .

We say that G realizes R if, for all $K \subseteq [1..k]$, it holds that $K \in R$ if and only if there is a solution $S \subseteq E(G)$ with $S \cap D = \{d_i \mid i \in K\}$.

The endpoints of the dangling edges are called portals.

When we replace a complex vertex v together with its relation R in a graph G by a graph H_R that realizes the relation, we proceed as follows. We create a copy of the graph H_R , connect all dangling edges of H_R to the neighbors of v in G , and finally remove v together with its incident edges.

This makes the dangling edges become normal edges.

For ease of notation, we also allow that we connect *realizations* to vertices. In this case we implicitly assume that we create a complex vertex with the appropriate relation and then replace the vertex and its relation by the realization.

With the definition of the *insertion* from Definition 2.1.5, which formalizes the intuitive idea, we observe that replacing a vertex with its realization preserves all solutions.

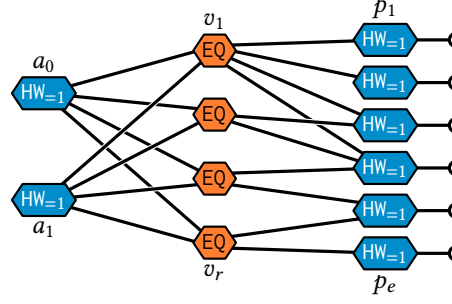


Figure 7.8: The realization of a relation R using only $\text{HW}_{=1}$ and EQ as relations from Main Lemma 7.5.3.

Here shown for a relation R with $R(000011) = R(110011) = R(111001) = R(111100) = 1$ and zero otherwise.

Lemma 7.5.2. *Let G be a $\Lambda\text{-FACTOR}^{\text{Rel}}$ instance, and R the relation of vertex $v \in V(G)$, and let H be a realization of R .*

Assume inserting H in G at v gives us a new graph G' . Then, there is a solution for G if and only if there is a solution for G' .

Proof. The result follows directly from the definition of the insertion from Definition 2.1.5 and the definition of the realization. $\square$

Now we have everything ready to realize arbitrary relations based on relations of type $\text{HW}_{=1}$ and EQ only. Formally, we prove Main Lemma 7.5.3. We refer to Figure 7.8 for an illustration of the corresponding construction.

Main Lemma 7.5.3. *Let $Y \subseteq \mathbb{N}$ be an arbitrary set that might even be a part of the input. There is a parsimonious polynomial-time many-one reduction that transforms an instance G of $Y\text{-FACTOR}^{\text{Rel}}$ into an instance H of $Y\text{-FACTOR}^{\text{HW}_{=1}, \text{EQ}}$ such that for some function f*

- *if given a path decomposition of G of width w_G , then we can simultaneously construct a path decomposition of H of width $w_H \leq w_G + f(\Delta^*(G))$,*
- *the size is bounded by $|H| \leq |G| \cdot f(\Delta^*(G))$,*
- *the total arity is bounded by $\Delta^*(H) \leq f(\Delta^*(G))$*
- *the number of $\text{HW}_{=1}$ -relations in H is even, and*
- *if the relations of G are even, then the arity of each EQ -relation in H is even.*

Additionally, the reduction is oblivious to the set Y and does not introduce any simple vertices.

Proof. We follow the outline of the construction from Lemma 3.3 in [35].

Consider an instance G of $Y\text{-FACTOR}^{\text{Rel}}$. Our goal is to replace each complex vertex v in G together with its relation R_v by the following gadget such that the resulting instance is

equivalent to the original one. Clearly it suffices to focus on relations that are different from $\text{HW}_{=1}$ or EQ .

For now, fix a complex vertex v of arity e with some relation R . We denote by $q_1, \dots, q_e$ the adjacent vertices of v for some arbitrary but fixed ordering. Then, we denote by $R = \{x_1, \dots, x_r\} \subseteq \{0, 1\}^e$ the accepted inputs of the relation R with $r = |R|$.

For ease of notation, we set $A := \{e \bmod 2, 1\}$ in the following, that is, we have $A := \{1\}$ if the arity e is odd and $A := \{0, 1\}$ if the arity e is even. We create the following vertices and edges for this particular relation R .

1. For all $k \in [1..e]$, create a complex vertex p_k with relation $\text{HW}_{=1}$ and connect it to vertex q_k by one edge.
2. For all $i \in [1..r]$, create a complex vertex v_i with relation EQ and connect v_i to all vertices p_k for $k \in [1..e]$ where $x_i[k] = 0$.
3. For all $j \in A$, create a complex vertex a_j with relation $\text{HW}_{=1}$ and connect it to vertex v_i for all $i \in [1..r]$.

This concludes the construction. We denote the resulting instance by H .

From the first and third step of the construction we conclude that there are $e + |A|$ complex vertices with $\text{HW}_{=1}$ as relation. As we know that $|A| = 2 - (e \bmod 2)$, the number of complex vertices with relation $\text{HW}_{=1}$ is even.

Moreover, every vertex v_i is connected to $|A| + (e - \text{hw}(x_i))$ other vertices where $\text{hw}(x_i)$ denotes the number of 1s in the string x_i . If the relation R is even, then $\text{hw}(x)$ is even and thus, the vertex v_i has even degree.

As the next step we show that the construction is correct.

Claim 7.5.3.1. *Every solution for G , can uniquely be extended to a solution for H .*

Proof of Claim. Assume that the selected edges between the vertices $p_1, \dots, p_e$ and $q_1, \dots, q_e$ correspond to an element $x_i \in R$ for some $i \in [1..r]$. We select all edges incident to v_i and no other edges for the solution for the new instance H .

As vertex v_i is adjacent to all vertices a_j , the vertices a_j are incident to one selected edge. Further, vertex v_i is adjacent to precisely those vertices p_k where $x_i[k] = 0$. Hence, each vertex p_k is either incident to the selected edge to q_k or to the selected edge to v_i and thus, incident to exactly one edge in the solution.

This also proves the uniqueness of the solution as every other extension of the solution of G to the gadget would violate the relation of vertex $v_{i'}$ for some $i' \neq i$. $\triangleleft$

Next we prove the reverse direction.

Claim 7.5.3.2. *Every solution for H , is a solution for G when restricting to the edges of G .*

Proof of Claim. Assume we are given a solution for the new instance H . Since the complex vertex a_1 must be incident to exactly one selected edge in the solution, we can find a complex vertex v_i where all incident edges are selected. This especially implies that the edge to a_0 (if it exists) must also be selected.

We denote by P_i the set of (indices of the) complex vertices p_k to which vertex v_i is adjacent. By the construction, vertex v_i corresponds to some $x_i \in R$ and hence, we get $P_i := \{k \in [1..e] \mid x_i[k] = 0\}$.

We denote by $P \subseteq [1..e]$ the set of indices corresponding to the edges selected between p_k and q_k . That is, if the edge $p_k q_k$ is selected, then we have $k \in P$.

Since each vertex p_k is assigned $\text{HW}_{=1}$ as relation, we get $P_i \dot{\cup} P = \{p_1, \dots, p_e\}$. Hence, the dangling edges precisely correspond to x_i which is accepted by relation R . $\triangleleft$

In Claims 7.5.3.1 and 7.5.3.2 we establish a one-to-one correspondence between the solutions of G and H . Hence, we conclude that the reduction is actually parsimonious.

It remains to bound the various measures of H .

Claim 7.5.3.3. *The size of H is bounded by $|G| \cdot f(\Delta^*(G))$, the total arity is bounded by $\Delta^*(G)$, the pathwidth is at most $\text{pw}(G) + f(\Delta^*(G))$.*

Proof of Claim. Assume the arity of a specific relation is at most e . Then, the size of the relation R , that is, the number of accepted inputs, is bounded by $|R| \leq 2^e$. This directly provides an upper bound of $2^e \cdot e + \mathcal{O}(1)$ for the size of the gadget we created.

Recall that we replace each relation of the original instance by an appropriate gadget. For each relation, this replacement increases the size of the instance by an additive factor bounded by a function depending on Δ^* as this bounds the arity of each relation and moreover, bounds the sum of the relations in one bag.

Now consider a path decomposition of the original instance G . In each bag, we can replace the complex vertex by the gadget from above. As the sum over the arity of each relation in the bag is bounded by Δ^* , the size of each bag increases at most by an additive term of $f(\Delta^*)$ for some appropriately chosen function. This directly provides a bound for the pathwidth of the new instance H but also bounds the total arity of the new instance by a function of the original value. $\triangleleft$

This concludes the proof of the reduction as the gadget does not create any simple vertex and thus, does not depend on the set Y . $\square$

We remark that Main Lemma 7.5.3 holds for *all* sets and not just for difficult or non-trivial sets. Hence, the difficulty lies in the realization of the remaining relations $\text{HW}_{=1}$ and EQ . This is precisely what we do next.

7.6 Decision Version: Realizing Relations

As a last step for the decision version, we realize the remaining relations of type $\text{HW}_{=1}$ and EQ , that is, we prove Main Lemma 7.6.7.

Main Lemma 7.6.7. *Fix a non-empty set $\Lambda \subseteq \mathbb{N}$ with $0 \notin \Lambda$ and max-gap $\Lambda > 1$ that is either finite or cofinite.*

Let $Y \subseteq \mathbb{N}$ be an arbitrary set that might even be a part of the input. There is a parsimonious many-one reduction from $Y\text{-FACTOR}^{\text{HW}_{=1}, \text{EQ}}$ with an even number of $\text{HW}_{=1}$ -relations and where all

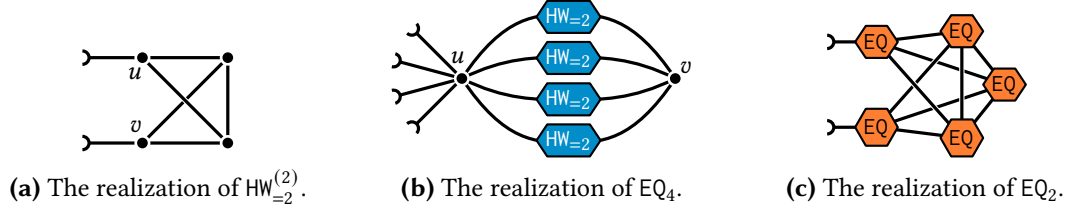


Figure 7.9: An illustration of the realization of the auxiliary relations from Lemma 7.6.1 for the set $\Lambda = \{3, 4, 8\}$.

EQ-relations have even arity to (Λ, Y) -FACTOR such that the width of a given path decomposition increases by at most $f(\Delta^*)$, and size increases by a factor of $f(\Delta^*)$.

Our goal is to realize the $\text{HW}_{=1}$ - and EQ -relations by a graph that uses vertices with set Λ only, that is, the existing vertices with set Y are not changed. We realize the two types of relations in three steps and consider various cases depending on the set Λ .

1. We first realize simple but powerful relations which we frequently use in later construction. We formalize these realizations by Lemma 7.6.1.
2. With these auxiliary relations, we realize EQ -relations of (almost) arbitrary arity. Depending on the set Λ , we consider two cases which are covered by Lemmas 7.6.2 and 7.6.3.
3. As a last step, we realize the $\text{HW}_{=1}$ -relations. Here we first show, that it suffices to consider $\text{HW}_{=1}$ of small arity (here at most 3) to realize these relations of arbitrary arity. See Lemma 7.6.4 for the formal statement. Then, depending on the set Λ , we consider two cases and realize the relations of small arity in Lemmas 7.6.5 and 7.6.6.

Finally, we combine these results to prove Main Lemma 7.6.7.

Auxiliary Relations. We start with three auxiliary relations $\text{HW}_{=2}^{(2)}$, EQ_{d+1} , and EQ_2 where d denotes the largest gap in the set Λ . Note that the relation $\text{HW}_{=2}^{(2)}$ can be seen as two copies of a relation $\text{HW}_{=1}^{(1)}$. We refer to Figure 7.9 for an illustration of these realizations.

Lemma 7.6.1. *For all non-empty sets $\Lambda \subseteq \mathbb{N}$ with $0 \notin \Lambda$ and $\max\text{-gap } \Lambda = d > 1$, we can realize each of the three relations $\text{HW}_{=2}^{(2)}$, EQ_{d+1} , and EQ_2 by simple graphs with $(\text{top } \Lambda)^{\mathcal{O}(1)}$ vertices of degree at most $\text{top } \Lambda$ and with assigned set Λ .*

Proof. We use the following construction for each of the relations.

$\text{HW}_{=2}^{(2)}$: We construct a clique with $\min \Lambda + 1$ simple vertices and split an arbitrary edge uv of this clique into two dangling edges attached to u and v . This is possible as $0 \notin \Lambda$ by assumption.

Every vertex of the clique needs at least $\min \Lambda > 0$ selected incident edges in any solution. Since each vertex of the clique is incident to precisely $\min \Lambda$ edges, all incident edges must be selected. Thus, especially the two dangling edges must be selected in any solution.

EQ_{d+1}: Denote by $a \in \Lambda$ an arbitrary integer such that a gap of size d starts at a . Formally, choose $a \in \Lambda$ such that $[a .. a + d + 1] \cap \Lambda = \{a, a + d + 1\}$.

We create two simple vertices u and v and a complex vertices with relation $\text{HW}_{=2}^{(2)}$. Then, we make each such complex vertex adjacent to both vertices u and v . We add $d + 1$ dangling edges to u and replace the complex vertices by their realization from the previous step.

Observe that in all solutions the vertices u and v are incident to a selected edges each as these edges are enforced by the adjacent vertices with $\text{HW}_{=2}^{(2)}$ as relation. Since vertex v does not have any further neighbors, it receives a valid number of selected incident edges.

Vertex u has $d + 1$ dangling edges and thus, either none or all of them have to be selected since $[a .. a + d + 1] \cap \Lambda = \{a, a + d + 1\}$.

EQ₂: We create a clique with $d + 2$ complex vertices with EQ_{d+1} as relation. We split an arbitrary edge uv of this clique into two dangling edges attached to u and v and replace the relations by their realization from the previous step.

Since all EQ_{d+1}-relations require that either all their incident edges are selected or all incident edges are unselected, the two dangling edges must be either both selected or both unselected in the solution.

As a last step of the construction, we bound the degree of the vertices. For this we observe that the vertex u from the realization EQ_{d+1} has degree $a + d + 1$ which is the maximum among these constructions. As this value can be bounded by $\max \Lambda$ when the set is finite and by $\max(\mathbb{N} \setminus \Lambda) + 1$ then the set is cofinite, the claim follows by the definition of $\text{top } \Lambda$. $\square$

Realizing EQ. With the realizations of the auxiliary relations at hand, we can realize $\text{HW}_{=1}$ and EQ of (almost) arbitrary arity. We start with the realization of EQ and distinguish two cases. In the first case all elements of the set Λ have the same parity and in the second case there are numbers of both parities. This distinction is necessary as in the first case we can only realize EQ_{2k}-relations, that is, equality relations of even arity.

Lemma 7.6.2. *Consider a set $\Lambda \subseteq \mathbb{N}$ with $0 \notin \Lambda$ and $\text{max-gap } \Lambda > 1$ such that all elements in Λ have the same parity.*

Then, for all $k \geq 1$, we can realize EQ_{2k} with at most $k \cdot (\text{top } \Lambda)^{\mathcal{O}(1)}$ vertices of degree at most $\text{top } \Lambda$.

Proof. For $k = 2$ we can use the construction of Lemma 7.6.1. Assume for a moment that we can also realize EQ₄. Then, we can realize EQ_{2k} for all $k \geq 2$ by a chain of $k - 1$ copies of EQ₄.

It remains to realize EQ₄. We create one complex vertex v with EQ_{d+1} as relation and add four dangling edges to it. We also create $(d + 1 - 4)/2$ complex vertices with EQ₂ as relation and make them adjacent to vertex v by both of their edges each. First, observe that this is well-defined, as all elements in Λ have the same parity and thus, the gap size d is odd. Second, we can replace all complex vertices along with their relations by the realization according to Lemma 7.6.1.

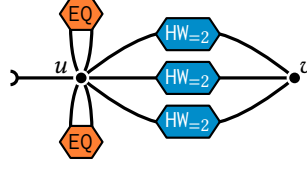


Figure 7.10: An illustration of the realization of EQ_1 from Lemma 7.6.1 for the set $\Lambda = \{3, 4, 8\}$ where we pick $b = 3$ and $c = 8$. (With $c = 4$ we can avoid the vertices with relation EQ .)

This concludes the construction and already provides the bound for the size and the degree. The graph is simple as the endpoints of the dangling edges in the realization of EQ_2 are different. Correctness follows immediately from the fact that either all (dangling and non-dangling) edges are selected or all edges are not selected for each relation involved. $\square$

When the set Λ contains an even and an odd number, then we can realize equality relations of arbitrary arity. This is quite natural as a gap of size 2 already realizes EQ_3 by Lemma 7.6.1.

Observe that the relation EQ_1 “does not care” whether the incident edge is selected or not. This property is quite useful when we have to “eat up” edges introduced by other gadgets.

Lemma 7.6.3. *Consider a set $\Lambda \subseteq \mathbb{N}$ with $0 \notin \Lambda$ and $\max\text{-gap } \Lambda = d > 1$ such that Λ contains even and odd numbers.*

Then, for all $k \geq 1$, we can realize EQ_k with at most $k \cdot (\text{top } \Lambda)^{O(1)}$ vertices of degree at most $\text{top } \Lambda$.

Proof. We first realize EQ_k for $k = 1, 2, 3$ and then provide a general construction for larger arity.

EQ_1 : Pick two numbers $b, c \in \Lambda$ with different parity such that $b < c$. We create two simple vertices u and v and make both vertices adjacent to the same b complex vertices with relation $HW_{=2}^{(2)}$. We additionally create $(c - b - 1)/2$ complex vertices with relation EQ_2 and make them adjacent to u via both edges. To complete the construction, we add one dangling edge to vertex u . See Figure 7.10 for an illustration of this construction.

Since vertex v does not have any dangling edges but is always incident to b selected edges, the vertex is always in a valid state.

If the dangling edge incident to vertex u is selected, then we also select the edges to the EQ_2 -relations. In this case, vertex u is incident to $b + 2 \cdot (c - b - 1)/2 + 1 = c$ selected edges. If the dangling edge is not selected, then we do not select the edges to the EQ_2 -relations. In this case, vertex u is incident to b edges in total.

EQ_2 : The realization of EQ_2 follows directly from Lemma 7.6.1.

EQ_3 : For the realization of EQ_3 we start with one complex vertex u with relation EQ_{d+1} which can be realized by Lemma 7.6.1. Then, we create $d - 2 \geq 0$ complex vertices with relation EQ_1 and make them adjacent to vertex u . Finally, we add three dangling edges to vertex u . When we replace all relations by their realization, this gadget realizes EQ_3 .

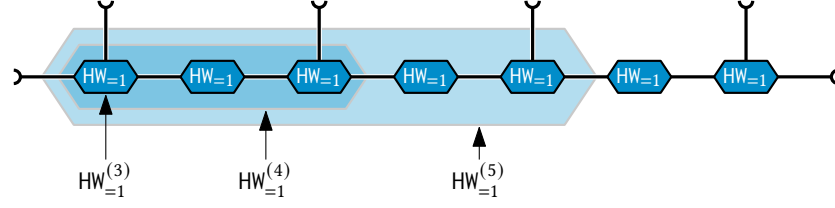


Figure 7.11: Example of the inductive construction from Lemma 7.6.4 for $\text{HW}_{=1}^{(6)}$ using only realizations of $\text{HW}_{=1}^{(3)}$ and $\text{HW}_{=1}^{(2)}$.

To realize EQ_k for $k \geq 4$, we construct a chain of $k - 2$ complex vertices, all with relation EQ_3 , and replace them by their realization from above.

If one dangling edge is selected, then all edges of the gadget must be selected because of the EQ_3 -relation. Hence, either all or none of the dangling edges are selected, which concludes the proof. $\square$

Realizing $\text{HW}_{=1}$. As a next step, we consider the other type of relations that are left in the graph, that is, we realize $\text{HW}_{=1}$. For this we first show that once we can realize $\text{HW}_{=1}$ of small arity, then we can realize relations of higher arity inductively. We refer to Figure 7.11 for an illustration of the following construction.

Lemma 7.6.4. *For all $k \geq 1$, we can realize $\text{HW}_{=1}^{(k)}$ with an odd number of $\text{HW}_{=1}$ -relations of arity at most three such that the following holds: Assume that, for all $k' \in [1 \dots 3]$, we can realize $\text{HW}_{=1}^{(k')}$ by a simple graph with at most N vertices of degree at most D . Then, the mentioned realization of $\text{HW}_{=1}^{(k)}$ has at most $\mathcal{O}(k \cdot N)$ vertices of degree at most D .*

Proof. We realize the relation $\text{HW}_{=1}^{(k)}$ inductively and start with $k \in [1 \dots 3]$ as the trivial base cases.

We assume for the inductive step that, for all arities $k' \leq k$, we can realize $\text{HW}_{=1}^{(k')}$ with an odd number of $\text{HW}_{=1}^{(\ell)}$ -relations where $\ell \in [1 \dots 3]$.

To realize $\text{HW}_{=1}^{(k+1)}$, we create a complex vertex u with relation $\text{HW}_{=1}^{(k)}$. We create two new complex vertices v and w and connect vertex v to the vertices u and w . We assign the relation $\text{HW}_{=1}^{(2)}$ to vertex v and assign $\text{HW}_{=1}^{(3)}$ to vertex w . As last step we add two dangling edges to w such that the entire gadget has $k - 1 + 2 = k + 1$ dangling edges.

Assume one dangling edge of u is selected, then the edge from u to v must not be selected but the edge from v to w must be selected. Hence, no dangling edge of w can be selected. The case when one of the dangling edges of vertex w is selected, follows symmetrically.

It is not possible that more than one or zero dangling edges are selected, as then the relation of one of the three vertices u , v , or w would not be satisfied.

Recall that the relation $\text{HW}_{=1}^{(k)}$ can be replaced by an odd number of $\text{HW}_{=1}^{(\ell)}$ -relations where $\ell \in [1 \dots 3]$. Since our construction introduced two new relations, the parity of the number of relations remains odd. Moreover, when we replace the relations by their realization, the bound on the size of the realization and on the degree of the vertices follow directly. $\square$

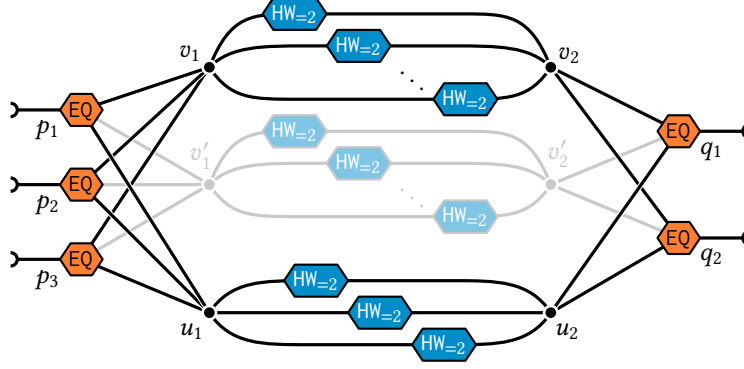


Figure 7.12: The realization of $\text{HW}_{=1}^{(3)}$ together with $\text{HW}_{=1}^{(2)}$ from Lemma 7.6.5 for the set $\Lambda = \{3, 4, 8\}$.

With the meta-result from Lemma 7.6.4, we now realize $\text{HW}_{=1}$. Again, we distinguish two cases depending on the parity of the elements in Λ . For a first general case we show that we can realize pairs of $\text{HW}_{=1}$ -relations instead of a single $\text{HW}_{=1}$ -relation. Afterward we consider those cases where we can realize a single $\text{HW}_{=1}$.

Lemma 7.6.5. *Consider a set $\Lambda \subseteq \mathbb{N}$ with $0 \notin \Lambda$ and $\max\text{-gap } \Lambda = d > 1$.*

Then, for all $k, \ell \geq 1$, we can realize $\text{HW}_{=1}^{(k)}$ together with $\text{HW}_{=1}^{(\ell)}$ by simple graphs with at most $(k + \ell) \cdot (\text{top } \Lambda)^{\mathcal{O}(1)}$ vertices of degree at most $\text{top } \Lambda + 2$.

Proof. We first show how to realize $\text{HW}_{=1}^{(k_1)}$ together with $\text{HW}_{=1}^{(k_2)}$ for all $k_1, k_2 \in [1 \dots 3]$. We refer to Figure 7.12 for an illustration of this construction.

Denote by $a \in \Lambda$ an arbitrary integer such that a gap of size d starts at a , that is, choose $a \in \Lambda$ such that $[a \dots a + d + 1] \cap \Lambda = \{a, a + d + 1\}$,

We create $k_1 + k_2$ complex vertices $p_1, \dots, p_{k_1}$ and $q_1, \dots, q_{k_2}$ with relation EQ_4 . Each of these vertices is incident to one dangling edge of the corresponding $\text{HW}_{=1}$ -relation. We create six simple vertices $u_1, u_2, v_1, v_2, v'_1, v'_2$ and make the three vertices u_1, v_1, v'_1 adjacent to the k_1 vertices $p_1, \dots, p_{k_1}$, and make the three vertices u_2, v_2, v'_2 adjacent to the k_2 vertices $q_1, \dots, q_{k_2}$.

Then, we create $a - 1$ complex vertices with relation $\text{HW}_{=2}^{(2)}$ and make each of them adjacent to u_1 and u_2 by one edge. Moreover, we create $a + d$ complex vertices with relation $\text{HW}_{=2}^{(2)}$ and make each of them adjacent to v_1 and v_2 by one edge. Similarly, create $a + d$ complex vertices with relation $\text{HW}_{=2}^{(2)}$ and make each of them adjacent to v'_1 and v'_2 by one edge.

We claim that this gadget works as expected.

Claim 7.6.5.1. *The previous gadget realizes $\text{HW}_{=1}^{(k_1)}$ together with $\text{HW}_{=1}^{(k_2)}$.*

Proof of Claim. Observe that in any solution, the vertices u_1 and u_2 are incident to $a - 1$ selected edges because of the $\text{HW}_{=2}^{(2)}$ -relations. Hence, the vertices u_1 and u_2 must be incident to *at most one* further selected edge. Indeed, since $k_1, k_2 \leq 3$ and by the gap starting at a , we know that $a + 1, a + 2 \notin \Lambda$. Similarly, the vertices v_1 and v_2 (and also their “twins” v'_1 and v'_2), are already incident to $a + d$ selected edges. Hence, the vertices v_1 and v_2 must receive *at least one* more selected edge because $a + d \notin \Lambda$.

Because of the assigned equality relation, exactly one vertex from $p_1, \dots, p_{k_1}$ must be incident to a selected dangling edge. As we get the same result for the vertices $q_1, \dots, q_{k_2}$, this concludes the proof. $\triangleleft$

To realize relations of larger arity, we use Lemma 7.6.4. Although each such realization uses an odd number of $\text{HW}_{=1}$ -relations of small arity (i.e., arity at most 3), their total number is even as we realize two relations simultaneously. Hence, we can replace pairs of these relations by the construction from above.

This concludes the proof as the size bound and the bound on the degree follow directly from the previous construction and Lemma 7.6.4. $\square$

If all elements of Λ are even, we cannot realize a single $\text{HW}_{=1}$ -relation because of parity issues. Hence, in order to realize a single such relation, it suffices if a single odd number is present in the set Λ .

Although Lemma 7.6.6 is not needed to prove Main Lemma 7.6.7, we present it here for completeness.

Lemma 7.6.6. *Consider a set $\Lambda \subseteq \mathbb{N}$ with $0 \notin \Lambda$ and $\max\text{-gap } \Lambda = d > 1$ such that Λ contains at least one odd number.*

Then, for all $k \geq 1$, we can realize $\text{HW}_{=1}^{(k)}$ by a simple graph with at most $k \cdot (\text{top } \Lambda)^{\mathcal{O}(1)}$ vertices of degree at most $\text{top } \Lambda + 2$.

Proof. We first claim that it suffice to realize $\text{HW}_{=1}^{(1)}$ or $\text{HW}_{\in\{0,1\}}^{(1)}$. We denote this relation for now by R . Then, we use Lemma 7.6.5 to realize $\text{HW}_{=1}^{(k)}$ together with $\text{HW}_{=1}^{(1)}$. When we connect the latter $\text{HW}_{=1}^{(1)}$ -relation to the relation R from above, we directly obtain a realization of $\text{HW}_{=1}^{(k)}$.

Note, this relation does not follow from Lemma 7.6.3 as the set Λ might not contain even numbers.

In the remainder we focus on the realization of $\text{HW}_{=1}^{(1)}$ or $\text{HW}_{\in\{0,1\}}^{(1)}$. Which relation is realized depends on the set Λ .

Denote the smallest odd number in Λ by $o \in \Lambda$. We create a new simple vertex v with one dangling edge and make it adjacent to $(o-1)/2$ complex vertices with relation $\text{HW}_{=2}^{(2)}$. As this forces $o-1$ edges to vertex v and $o \in \Lambda$, the dangling edge can be selected in a solution. If we additionally have that $o-1 \in \Lambda$, we realize $\text{HW}_{\in\{0,1\}}^{(1)}$ and otherwise, we directly realize $\text{HW}_{=1}^{(1)}$, which concludes the proof. $\square$

Combining the Realizations. With Lemmas 7.6.2, 7.6.3 and 7.6.5 we can remove all relations from the initially given instance and complete the reduction from $Y\text{-FACTOR}^{\text{HW}_{=1}, \text{EQ}}$ to $(Y, \Lambda)\text{-FACTOR}$. Formally, we prove Main Lemma 7.6.7.

Main Lemma 7.6.7. *Fix a non-empty set $\Lambda \subseteq \mathbb{N}$ with $0 \notin \Lambda$ and $\max\text{-gap } \Lambda > 1$ that is either finite or cofinite.*

Let $Y \subseteq \mathbb{N}$ be an arbitrary set that might even be a part of the input. There is a parsimonious many-one reduction from $Y\text{-FACTOR}^{\text{HW}_{=1}, \text{EQ}}$ with an even number of $\text{HW}_{=1}$ -relations and where all EQ-relations have even arity to $(\Lambda, Y)\text{-FACTOR}$ such that the width of a given path decomposition increases by at most $f(\Delta^*)$, and size increases by a factor of $f(\Delta^*)$.

Proof. Assume we are given an instance G of $Y\text{-FACTOR}^{\text{HW}_{=1}, \text{EQ}}$ with an even number of $\text{HW}_{=1}$ -relations and where the EQ-relations are even (and thus, have even arity).

Depending on the set Λ , we replace all EQ-relations by their realization according to Lemma 7.6.2 or Lemma 7.6.3. Then, we replace all remaining $\text{HW}_{=1}$ -relations by their realization according to Lemma 7.6.5. We refer to the final instance of (Y, Λ) -FACTOR as H .

Observe that the size of the realizations of a relation with arity d is bounded by $f(d)$ for some function f which possibly depends on Λ . Hence, we bound the size of H by $|H| \leq |G| \cdot f(\Delta^*(G))$.

To bound the pathwidth of the graph H , we modify the path decomposition of G . We replace each complex vertex in a bag β by the vertices of the realization of the corresponding relation. Thus, the size of the bags increases from $\text{pw}(G)$ to at most by $\text{pw}(G) + \Delta^*(G) \cdot f(\Delta^*(G))$ which we bound by $\text{pw}(G) + f'(\Delta^*(G))$ for some appropriately chosen function f' . $\square$

7.7 Counting Version: Realizing Relations

In this section we realize the relations for the counting version. That is, we prove the reduction from Y -#FACTOR $^{\text{HW}_{=1}, \text{EQ}}$ to (Y, Λ) -#FACTOR as formalized in Main Lemma 7.7.14

Main Lemma 7.7.14. *Let $\Lambda \subseteq \mathbb{N}$ be a non-empty set with $\Lambda \notin \{\{0\}, \mathbb{N} \setminus \{0\}, \mathbb{N}\}$ that is finite or cofinite. Let $Y \subseteq \mathbb{N}$ be an arbitrary set that might even be given as input.*

There is a Turing-reduction from Y -#FACTOR $^{\text{HW}_{=1}, \text{EQ}}$ to (Y, Λ) -#FACTOR with running time $f(\text{top } \Lambda + \Delta^*) \cdot n^{\mathcal{O}(1)}$, increasing the size by a factor of $f(\text{top } \Lambda)$, decreasing Δ^* to zero, and increasing the width of the path decomposition by an additive term of $\Delta^* \cdot f(\text{top } \Lambda)$ for each oracle call.

Recall, in Main Lemma 7.6.7, that is, the corresponding result for the decision version of Λ -FACTOR, we assume that $\min \Lambda > 0$ and $\max\text{-gap } \Lambda > 1$ when realizing the relations. For the counting version, we can strengthen the result and drop both requirements but only exclude three explicit cases. When $\Lambda = \emptyset$ there is no solution for Λ -FACTOR, when $\Lambda = \{0\}$ there is only one solution where we select no edges at all, and when $\Lambda = \mathbb{N}$ there are 2^m solutions corresponding to all subsets of the m edges. By the design of our reduction, we also need to exclude the case when $\Lambda = \mathbb{N} \setminus \{0\}$. However, in this case we consider a separate reduction in Section 7.9.

Because of the change in requirements, we need to consider significantly more cases when realizing the relations. Moreover, because of the potential absence of a gap, we must also realize EQ by a different method than before. Surprisingly, for the counting version we can reduce EQ to $\text{HW}_{=1}$ which we must realize anyway. Since we can have $\min \Lambda = 0$, we also need a new way to force edges to be selected which is the stepping-stone for other relations.

To formalize the different steps of the reduction, we first introduce the Holant framework. We note that this framework was already used by Curticapean and Marx to prove the #SETH-based lower bounds for COUNTING PERFECT MATCHINGS [35]. Hence, the following results can be seen as an extension of this work.

The Holant Framework. We use the Holant framework to contextualize several versions of #GENFAC. See [20, 21, 71, 79, 92, 99] for further information and results about this concept.

In the Holant framework, we are given a signature graph $\Omega = (V, E)$, where every edge $e \in E$ has a weight w_e . Every vertex $v \in V$ is labeled with a signature $f_v: 2^{I(v)} \rightarrow \mathbb{Q}$, where $I(v)$

is the incidence vector of edges incident to v . A *solution* is a subset of edges such that the signature for each vertex is non-zero. The *weight of a solution* is the product of the weights of all edges in the solution multiplied by the product of the values of the signatures of all the vertices. Then, the Holant of Ω is defined as the sum of the weights of all solutions. Formally, we have

$$\text{Holant}(\Omega) := \sum_{x \subseteq E(\Omega)} \prod_{e \in x} w_e \prod_{v \in V(\Omega)} f_v(x \cap I(v)).$$

The Holant problem is easily seen to be a weighted generalization of the counting versions of GENFAC. For example, the problem Λ -#FACTOR corresponds to the Holant problem on *unweighted* graphs where every vertex v has the symmetric relation f where, for all $z \subseteq I(v)$, we set

$$f(z) := \begin{cases} 1 & \text{if } |z| \in \Lambda, \\ 0 & \text{if } |z| \notin \Lambda. \end{cases}$$

We say that vertices with such a signature are $\text{HW}_{\in \Lambda}$ -vertices.

In general, we say that a signature f is unweighted if $\text{img}(f) \subseteq \{0, 1\}$, that is, the signature either rejects an input or accepts with weight 1.

Definition 7.7.1. Consider d relations $R_1, \dots, R_d$. We define $\text{HOLANT}(R_1, \dots, R_d)$ as the set of Holant problems where every edge of the input graph is unweighted, and every vertex has an unweighted signature R_j for some $j \in [1 \dots d]$.

By an abuse of notation, we also use R_j in Definition 7.7.1 to denote a *family* of relations. For example, we may use $\text{HOLANT}(\text{HW}_{=1})$ when every vertex has relation $\text{HW}_{=1}^{(k)}$ for some k .

When given a path decomposition of the graph, we define the total arity Δ^* of a Holant problem in the same way as for Λ -FACTOR^{Rel}. That is, we set

$$\Delta^* := \max_{\text{bag } \beta} \sum_{v \in \beta} \deg(v).$$

While we mostly use the unweighted problem, we use the edge-weighted Holant problem for one of the first reductions. The *edge-weighted* version of $\text{HOLANT}(R_1, \dots, R_d)$ corresponds to the restriction of the general Holant problem where all signatures reject an input or accept an input with weight 1, that is, there is no requirement that all edges have weight 1.

To realize the relations, we use so-called “gates”. In essence, the gates in the Holant framework correspond to the realizations for the decision version.

Definition 7.7.2 (Gate). A gate is a signature graph Γ , possibly containing a set $D \subseteq E(\Gamma)$ of dangling edges, and all edges have weight 1 (unless explicitly mentioned).

The signature realized by Γ is the function $\text{SIG}(\Gamma): 2^D \rightarrow \mathbb{Q}$ that maps a selection $x \subseteq D$ of dangling edges to

$$\text{SIG}(\Gamma, x) = \sum_{y \subseteq E(\Gamma) \setminus D} \left(\prod_{e \in y} w(e) \cdot \prod_{v \in V(\Gamma)} f_v((x \cup y) \cap I(v)) \right)$$

Outline. To prove the reduction from $Y\text{-}\#\text{FACTOR}^{\text{HW}_{=1}, \text{EQ}}$ to $(Y, \Lambda)\text{-}\#\text{FACTOR}$, which is identical to a reduction from $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{=1}, \text{EQ})$, to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda})$, we provide a chain of reductions in order to keep the complexity of the individual results manageable.

The main outline consists of the following four steps.

1. We first realize the EQ-relations by gadgets that use edge-weights. Formally, we first prove a reduction from $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{=1}, \text{EQ})$ to edge-weighted $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{=1})$ in Lemma 7.7.4.

Then, we remove these edge-weights by the approach introduced in Section 7.4. We state the reduction to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{=1})$ in Lemma 7.7.5.

2. In the next step, we realize $\text{HW}_{=1}$ by forcing single edges to be selected. That is, we provide a reduction from $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{=1})$ to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda}, \text{HW}_{=1}^{(1)})$. As this reduction is non-trivial we split this step further. We formalize these reductions in Lemmas 7.7.7 to 7.7.10.

3. For the last step we reduce from $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda}, \text{HW}_{=1}^{(1)})$ to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda})$. Again we consider different cases that are formalized by Lemmas 7.7.10 to 7.7.13.

Note that Lemma 7.7.10 is used in two steps.

Finally, we combine these results to prove Main Lemma 7.7.14. We refer to Figure 7.13 for an illustration of the full chain of reductions.

Step 1. For our first step, we realize the equality relations. Recall that for the decision version we exploit the gap. For the counting version, we need a different approach as there might be no gap anymore. Instead, we use a result by Curticapean and Marx [35] for COUNTING PERFECT MATCHINGS to realize the relation EQ in our setting. Since this construction is for perfect matchings, every vertex can be replaced by a vertex with signature $\text{HW}_{=1}$.

Fact 7.7.3 ([35, Lemma 3.2]). *For all even $k \in \mathbb{N}$, there is a gate realizing EQ_k using only $\text{HW}_{=1}$ as signature. This gate features $\mathcal{O}(k)$ vertices and edges, and only $\{-1, 1, 1/2\}$ as edge weights.*

Although the construction from Fact 7.7.3 consists of a non-trivial gadget, it does not provide much insight into the problem and we omit its proof. However, we note that the gate heavily depends on the counting version of the problem. More precisely, the gate exploits that certain edge selections can have multiple extensions that might cancel-out eventually. Thus, the gate cannot be used for the decision version.

The gate was found by a computer algebra system [35]. See Appendix C in [32] for further details.

With this result we can now formally state the next step of the reduction.

Lemma 7.7.4. *Let $R_1, \dots, R_d$ be d arbitrary relations for some $d \geq 0$. There is a polynomial-time many-one reduction from $\text{HOLANT}(R_1, \dots, R_d, \text{EQ})$ to $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{=1})$ with edge weights from $\{-1, 1/2, 1\}$ such that the pathwidth increases at most by $f(\Delta^*)$ for some fixed function f depending only on the total arity Δ^* , and the maximum degree is unaffected, or increases to 6.*

Proof. For a given instance of $\text{HOLANT}(R_1, \dots, R_d, \text{EQ})$, we replace each complex EQ-vertex by the appropriate gate from Fact 7.7.3. This yields an instance of $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{=1})$ with edge weights from $\{-1, 1/2, 1\}$. Further, this only alters the pathwidth by an additive factor depending only on Δ^* . $\square$

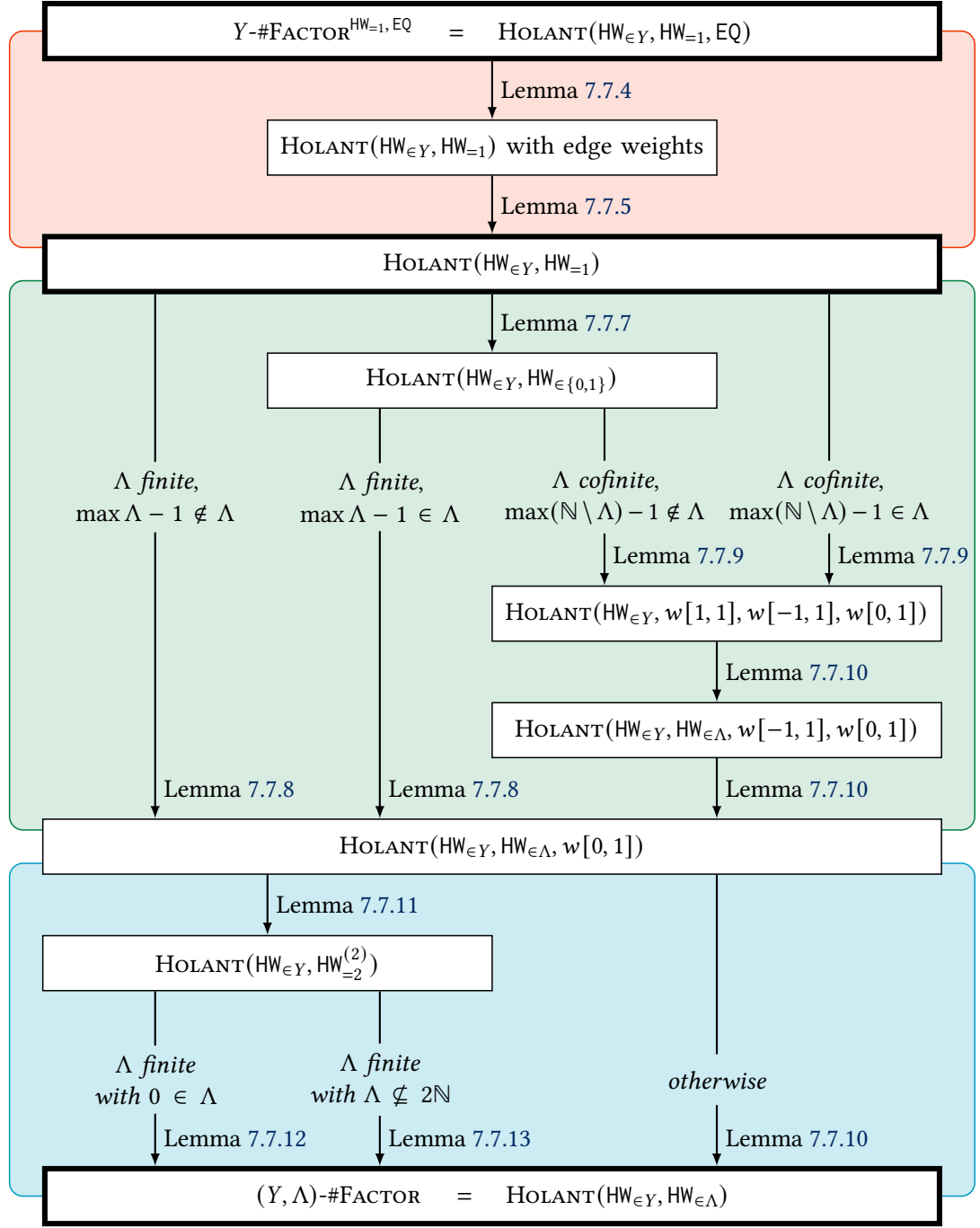


Figure 7.13: An illustration of the chain of reductions in Section 7.7 which we use to prove Main Lemma 7.7.14, that is, the reduction from $Y\text{-}\#\text{FACTOR}^{\text{HW}=1, \text{EQ}}$, which is equal to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{=1}, \text{EQ})$, to $(Y, \Lambda)\text{-}\#\text{FACTOR}$, which is equal to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda})$.

As a next step we remove the edge-weights from the instance. For this we use the reduction from Lemma 7.4.5 which is based on polynomial interpolation.

By an inspection of the proof of Lemma 7.4.5 we can see that the reduction only changes the weighted edges and introduces additional vertices with relation $\text{HW}_{=1}$. Hence, the existence of further vertices with relations does not affect the reduction in any particular way. Thus, we can strengthen Lemma 7.4.5 to obtain Lemma 7.7.5.

Lemma 7.7.5 (Strengthened Version of Lemma 7.4.5). *Let $R_1, \dots, R_d$ be d arbitrary relations for some $d \geq 0$. There is a pathwidth-preserving reduction from $\text{HOLANT}(R_1, \dots, R_d)$ on instances of size n and with r edge weights to unweighted $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{=1})$ operating in time $n^{\mathcal{O}(r)}$. The reduction is such that*

- the size increases by a multiplicative factor of $\mathcal{O}(\log^2(n))$, and
- the total arity increases to at most $\Delta^* + \mathcal{O}(1)$.

Proof. The proof of Lemma 7.4.5 directly works in this setting as well. $\square$

In the following we inductively construct gadgets and describe the number of extensions by sequences. To show that there are no repetitions in these sequences we first prove an auxiliary result in Proposition 7.7.6.

Proposition 7.7.6. *Given three non-negative constants F_0, F_1 , and F_2 with $F_0 \cdot F_2 \neq (F_1)^2$ and $F_0, F_1 \neq 0$. Consider two sequences $\{A_n\}_{n \in \mathbb{N}}$ and $\{B_n\}_{n \in \mathbb{N}}$ that satisfy*

$$\begin{bmatrix} A_n \\ B_n \end{bmatrix} = M \cdot \begin{bmatrix} A_{n-1} \\ B_{n-1} \end{bmatrix} = M^n \cdot U \quad \text{where} \quad M = \begin{bmatrix} F_0 & F_1 \\ F_1 & F_2 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} A_0 \\ B_0 \end{bmatrix} = \begin{bmatrix} F_0 \\ F_1 \end{bmatrix}.$$

Then, the sequence $\{A_n/B_n\}_{n \in \mathbb{N}}$ does not contain any repetitions.

Proof. Suppose that the sequence $\{A_n/B_n\}_{n \in \mathbb{N}}$ contains a repetition such that for two positive integers ℓ and r we have $A_\ell/B_\ell = A_{\ell+r}/B_{\ell+r}$.

Since F_0 and F_1 are positive, and F_2 is non-negative, both sequences $\{A_n\}_{n \in \mathbb{N}}$ and $\{B_n\}_{n \in \mathbb{N}}$ are non-zero. Thus, there is some rational value $k \in \mathbb{Q}$ such that

$$k \cdot \begin{bmatrix} A_\ell \\ B_\ell \end{bmatrix} = \begin{bmatrix} A_{\ell+r} \\ B_{\ell+r} \end{bmatrix}.$$

Now observe that, by the choice of the matrix M and the vector U , we have that $U = M \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Hence, the definition of the sequences yields

$$k \cdot M^{\ell+1} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = k \cdot M^\ell \cdot U = k \cdot \begin{bmatrix} A_\ell \\ B_\ell \end{bmatrix} = \begin{bmatrix} A_{\ell+r} \\ B_{\ell+r} \end{bmatrix} = M^{\ell+r} \cdot U = M^{\ell+r+1} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

As the determinant of M is non-zero by assumption, the matrix M is invertible which yields $k \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = M^r \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Thus, the vector $U' := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ is an eigenvector of the matrix M^r .

We show the following claim to arrive at a contradiction.

Claim 7.7.6.1. *For all $r \geq 1$, it holds that $M^r[2, 1] > 0$.*

Proof of Claim. We prove the claim by induction on r . For $r = 1$ this directly follows as $F_0 \neq 0$ by assumption.

Now, inductively assume that the claim holds for some arbitrary but fixed r . By the definition of the matrix multiplication and the fact that all entries of M are non-negative we know that $M^{r+1}[2, 1] \geq M^r[2, 1] \cdot M[1, 1]$. Since F_0 is positive by assumption, the claim immediately follows. $\triangleleft$

As the values F_0, F_1 , and F_2 are non-negative, the matrix M^r consists only of non-negative entries. Hence, we know that $(M^r \cdot U')[2] > 0$ and thus $M^r \cdot U' \neq k \cdot U'$. Therefore, there are no repetitions in the sequence, which concludes the proof. $\square$

Step 2. As a next step, we replace the $\text{HW}_{=1}$ -relations by gadgets based on simple vertices and relations that force a single edge to be selected, that is, we use simple vertices and $\text{HW}_{=1}^{(1)}$ -relations.

However, this replacement is not always directly possible. As an example, consider the case when $\Lambda = \{0, 1, 2, 3, 4\}$. Even if we can force edges to be selected, it is not clear how to design a gadget that realizes $\text{HW}_{=1}$. Instead, we can easily realize $\text{HW}_{\in\{0,1\}}$ by forcing three edges to a vertex.

Unfortunately, we cannot simply replace all $\text{HW}_{=1}$ -relations by $\text{HW}_{\in\{0,1\}}$ -relations as this would introduce many new solutions. Instead, we use the following Turing-reduction which attaches paths to the vertices and then uses interpolation techniques to recover the original values.

Lemma 7.7.7. *Let $R_1, \dots, R_d$ be d arbitrary relations for some $d \geq 0$. There is a pathwidth-preserving reduction from $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{=1})$ to $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{\in\{0,1\}})$.*

Proof. Suppose we are given an instance H of $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{=1})$. Denote by U the set of all complex vertices of H with $\text{HW}_{=1}$ as relation.

We construct a new instance H' where we replace the relation of the vertices in U by the relation $\text{HW}_{\in\{0,1\}}$. Then, we can think of the value $\text{Holant}(H')$ as a sum over different Holants corresponding to the choice of assigning either $\text{HW}_{=0}$ or $\text{HW}_{=1}$ as relation to a vertex from U .

More formally, we define a_i as the number of solutions of H' where precisely i vertices from U are incident to a selected edge and the remaining $|U| - i$ vertices in U are not incident to any selected edge. We get that $\text{Holant}(H') = \sum_{i=0}^{|U|} a_i$. Then, the value of $a_{|U|}$ corresponds to the number of solutions of H , that is, we have $a_{|U|} = \text{Holant}(H)$.

In the remainder we find these values a_i . For an integer $\ell \geq 1$, we denote by H'_ℓ a graph constructed from H' where we attach a path on ℓ vertices with $\text{HW}_{\in\{0,1\}}$ as relation to every vertex in U , that is, the vertex in U serves as the first vertex of the path and is followed by ℓ other vertices.

We define $P_1(\ell)$ as the number of solutions when the leading edge of the path is selected. Similarly, we define $P_0(\ell)$ as the number of solutions when the leading edge of the path is *not* selected.

From the construction of the path and the $\text{HW}_{\in\{0,1\}}$ -relations, every vertex of the path can be incident to at most one selected edge in a solution. Thus, for all $\ell \geq 0$, we get

$$\begin{aligned} P_0(\ell) &:= P_0(\ell - 1) + P_1(\ell - 1) && \text{with } P_0(0) := 1, \quad \text{and} \\ P_1(\ell) &:= P_0(\ell - 1) && \text{with } P_1(0) := 1. \end{aligned}$$

Next, we analyze the number of solutions of H'_ℓ . The previous observation yields

$$\begin{aligned} \text{Holant}(H'_\ell) &= \sum_{i=0}^{|U|} a_i \cdot (P_0(\ell))^i \cdot (P_0(\ell) + P_1(\ell))^{|U|-i} \\ &= (P_0(\ell) + P_1(\ell))^{|U|} \cdot \sum_{i=0}^{|U|} a_i \cdot \left(\frac{P_0(\ell)}{P_0(\ell) + P_1(\ell)} \right)^i \\ &= (P_0(\ell + 1))^{|U|} \cdot \sum_{i=0}^{|U|} a_{|U|-i} \cdot \left(\frac{P_1(\ell + 1)}{P_0(\ell + 1)} \right)^i. \end{aligned} \tag{7.7.1}$$

If we replace $P_1(\ell + 1)/P_0(\ell + 1)$ in Equation (7.7.1) by an indeterminate x , we can treat $\text{Holant}(H'_\ell)$ as a polynomial P in x of degree at most $|U|$.

With the oracle-access to $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{\in\{0,1\}})$, we can compute $\text{Holant}(H'_\ell)$ for $|U|$ choices of ℓ . When we set $M := \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ and $U := \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, we directly see that, according to Proposition 7.7.6, the matrix M and the vector U generate the sequences $\{P_0(\ell)\}_{\ell \in \mathbb{N}}$ and $\{P_1(\ell)\}_{\ell \in \mathbb{N}}$ and $P_0(\ell), P_1(\ell) > 0$. Hence, it follows from Proposition 7.7.6 that $(P_0(\ell)/P_1(\ell))$ has no repetitions. Thus, the sequence $P_1(\ell)/P_0(\ell)$ always takes different values, too.

With this observation, we can evaluate the polynomial P for $|U| + 1$ different choices of x by choosing $|U| + 1$ different values of ℓ and compute $\text{Holant}(H'_\ell)$.

Then, we can use Fact 2.1.7 to recover the polynomial P which, in particular, also recovers the values for all a_i . Note that we can take care of the additional factor of $P_0(\ell + 1)^{|U|}$ easily during the process.

By the construction of H'_ℓ , we only attach paths to the vertices from U . Thus, the pathwidth of the graph H'_ℓ increases by at most 2 compared to the graph H which concludes the proof. $\square$

Before we proceed with the next steps, we define a new type of relation. For all $x, y \in \mathbb{Z}$, we define $w[x, y]$ as a relation of arity 1 and the following signature f when we denote by e the dangling edge:

$$f(e) := \begin{cases} x & \text{if } e \text{ is not selected,} \\ y & \text{if } e \text{ is selected.} \end{cases}$$

Observe that $w[0, 1]$ corresponds to $\text{HW}_{=1}^{(1)}$ and $w[1, 1]$ corresponds to $\text{HW}_{\in\{0,1\}}^{(1)}$ which is equal to EQ_1 . In the following constructions we additionally use $w[-1, 1]$. We use the notation $w[x, y]$ in the following wherever possible.

For the first case we consider finite sets.

Lemma 7.7.8. *Let $R_1, \dots, R_d$ be d arbitrary relations for some $d \geq 0$.*

For every finite set $\Lambda \subseteq \mathbb{N}$, there is a pathwidth-preserving reduction

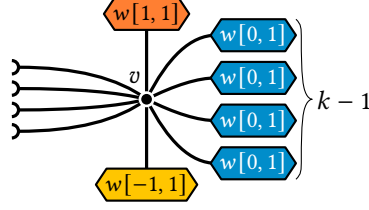


Figure 7.14: An illustration of the gadget Q from Lemma 7.7.9. Here shown for $\Lambda = \{4, 6, 7, 8, 9, \dots\}$ with $k = \max(\mathbb{N} \setminus \Lambda) = 5$ such that Q realizes $\text{HW}_{=1}$.

- from $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{\in\{0,1\}})$ if $\max \Lambda - 1 \in \Lambda$, or
- from $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{=1})$ if $\max \Lambda - 1 \notin \Lambda$,

to $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{\in\Lambda}, w[0, 1])$ increasing the degrees of the vertices with $\text{HW}_{\in\{0,1\}}$ or $\text{HW}_{=1}$ as signature by at most $\max \Lambda$.

Proof. We use the same construction for both cases.

For this we first construct a gadget H that realizes $\text{HW}_{\in\{0,1\}}$ if $\Lambda - 1 \in \Lambda$ and realizes $\text{HW}_{=1}$ if $\Lambda - 1 \notin \Lambda$.

The gadget Q consists of a single vertex v to which we force $\max \Lambda - 1$ edges by adding the same number of complex vertices with relation $w[0, 1]$ to it. Depending on whether $\max \Lambda - 1 \in \Lambda$ or not, this gadget Q immediately realizes $\text{HW}_{\in\{0,1\}}$ or $\text{HW}_{=1}$.

Now, assume we are given an instance H of $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{\in\{0,1\}})$ or an instance of $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{=1})$ if $\max \Lambda - 1 \notin \Lambda$. We replace all complex vertices in H with $\text{HW}_{=1}$ or $\text{HW}_{\in\{0,1\}}$ as relation (depending on Λ) by the gadget Q to obtain an equivalent instance G of $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{\in\Lambda}, w[0, 1])$.

Since we are only adding single vertices to the vertices from H , the pathwidth increases by at most 1. $\square$

The next result is essentially the counterpart of Lemma 7.7.8 for the cofinite setting. As we cannot use the finiteness of the set Λ anymore, we use different relations of arity one instead.

Lemma 7.7.9. *Let $R_1, \dots, R_d$ be d arbitrary relations for some $d \geq 0$.*

For every cofinite set $\Lambda \subseteq \mathbb{N}$ with $\Lambda \neq \mathbb{N} \setminus \{0\}$, there is a pathwidth-preserving reduction

- from $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{=1})$ if $\max(\mathbb{N} \setminus \Lambda) - 1 \in \Lambda$, or
- from $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{\in\{0,1\}})$ if $\max(\mathbb{N} \setminus \Lambda) - 1 \notin \Lambda$,

to $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{\in\Lambda}, w[1, 1], w[-1, 1], w[0, 1])$ with running time $(\text{top } \Lambda)^{\mathcal{O}(1)} \cdot n^{\mathcal{O}(1)}$ such that the total arity increases to $\Delta^* \cdot f(\text{top } \Lambda)$.

Proof. Similar to the case of a finite set in Lemma 7.7.8, we use the same construction for both cases. For ease of notation, we set $k := \max \mathbb{N} \setminus \Lambda$. We directly get that $k \notin \Lambda$ but $k + 1 \in \Lambda$.

We first create the following gadget Q which either realizes $\text{HW}_{=1}$ if $k - 1 \in \Lambda$ or realizes

The constraints are “flipped” compared to Lemma 7.7.8.

$\text{HW}_{\in\{0,1\}}$ if $k - 1 \notin \Lambda$. We refer to Figure 7.14 for an illustration of this gadget.

The gadget Q consists of a simple vertex v , that is, we consider v to have $\text{HW}_{\in\Lambda}$ as relation. Moreover, we create $k - 1$ copies of a vertex with $w[0, 1]$ as relation and make all these vertices adjacent to the vertex v . Additionally, we create one $w[1, 1]$ -vertex and one $w[-1, 1]$ -vertex, and connect both vertices to the vertex v . Finally, we add ℓ dangling edges to v .

Claim 7.7.9.1. *If $k - 1 \in \Lambda$, then Q realizes $\text{HW}_{=1}^{(\ell)}$.*

Proof of Claim. We know that in any solution, the vertex v must be incident to $k - 1$ selected edges because of the $w[0, 1]$ -relations. Since $k \notin \Lambda$, the vertex v cannot be incident to exactly one more selected edge.

Now we consider the following three cases.

- Suppose no dangling edge is selected. Then, we can extend the solution by either selecting none of the edges to the vertices with $w[1, 1]$ and $w[-1, 1]$ (as $k - 1 \in \Lambda$) or by selecting both of these edges (as $k + 1 \in \Lambda$).

The weighted sum of these two solutions is $(-1) + 1 = 0$ and therefore, they do not contribute to the final solution count. That is, they contribute with weight 0 and thus, they can be ignored (or treated as invalid).

- Suppose exactly one dangling edge is chosen. Then, we must pick *at least one* of the edges to the $w[1, 1]$ - and $w[-1, 1]$ -vertices, that is, selecting none of the two edges is not allowed. Therefore, the number of solutions is $1 - 1 + 1 = 1$.
- Suppose more than one dangling edge is selected. Then, we can select any combination of the edges to the vertices with $w[1, 1]$ and $w[-1, 1]$ as relation. Hence, the number of solutions is $1 - 1 + 1 - 1 = 0$. As before, this does not contribute to the final solution.

We conclude that the only way that does not contribute 0 to the final count is when exactly one dangling edge is selected. Hence, the gadget realizes $\text{HW}_{=1}^{(\ell)}$. $\triangleleft$

Now we consider the other case.

Claim 7.7.9.2. *If $k - 1 \notin \Lambda$, then Q realizes $\text{HW}_{\in\{0,1\}}^{(\ell)}$.*

Proof of Claim. As for the previous case, vertex v is incident to the $k - 1$ selected edges to the $w[0, 1]$ -vertices. Since $k - 1 \notin \Lambda$, and $k \notin \Lambda$ by the choice of k , in a valid solution the vertex v must be incident to at least two more selected dangling edges.

- Suppose no dangling edge is selected. Then, we must select the edges to the vertices $w[1, 1]$ and $w[-1, 1]$ simultaneously. In this case the solution contributes 1.
- Suppose exactly one dangling edge is selected. Then, we must pick *at least one* of the edges to the vertices with $w[1, 1]$ and $w[-1, 1]$ as relation. Therefore, the number of solutions is $1 - 1 + 1 = 1$.

- Suppose more than one dangling edge is selected. Then, we can pick any combination of the edges to the $w[1, 1]$ - or $w[-1, 1]$ -vertices. Thus, the number of solutions is $1 - 1 + 1 - 1 = 0$ and the solution is treated as invalid.

We conclude that the only solutions that contribute non-zero are the ones where one or zero dangling edges are selected. Thus, the gadget realizes $\text{HW}_{\in\{0,1\}}$. $\blacktriangleleft$

As we only add single vertices to the existing vertices of H , the pathwidth increases by at most 1. Moreover, the size of the instance increases by a factor of at most $\text{top } \Lambda$ as we might have to add these vertices to every vertex from H . $\square$

Note that the omitted case $\Lambda = \mathbb{N} \setminus \{0\}$ corresponds to $\# \text{EDGECOVER}$ for which we refer to Section 7.9.

Next, we show that we can realize $w[x, y]$. In particular, we apply the following reduction to the relations $w[1, 1]$, $w[-1, 1]$, and $w[0, 1]$ that are introduced by Lemma 7.7.9.

Lemma 7.7.10. *Let $\Lambda \subseteq \mathbb{N}$ denote a non-empty set that is (1) finite with $0 \in \Lambda$ and $\Lambda \not\subseteq 2\mathbb{N}$ or (2) cofinite. Let $R_1, \dots, R_d$ be d arbitrary relations for some $d \geq 0$. The following holds for arbitrary values $x, y \in \mathbb{R}$.*

There is a Turing-reduction from $\text{HOLANT}(R_1, \dots, R_d, w[x, y])$ to $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{\in\Lambda})$ with running time $f(\text{top } \Lambda) \cdot n^{\mathcal{O}(1)}$ such that the total arity Δ^* decreases and the pathwidth increases from w to $w + \Delta^* \cdot f(\text{top } \Lambda)$.

Proof. Consider an instance G of $\text{HOLANT}(R_1, \dots, R_d, w[x, y])$. Let U denote the set of vertices with $w[x, y]$ in G . We define a_i as the number of possible solutions in G where for exactly i of the $w[x, y]$ -vertices the dangling edge is selected and for the other $|U| - i$ vertices the dangling edge is not selected.

Then, we have

$$\text{Holant}(G) = \sum_{i=0}^{|U|} a_i \cdot x^{|U|-i} \cdot y^i. \quad (7.7.2)$$

For a new parameter ℓ , we construct graphs G_ℓ from G where we replace each vertex v with relation $w[x, y]$ and neighbor v_u by a gadget H_ℓ which has exactly one dangling edge. We connect this dangling edge to the neighbor u_v of v . The construction of H_ℓ is given later as it depends on Λ .

We define the gadget H_ℓ such that it satisfies certain properties. Let $h_0(\ell)$ denote the number of solutions for H_ℓ when the dangling edge is not selected. Similarly, let $h_1(\ell)$ denote the number of solutions for H_ℓ when the dangling edge is selected. We have that

$$\text{Holant}(G_\ell) = \sum_{i=0}^{|U|} a_i \cdot h_0(\ell)^{|U|-i} \cdot h_1(\ell)^i = (h_0(\ell))^{|U|} \cdot \sum_{i=0}^{|U|} a_i \cdot \left(\frac{h_1(\ell)}{h_0(\ell)} \right)^i.$$

Assume we can find at least $|U| + 1$ values for ℓ such that for all values the ratios $h_0(\ell)/h_1(\ell)$ and thus, the ratios $h_1(\ell)/h_0(\ell)$ are pairwise different. After computing $\text{Holant}(G_\ell)$ for these

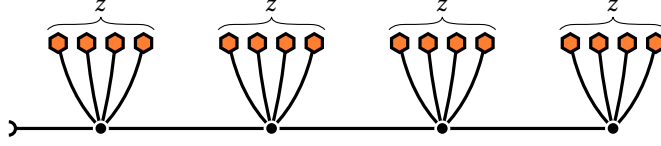


Figure 7.15: The gadget H_ℓ from Claims 7.7.10.1 and 7.7.10.3 for $\ell = 4$ where the value of z depends on the claim.

The hexagons represent vertices with EQ as relation and black circles represent vertices with $\text{HW}_{\in \Lambda}$ as relation.

values of ℓ we can recover the value of each a_i by Fact 2.1.7. Once we know the values for each a_i , we compute the value of $\text{Holant}(G)$ by Equation (7.7.2).

It remains to construct the gadgets H_ℓ and to find the values for ℓ . We construct the gadgets in a way such that there are non-negative constants F_0 , F_1 , and F_2 only depending on Λ with

$$\begin{aligned} h_0(\ell) &\coloneqq F_0 \cdot h_0(\ell - 1) + F_1 \cdot h_1(\ell - 1) & h_0(1) &\coloneqq F_0 \neq 0, \\ h_1(\ell) &\coloneqq F_1 \cdot h_0(\ell - 1) + F_2 \cdot h_1(\ell - 1) & h_1(1) &\coloneqq F_1 \neq 0 \quad \text{and} \\ &F_1^2 \neq F_0 \cdot F_2. \end{aligned} \quad (7.7.3)$$

If the constructed gadget H_ℓ satisfies the three requirements from Equation (7.7.3), then we can use Proposition 7.7.6 to find sufficiently (actually infinitely) many values for ℓ to apply the interpolation from Fact 2.1.7.

For the construction of the gadget H_ℓ , we consider four different cases. We start with two cases for finite sets and afterward consider two cases for cofinite sets.

Claim 7.7.10.1. *If Λ is finite with $0 \in \Lambda$ and $1 \in \Lambda$, then the claimed gadget H_ℓ exists and satisfies Equation (7.7.3).*

Proof of Claim. We define $z \geq 0$ such that $z + 2$ is the smallest integer not in Λ , which exists because Λ is finite.

We define a path on ℓ simple vertices and make the first vertex adjacent to the dangling edge of the gadget H_ℓ . Then, we make each vertex of this path adjacent to z simple vertices (we refer to them as pendant vertices in the following) Recall that simple vertices are assigned the relation $\text{HW}_{\in \Lambda}$.

We refer to Figure 7.15 for an illustration of the gadget.

When we write $k \coloneqq 2^z$, the construction of the gadget yields

$$\begin{aligned} h_0(\ell) &\coloneqq k \cdot h_0(\ell - 1) + k \cdot h_1(\ell - 1) & \text{with } h_0(1) &\coloneqq k \quad \text{and} \\ h_1(\ell) &\coloneqq k \cdot h_0(\ell - 1) + (k - 1) \cdot h_1(\ell - 1) & \text{with } h_1(1) &\coloneqq k. \end{aligned}$$

Indeed, when the first vertex of the path, say v , is incident to at most one selected edge of the path, then we can select any combination of the z edges between v and a pendant vertex. Otherwise, that is, when vertex v is incident to two selected edges, then it is not possible to select all attached edges because $z + 2 \notin \Lambda$.

Since $z \geq 0$, we have $k \geq 1$. With $F_0 \coloneqq F_1 \coloneqq k$ and $F_2 \coloneqq k - 1$, the gadget H_ℓ directly satisfies the constraints in Equation (7.7.3) and thus, this concludes the proof. $\triangleleft$

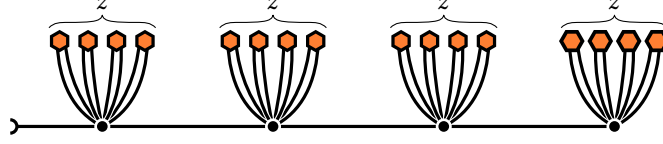


Figure 7.16: The gadget H_ℓ from Claims 7.7.10.2 and 7.7.10.4 for $\ell = 4$ where the value of z depends on the result.

The hexagons represent vertices with EQ as relation and black circles represent vertices with $\text{HW}_{\in \Lambda}$ as relation.

For the next case we still assume that Λ is finite and $0 \in \Lambda$. However, now we assume that $1 \notin \Lambda$, but Λ contains some other odd number.

Claim 7.7.10.2. *If Λ is finite with $0 \in \Lambda$ and $1 \notin \Lambda$ but Λ contains some odd number, then the claimed gadget H_ℓ exists and satisfies Equation (7.7.3).*

Proof of Claim. In this case, we first observe that we can realize EQ_2 . Let b be the smallest non-zero number in the list Λ that is, we set $b := \min(\Lambda \setminus \{0\})$. Then, we create a clique with $b + 1$ (simple) vertices with relation $\text{HW}_{\in \Lambda}$, and split one edge into two dangling edges. This realizes EQ_2 .

The gadget H_ℓ consists of a path on ℓ simple vertices where the first vertex is incident to the dangling edge. Moreover, for an integer z to be determined later, we make each vertex of this path adjacent to z vertices with relation EQ_2 by two edges. We refer to Figure 7.16 for an illustration of this construction.

From the construction of the gadget H_ℓ , we get

$$F_0 := \sum_{i \geq 0: 2i \in \Lambda} \binom{z}{i}, \quad F_1 := \sum_{i \geq 0: 2i+1 \in \Lambda} \binom{z}{i}, \quad \text{and} \quad F_2 := \sum_{i \geq 0: 2i+2 \in \Lambda} \binom{z}{i}.$$

Indeed, for $j \in [0..2]$, we can interpret F_j as the number of ways a vertex v that is incident to j selected edges of the path can receive a valid number of selected pairs of edges going to a vertex with EQ_2 as relation.

Moreover, since Λ contains some odd integer by assumption, and some even integer as $0 \in \Lambda$, we immediately get that $F_0, F_1 \geq 1$.

To find the value for z , we interpret F_0, F_1 , and F_2 as polynomials in z with degree at most $\lfloor \max \Lambda / 2 \rfloor$.

We pick z as the smallest positive integer that is not a root of the polynomial $F_0 \cdot F_2 - (F_1)^2$ over z . As this polynomial has degree at most $\max \Lambda$, it has at most that many roots. Therefore, we get that $z \leq \max \Lambda + 1$ and thus, can find z in time polynomial in $\max \Lambda$. $\blacktriangleleft$

For the remaining two cases we assume that Λ is cofinite. We first consider the case when $0 \notin \Lambda$ or $1 \in \Lambda$. This case follows similarly to Claim 7.7.10.1.

Claim 7.7.10.3. *If Λ is cofinite with $0 \notin \Lambda$ or $1 \in \Lambda$, then the claimed gadget H_ℓ exists and satisfies Equation (7.7.3).*

To remove the seemingly parallel edges, we can place another EQ_2 -vertex on one of the edges.

Proof of Claim. We first realize EQ_1 . For this we consider three different cases.

- If $0, 1 \in \Lambda$, then a single vertex with a dangling edge realizes EQ_1 .
- If $0 \notin \Lambda$ but $1 \in \Lambda$, then attach $\max(\mathbb{N} \setminus \Lambda) + 1$ simple vertices to a vertex v . When we attach a dangling edge to v , this realizes EQ_1 by exploiting that Λ is cofinite.
- If $0, 1 \notin \Lambda$, then take a clique with $\min \Lambda + 1$ vertices and split the edge between two vertices into two dangling edges. This gadget realizes $\text{HW}_{=2}^{(2)}$.

Next, we attach $\lceil (\max(\mathbb{N} \setminus \Lambda) + 1)/2 \rceil$ copies of a vertex with relation $\text{HW}_{=2}^{(2)}$ to a new vertex with one dangling edge. This realizes EQ_1 as claimed, again by using that Λ is cofinite.

With the realization of EQ_1 we can construct the gadget H_ℓ as follows. As before, the gadget contains ℓ simple vertices that are connected as a path, and with a dangling edge on the first vertex. For an integer $z \geq \max X + 1$ that we choose later, attach z vertices with relation EQ_1 to each of the ℓ vertices of the path. See Figure 7.15 for an illustration of this construction.

By the definition of the gadget H_ℓ , we immediately get

$$F_0 := \sum_{\substack{i \geq 0: \\ i \in \Lambda}} \binom{z}{i}, \quad F_1 := \sum_{\substack{i \geq 0: \\ i+1 \in \Lambda}} \binom{z}{i}, \quad \text{and} \quad F_2 := \sum_{\substack{i \geq 0: \\ i+2 \in \Lambda}} \binom{z}{i}.$$

We claim that there is a value for z such that the requirements from Equation (7.7.3) are satisfied. That is, we especially have that $F_0 \cdot F_2 \neq (F_1)^2$.

If we choose $z \geq \max X + 1$, then F_0 and F_1 are never equal to 0.

Now suppose for the sake of a contradiction that $F_0 \cdot F_2 = (F_1)^2$ holds for all choices of z . We first expand the equations above. Then, for every z , we have that

$$\begin{aligned} F_0 \cdot F_2 &= \left(\sum_{\substack{i \geq 0: \\ i \in \Lambda}} \binom{z}{i} \right) \cdot \left(\sum_{\substack{i \geq 0: \\ i+2 \in \Lambda}} \binom{z}{i} \right) = \left(2^z - \sum_{\substack{i \geq 0: \\ i \in X}} \binom{z}{i} \right) \cdot \left(2^z - \sum_{\substack{i \geq 0: \\ i+2 \in X}} \binom{z}{i} \right) \\ &= \left(2^z - \sum_{\substack{i \geq 0: \\ i+1 \in X}} \binom{z}{i} \right)^2 = \left(\sum_{\substack{i \geq 0: \\ i+1 \in \Lambda}} \binom{z}{i} \right)^2 = (F_1)^2. \end{aligned}$$

When we set

$$Q_1(z) := 2 \sum_{i+1 \in X} \binom{z}{i} - \sum_{i \in X} \binom{z}{i} - \sum_{i+2 \in X} \binom{z}{i}$$

and

$$Q_2(z) := \left(\sum_{i+1 \in X} \binom{z}{i} \right)^2 - \left(\sum_{i \in X} \binom{z}{i} \right) \cdot \left(\sum_{i+2 \in X} \binom{z}{i} \right),$$

we get $2^z \cdot Q_1(z) = Q_2(z)$ by rearranging the terms.

For large enough z , we argue that $Q_1(z)$ is not identically zero. Observe that the second term in $Q_1(z)$ gives a non-zero monomial $z^{\max X}$ whereas the other two terms cannot give a monomial of this degree as X is finite.

Now, since Λ is fixed and cofinite, the set X is cofinite. Thus, the two polynomials Q_1 and Q_2 are non-zero and have constant degree. Therefore, the polynomial $Q_1(z)$ is zero only for finitely many values of z .

We conclude that there are infinitely many (positive) values for z such that $Q_1(z)$ is non-zero. Thus, for *each* such z , we have

$$|2^z \cdot Q_1(z)| = |Q_2(z)|.$$

This immediately gives a contradiction since $2^z \in \omega(z^c)$, for any constant c assuming that z is large enough. Thus, there is some positive integral value for z such that $F_0 \cdot F_2 \neq (F_1)^2$.

We use this value of z in the construction of the gadget. As the value of z only depends on X , it can be precomputed and thus, this concludes the proof. $\triangleleft$

For the last step, we consider the omitted case from the previous result where $0 \in \Lambda$ and $1 \notin \Lambda$. This result follows similarly to Claim 7.7.10.2.

Claim 7.7.10.4. *If Λ is cofinite with $0 \in \Lambda$ and $1 \notin \Lambda$, then the claimed gadget H_ℓ exists and satisfies Equation (7.7.3).*

Proof of Claim. We first realize EQ_2 as follows. Create a clique with $\min(\Lambda \setminus \{0\}) + 1$ vertices. When we split an arbitrary edge into two dangling edges, this immediately realizes EQ_2 .

For the gadget H_ℓ , we create a path with ℓ vertices and add a dangling edge to the first vertex. For a value z to be chosen later, we attach z vertices with EQ_2 as relation via both dangling edges to each vertex in the path. See Figure 7.16 for an illustration.

The construction of the gadget immediately yields

$$F_0 := \sum_{i \geq 0: 2i \in \Lambda} \binom{z}{i}, \quad F_1 := \sum_{i \geq 0: 2i+1 \in \Lambda} \binom{z}{i}, \quad \text{and} \quad F_2 := \sum_{i \geq 0: 2i+2 \in \Lambda} \binom{z}{i}.$$

Suppose that for all z it holds that $F_0 \cdot F_2 = (F_1)^2$. Then, for every z , we have

$$\begin{aligned} F_0 \cdot F_2 &= \left(\sum_{2i \in \Lambda} \binom{z}{i} \right) \cdot \left(\sum_{2i+2 \in \Lambda} \binom{z}{i} \right) = \left(2^z - \sum_{2i \in X} \binom{z}{i} \right) \cdot \left(2^z - \sum_{2i+2 \in X} \binom{z}{i} \right) \\ &= \left(2^z - \sum_{2i+1 \in X} \binom{z}{i} \right)^2 = \left(\sum_{2i+1 \in \Lambda} \binom{z}{i} \right)^2 = (F_1)^2. \end{aligned}$$

When we set

$$Q_1(z) = 2 \sum_{2i+1 \in X} \binom{z}{i} - \sum_{2i \in X} \binom{z}{i} - \sum_{2i+2 \in X} \binom{z}{i}$$

and

$$Q_2(z) = \left(\sum_{2i+1 \in X} \binom{z}{i} \right)^2 - \left(\sum_{2i \in X} \binom{z}{i} \right) \cdot \left(\sum_{2i+2 \in X} \binom{z}{i} \right)$$

we get $2^z \cdot Q_1(z) = Q_2(z)$ by rearranging the terms.

As before, the polynomial $Q_1(z)$ is not identically zero for large enough z . Indeed, the first term of $Q_1(z)$ gives a constant of 2, the second term cannot give a constant since $0 \in \Lambda$, and the third term gives a constant of -1 if $2 \in X$.

Thus, the polynomial $Q_1(z)$ has a non-zero constant term and hence, is zero only for finitely many z . Therefore, for infinitely many positive values of z , we have

$$|2^z Q_1(z)| = |Q_2(z)|.$$

This immediately gives a contradiction since $2^z \in \omega(z^c)$, for any constant c , and since Q_1 and Q_2 are polynomials of constant degree. Thus, we can choose a value for z such that $F_0 \cdot F_2 \neq (F_1)^2$. As the value of z only depends on the set Λ , the value can be precomputed and thus, this concludes the proof. $\triangleleft$

This concludes the proof of Lemma 7.7.10 by the application of Claims 7.7.10.1 to 7.7.10.4 for the different cases. $\square$

Step 3. It remains to realize $w[0, 1]$ for the cases where Lemma 7.7.10 is not applicable. Because of parity issues, this might not always be possible, for example, if the set Λ contains only even numbers. Therefore, we first show that it suffices to realize $\text{HW}_{=2}^{(2)}$ which can be seen as two vertices with relation $w[0, 1]$, that is, two vertices with relation $\text{HW}_{=1}^{(1)}$.

Lemma 7.7.11. *Let $R_1, \dots, R_d$ be d arbitrary relations for some $d \geq 0$. There is a pathwidth-preserving reduction from $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{=1}^{(1)})$ to $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{=2}^{(2)})$ leaving the maximum degree unaffected, or increasing it to at most Δ^* .*

Proof. Consider an arbitrary instance G of $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{=1}^{(1)})$.

We consider two cases depending on the parity of the relations $R_1, \dots, R_d$. We first consider the case when not all relations are even. In this case we can realize $\text{HW}_{=1}^{(1)}$ using only existing relations and $\text{HW}_{=2}^{(2)}$.

Claim 7.7.11.1. *If, for some $\delta \in [1..d]$, the relation R_δ is not even, then we can realize $\text{HW}_{=1}^{(1)}$ with relations R_δ and $\text{HW}_{=2}^{(2)}$. The gadget has pathwidth 2 and size at most Δ^* .*

Proof of Claim. Assume that for some $\delta \in [1..d]$, the relation R_δ is not even, that is, the relation R_δ is odd or has undefined parity. In particular, denote by r an arbitrary accepted input of R_δ that is odd, that is, where $\text{hw}(r)$ is odd.

We create a simple vertex v with relation R_δ . We create $\lfloor \text{hw}(r)/2 \rfloor$ vertices with relation $\text{HW}_{=2}^{(2)}$ and make them adjacent to vertex v . Additionally, we create one more vertex with relation $\text{HW}_{=2}^{(2)}$ make it adjacent to v by one edge and use the other edge as the dangling edge of the gadget.

By correctly identifying the incident edges of v , we can ensure that the $\text{HW}_{=2}^{(2)}$ -relations represent the input r for the vertex v . Hence, by the properties of the relations $\text{HW}_{=2}^{(2)}$, the dangling edge must be selected as well in any solution.

The size of the gadget depends on the arity of R_δ , which can be bounded by Δ^* . As the pathwidth of the gadget is 2, this concludes the proof. $\triangleleft$

Now we can replace each vertex with relation $\text{HW}_{=1}^{(1)}$ by the gadget from Claim 7.7.11.1. As the pathwidth of the gadget is constant, and the size only depends on Δ^* , this concludes the reduction in this case.

It remains to analyze the case when all relations R_δ are even, which also covers the setting when $d = 0$. We claim that $\text{Holant}(G)$ is zero if the number of $\text{HW}_{=1}^{(1)}$ -vertices is odd.

Claim 7.7.11.2. *If all relations $R_1, \dots, R_d$ are even, and the number of $\text{HW}_{=1}^{(1)}$ -vertices is odd, then $\text{Holant}(G) = 0$.*

Proof of Claim. Consider an arbitrary solution S for G and sum the degree of all vertices in solution S . As the number of vertices with relation $\text{HW}_{=1}^{(1)}$ is odd, they contribute an odd number to the sum. In contrast, all vertices with a relation R_δ contribute an even number to the sum because all relations are even by assumption.

Thus, the sum of the degrees is odd which contradicts that fact that the sum must be even as every selected edge is counted twice. $\triangleleft$

If Claim 7.7.11.2 applies, we conclude the reduction with a trivial no-instance of the problem.

Otherwise, we know that there is an even number of vertices with $\text{HW}_{=1}^{(1)}$ as relation. In the remainder, we replace pairs of these relations by $\text{HW}_{=2}^{(2)}$. We do this based on the path decomposition to keep pathwidth low.

Denote by $\beta(1), \beta(2), \beta(3), \dots$ the bags of some nice path decomposition of G . We denote by U the set of vertices of G with relation $\text{HW}_{=1}^{(1)}$. We order the vertices in U according to their introduce bag and denote them by $v_1, \dots, v_{|U|}$.

Formally, for every vertex $v_i \in U$, we define $\text{intro}(v_i)$ as the index of the bag introducing the vertex v_i , that is, the index $\text{intro}(v_i)$ is the unique value such that $\{v_i\} = \beta(\text{intro}(v_i)) \setminus \beta(\text{intro}(v_i) - 1)$.

Based on the labeling of the vertices, we have $\text{intro}(v_i) < \text{intro}(v_{i+1})$ for all $i \in [1 \dots |U| - 1]$, where a strict ordering is possible as every introduce node introduces exactly one vertex.

Now we have everything ready to state the reduction. For all $i \in [1 \dots |U|/2]$, we remove the two vertices v_{2i-1} and v_{2i} , and replace them by a new vertex $\hat{v}_i$ with relation $\text{HW}_{=2}^{(2)}$. Moreover, we make vertex $\hat{v}_i$ adjacent to the original neighbors of v_{2i-1} and v_{2i} .

We replace every occurrence of the vertices v_{2i-1} and v_{2i} in the path decomposition by $\hat{v}_i$, and add $\hat{v}_i$ to the bags $\beta(\text{intro}(v_{2i-1})), \dots, \beta(\text{intro}(v_{2i}) - 1)$ to ensure that all bags containing $\hat{v}_i$ form a path.

By assumption, there is no other vertex v_ℓ introduced in the bags $\beta(\text{intro}(v_{2i-1}) + 1), \dots, \beta(\text{intro}(v_{2i}) - 1)$ and hence, the pathwidth increases by at most 1. Further, the degree of the graph does not increase as we can assume, without loss of generality, that it is already at least 2. Otherwise, the instance consists only of pairs of vertices. $\square$

As a last step of the chain of reductions, we now realize $\text{HW}_{=2}^{(2)}$. We first consider the case when $0 \notin \Lambda$. In this case it is easy to realize $\text{HW}_{=2}^{(2)}$ by the approach from the decision version in Lemma 7.6.1.

Lemma 7.7.12. *Let $R_1, \dots, R_d$ be d arbitrary relations for some $d \geq 0$. Let $\Lambda \subseteq \mathbb{N}$ be a fixed set with $0 \notin \Lambda$. There is a polynomial-time many-one reduction from $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{=2}^{(2)})$ to $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{\in \Lambda})$ increasing pathwidth by $\text{top } \Lambda$ and leaving the maximum degree unaffected, or increasing it to $\text{top } \Lambda$.*

Proof. We use the construction from Lemma 7.6.1 to realize $\text{HW}_{=2}^{(2)}$. That is, we use a clique with $\min \Lambda + 1$ vertices where one edge is split into two dangling edges.

We replace each vertex v with relation $\text{HW}_{=2}^{(2)}$ by this gadget and adjust the corresponding path decomposition accordingly. More precisely, we duplicate one bag of the original decomposition and add all vertices of the gadget to it. Then, we replace the remaining occurrences of vertex v by one of the two portal vertices of the gadget.

As the size of the gadget and the degree of the vertices is at most $\text{top } \Lambda$, this concludes the proof. $\square$

For the remaining case we assume that $0 \in \Lambda$ and Λ contains only even numbers as the case when Λ contains odd number is covered by Lemma 7.7.10. The idea is similar to Lemma 7.7.10 except that we duplicate the edges to force two edges to be selected simultaneously.

Lemma 7.7.13. *Let $R_1, \dots, R_d$ be d arbitrary relations for some $d \geq 0$. Let $\Lambda \subseteq \mathbb{N}$ be a fixed, finite set with $\{0\} \subsetneq \Lambda$ that contains only even numbers. There is a pathwidth-preserving reduction from $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{=2}^{(2)})$ to $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{\in \Lambda})$ increasing pathwidth only by a fixed constant and leaving the max degree unaffected, or increasing it to $2 \max B + 6$.*

Proof. We use similar arguments as for Lemma 7.7.10 and basically duplicate edges of the gadget there as the arity of the relation we realize is now doubled. In the following we present the formal reduction.

Consider an instance G of $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{=2}^{(2)})$. Let U denote the set of vertices in G with $\text{HW}_{=2}^{(2)}$ as relation.

For $i, j \in [0 \dots |U|]$ with $i + j \leq |U|$, we define $a_{i,j}$ as the number of edge sets from G that satisfy all relations at the vertices in $V(G) \setminus U$, and exactly i vertices of U are incident to no selected edges, exactly j vertices in U are incident to exactly one selected edge, and the remaining $|U| - i - j$ vertices in U are incident to two selected edges.

By the design of the relation $\text{HW}_{=2}^{(2)}$, we immediately get that $\text{Holant}(G) = a_{0,0}$ as all vertices in U must be incident to two selected edges.

For all $\ell \geq 1$, we construct a gadget H_ℓ with two dangling edges. This gadget uses only vertices with relation $\text{HW}_{\in \Lambda}$. We postpone the formal definition of the gadget H_ℓ , but we assume that it has $h_0(\ell)$ partial solution when no dangling edge is selected, it has $h_1(\ell)$ partial solutions when exactly one dangling edge is selected, and it has $h_2(\ell)$ partial solutions when both dangling edges are selected.

Then, we construct a graph G_ℓ from G where we replace each vertex from U along with its edges by a copy of the gadget H_ℓ and connect the two dangling edges to the endpoints of the

The gadget is defined such that it cannot distinguish between the two edges.

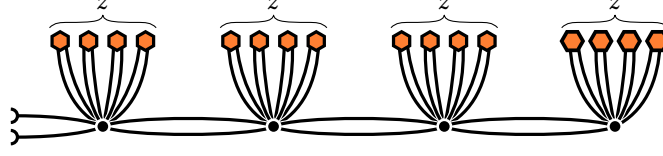


Figure 7.17: The gadget H_ℓ from Lemma 7.7.13 for $\ell = 4$.

The hexagons represent vertices with EQ as relation and black circles represent vertices with $\text{HW}_{\in \Lambda}$ as relation.

replaced edges. With the previous definition of the numbers $a_{i,j}$ and the properties of H_ℓ , we immediately get

$$\text{Holant}(G_\ell) = \sum_{i=0}^{|U|} \sum_{j=0}^{|U|-i} a_{i,j} \cdot h_0(\ell)^i \cdot h_1(\ell)^j \cdot h_2(\ell)^{|U|-i-j}.$$

As the set Λ contains only even numbers, we directly get that $h_1(\ell) = 0$ for all choices of ℓ . This follows by the observation that otherwise the sum of the degrees in any solution for H_ℓ would be odd which is not possible. Hence, we can simplify the previous equation and get

$$\text{Holant}(G_\ell) = \sum_{i=0}^{|U|} a_{i,0} \cdot h_0(\ell)^i \cdot h_2(\ell)^{|U|-i} = (h_2(\ell))^{|U|} \cdot \sum_{i=0}^{|U|} a_{i,0} \cdot \left(\frac{h_0(\ell)}{h_2(\ell)} \right)^i.$$

Assume we can find at least $|U| + 1$ values for ℓ such that for all these values the ratios $h_0(\ell)/h_2(\ell)$ are pairwise different. After computing $\text{Holant}(G_\ell)$ for these values of ℓ , which is possible by the oracle access to $\text{HOLANT}(R_1, \dots, R_d, \text{HW}_{\in \Lambda})$, we can recover the value of each $a_{i,0}$ by Fact 2.1.7. Once we know the values for each $a_{i,0}$, we especially know $a_{0,0}$ which is identical to $\text{Holant}(G)$ by our earlier observation.

It remains to construct the gadgets H_ℓ and to find the values for ℓ . We construct the gadgets in a way such that there are constants F_0 , F_2 , and F_4 only depending on Λ with

$$\begin{aligned} h_0(\ell) &:= F_0 \cdot h_0(\ell - 1) + F_2 \cdot h_2(\ell - 1) & h_0(1) &:= F_0 \neq 0, \\ h_2(\ell) &:= F_2 \cdot h_0(\ell - 1) + F_4 \cdot h_2(\ell - 1) & h_2(1) &:= F_2 \neq 0 \quad \text{and} \\ F_2^2 &\neq F_0 \cdot F_4. \end{aligned} \tag{7.7.4}$$

Observe that the value $h_1(\ell)$ is not present in these constraints, as the requirement that $h_1(\ell) = 0$ follows a priori by the choice of the set Λ and does not need any further treatment.

If the constructed gadget H_ℓ satisfies the three requirements from Equation (7.7.3), then we can use Proposition 7.7.6 to find sufficiently (actually infinitely) many values for ℓ to apply the interpolation from Fact 2.1.7.

Claim 7.7.13.1. *The gadget H_ℓ exists and satisfies Equation (7.7.4).*

Proof of Claim. We use essentially the idea from Claim 7.7.10.2 which covers one case in the proof of Lemma 7.7.10. We refer to Figure 7.17 for an illustration of the following construction.

We first observe that we can realize EQ_2 . Let b be the smallest non-zero number in the set Λ that is, we set $b := \min(\Lambda \setminus \{0\})$. Then, we create a clique with $b + 1$ (simple) vertices with relation $\text{HW}_{\in \Lambda}$, and split one edge into two dangling edges. This realizes EQ_2 .

For the gadget H_ℓ , we create a path on ℓ vertices with relation $\text{HW}_{\in \Lambda}$. Then, we duplicate all edges such they become parallel edges, and we make the first vertex incident to two dangling edges. Moreover, for an integer z to be determined later, we make each vertex of this path adjacent to z vertices with relation EQ_2 by two edges. We refer to Figure 7.17 for an illustration of this construction.

To remove parallel edges, we can place EQ_2 -vertices on these edges.

From the construction of the gadget H_ℓ , we get

$$F_0 := \sum_{i \geq 0: 2i \in \Lambda} \binom{z}{i}, \quad F_2 := \sum_{i \geq 0: 2i+2 \in \Lambda} \binom{z}{i}, \quad \text{and} \quad F_4 := \sum_{i \geq 0: 2i+4 \in \Lambda} \binom{z}{i}.$$

Indeed, for $j \in [0 \dots 4]$, we can interpret F_j as the number of ways a vertex v that is incident to j selected edges of the path (with parallel edges) can receive a valid number of selected pairs of edges going to a vertex with EQ_2 as relation.

Naturally, as the set Λ contains only even numbers, we get $F_1 = F_3 = 0$. Indeed, since Λ does not contain any odd number, the terminal vertex must be incident to an even number of selected edges. As this applies to the other vertices as well, there is no solution where a vertex of the path is incident to an odd number of selected edges from the path.

Moreover, since Λ contains 0 and some even number different from 0, we immediately get that $F_0, F_2 \geq 1$.

To find the value for z , we interpret F_0, F_2 , and F_4 as polynomials in z . Each of them has degree at most $\max \Lambda / 2$ as the set Λ is finite. Therefore, the polynomial $F_0 \cdot F_2 - (F_1)^2$ has degree at most $\max \Lambda$ and thus, at most that many roots. Hence, we can pick z as the smallest positive integer that is not a root of the polynomial. We conclude that $z \leq \max \Lambda + 1$ and thus, can find z in time polynomial in $\max \Lambda$ by iterating over all possible values. $\triangleleft$

This concludes the proof of Lemma 7.7.13 by our earlier observations. $\square$

Combining the Results. Now we have everything ready to prove the main result of this section, that is, the reduction from $Y\text{-}\#\text{FACTOR}^{\text{Rel}}$ to $(Y, \Lambda)\text{-}\#\text{FACTOR}$. We again refer to Figure 7.13 for an illustration of the outline of the proof.

Main Lemma 7.7.14. *Let $\Lambda \subseteq \mathbb{N}$ be a non-empty set with $\Lambda \notin \{\{0\}, \mathbb{N} \setminus \{0\}, \mathbb{N}\}$ that is finite or cofinite. Let $Y \subseteq \mathbb{N}$ be an arbitrary set that might even be given as input.*

There is a Turing-reduction from $Y\text{-}\#\text{FACTOR}^{\text{HW}=1, \text{EQ}}$ to $(Y, \Lambda)\text{-}\#\text{FACTOR}$ with running time $f(\text{top } \Lambda + \Delta^*) \cdot n^{O(1)}$, increasing the size by a factor of $f(\text{top } \Lambda)$, decreasing Δ^* to zero, and increasing the width of the path decomposition by an additive term of $\Delta^* \cdot f(\text{top } \Lambda)$ for each oracle call.

Proof. We first note that the problems $Y\text{-}\#\text{FACTOR}^{\text{HW}=1, \text{EQ}}$ and $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{=1}, \text{EQ})$ are identical, and the problems $(Y, \Lambda)\text{-}\#\text{FACTOR}$ and $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda})$ are identical. Thus, it suffices to give the claimed reduction from $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{=1}, \text{EQ})$ to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda})$. We do this in three steps.

Step 1. We start with an instance of $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{=1}, \text{EQ})$ and use Lemma 7.7.4 to remove the equality relations. Then, we apply Lemma 7.7.5 to the resulting instance of edge-weighted $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{=1})$ to reduce to unweighted $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{=1})$.

Step 2. Next, we reduce from $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{=1})$ to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda}, w[0, 1])$. We consider four cases for this reduction.

- If the set Λ is finite but $\max \Lambda - 1 \notin \Lambda$, then we reduce to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda}, w[0, 1])$ by Lemma 7.7.8.
- If the set Λ is finite but $\max \Lambda - 1 \in \Lambda$, then we first use Lemma 7.7.7 to reduce to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \{0,1\}})$. Afterward, we reduce to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda}, w[0, 1])$ by Lemma 7.7.8.
- If the set Λ is cofinite but $\max(\mathbb{N} \setminus \Lambda) \notin \Lambda$, then we first apply Lemma 7.7.7 to reduce to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \{0,1\}})$.
Afterward, we reduce to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda}, w[1, 1], w[-1, 1], w[0, 1])$ in Lemma 7.7.9. Then, we use Lemma 7.7.10 to reduce to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda}, w[-1, 1], w[0, 1])$ and finally, use Lemma 7.7.10 once more to reduce to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda}, w[0, 1])$.
- If the set Λ is cofinite with $\max(\mathbb{N} \setminus \Lambda) \in \Lambda$, then we directly use Lemma 7.7.9 once and apply Lemma 7.7.10 twice to reduce to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda}, w[0, 1])$.

Step 3. As the last step, we realize $w[0, 1]$, that is, we reduce $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda}, w[0, 1])$ to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda})$. Again, we consider different cases.

- If (1) Λ is cofinite, or (2) Λ is finite with $0 \in \Lambda$ and $\Lambda \not\subseteq 2\mathbb{N}$, then we use Lemma 7.7.10 to reduce to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda})$.
- Otherwise, we first use Lemma 7.7.11 to reduce to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda}, \text{HW}_{=2}^{(2)})$.

Now, if $0 \notin \Lambda$, then we use Lemma 7.7.12 to reduce to $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda})$.

If $0 \in \Lambda$, it remains to check the case when $\Lambda \subseteq 2\mathbb{N}$ as the other cases are covered by the first case of Step 3. Since we know that Λ contains only even numbers and some even number different from 0, we use Lemma 7.7.13 to conclude the proof.

It remains to analyze the running time and the change of the pathwidth of this reduction.

Given an instance of $Y\text{-}\#\text{FACTOR}^{\text{HW}_{=1}, \text{EQ}}$, our chain of reductions constructs a polynomial number of instances of $\text{HOLANT}(\text{HW}_{\in Y}, \text{HW}_{\in \Lambda})$. Moreover, in each of these instances the pathwidth is affected only by an additive term bounded by some function of Δ^* and $\text{top } \Lambda$. $\square$

7.8

Special Case:

Counting Version Parameterized by List Size

In this section, we prove the lower bound and $\#W[2]$ -hardness for $\#\text{ANTIFACTOR}^{\text{Rel}}_1$ parameterized by treewidth, that is, we prove Main Lemma 7.8.4.

Main Lemma 7.8.4. *There exists a constant c such that there is no $\mathcal{O}(n^{w-c})$ algorithm for $\#ANTIFACTOR^{Rel}_1$ on n -vertex graphs with $\Delta^* \in \mathcal{O}(1)$, even if only one set is used and we are given a path decomposition of width w , unless $\#SETH$ fails.*

Moreover, $\#ANTIFACTOR^{Rel}_1$ is $\#W[2]$ -hard parameterized by pathwidth.

For this bound we do not use the previous results but provide a self-contained reduction. Moreover, our main reduction starts from COUNTING k -SET COVERS ($\#SETCOVER$) although, we provide a lower bound for $\#SETCOVER$ under $\#SETH$.

Note that counting set covers of size at most k is equivalent to counting set covers of size exactly k .

Definition 7.8.1 ($\#SETCOVER$). *Given a family $\mathcal{F}$ of n sets from a universe of U of size m , the task is to find the size of the set*

$$\{S \subseteq \mathcal{F} \mid |S| \leq k \wedge \forall u \in U \exists S \in S: u \in S\},$$

that is, count the selections of subsets of $\mathcal{F}$ such that every element of the universe is hit by some selected set.

We start with two easy lower bounds for $\#SETCOVER$.

Lower Bounds for $\#SETCOVER$. We use the following result for $\#SETCOVER$ as an initial step of our proof. We adjust the proof for COUNTING DOMINATING SETS from Lemma 2.1 in [111] in a straightforward manner to prove Lemma 7.8.2.

Lemma 7.8.2. *For each fixed $k \geq 2$, the following holds. If there exists an $\varepsilon > 0$ such that there is an $\mathcal{O}(m \cdot n^{k-\varepsilon})$ -time algorithm for $\#SETCOVER$ on instances with n sets over a universe of size $m = \text{poly}(\log n)$, then $\#SETH$ fails.*

Proof. We use the ideas from the proof of Lemma 2.1 in [111] which covers the corresponding result for COUNTING DOMINATING SETS.

Consider a given SAT-formula ϕ with n variables and $m = \text{poly}(n)$ clauses $C_1, \dots, C_m$. As a first step, we partition the variables of ϕ into k groups of equal size. We denote these groups by $V_1, \dots, V_{n/k}$.

We assume without loss of generality that k divides n .

Second, as universe for the new instance, we define $U := [1..m] \cup \{v_1, \dots, v_k\}$. Intuitively, the integer values in $[1..m]$ represent the clauses and the elements v_i correspond to the i th group V_i .

Now, for all $i \in [1..k]$, and all assignments $\pi_i: V_i \rightarrow \{0, 1\}$, we define the set $F_{\pi_i}^i \subseteq U$ as

$$F_{\pi_i}^i := \{v_i\} \cup \{j \in [1..m] \mid \pi_i \text{ satisfies clause } C_j\}.$$

We denote by $\mathcal{F}$ the family of all sets $F_{\pi_i}^i$.

We claim that our reduction is correct.

Claim 7.8.2.1. *The number of solutions for ϕ and $(\mathcal{F}, U)$ are equal.*

Proof of Claim. Given a satisfying assignment $\pi: V(\phi) \rightarrow \{0, 1\}$, we can select the set $F_{\pi[V_i]}^i$ for all $i \in [1..k]$ to cover all elements from the universe.

Now consider a solution for $(\mathcal{F}, U)$ of size at most k . We first observe that for each $i \in [1 \dots k]$, we must select at least one set $F_{\pi_i}^i$. Moreover, as we select at most k sets, for each $i \in [1 \dots k]$, precisely one set $F_{\pi_i}^i$ must be selected.

We claim that the assignment determined by $\pi_1, \dots, \pi_k$ satisfies the formula ϕ . Indeed, as every index $j \in [1 \dots m]$ must be contained in some set $F_{\pi_i}^i$ and the index is only contained in π_i satisfies clause C_j , the claim follows. $\square$

Before we conclude the proof, we observe that there are $k \cdot 2^{n/k}$ sets in $\mathcal{F}$ and the universe U has size $k + m \in k + \text{poly}(n)$. Hence, we especially have that $|U| \in \text{poly}(\log|\mathcal{F}|)$.

Therefore, an algorithm solving #SETCOVER in the claimed running time, would solve SAT in time

$$\begin{aligned} \mathcal{O}((k + m) \cdot (k \cdot 2^{n/k})^{k-\epsilon}) &\leq \mathcal{O}(m \cdot k^k \cdot 2^{n/k \cdot (k-\epsilon)}) \\ &\leq \mathcal{O}(m \cdot k^k \cdot 2^{n(1-\epsilon/k)}) \end{aligned}$$

which contradicts #SETH as the value of k is fixed and thus does not depend on the formula ϕ . $\square$

Additionally, we use that #SETCOVER is #W[2]-hard parameterized by k .

Lemma 7.8.3. *#SETCOVER is #W[2]-hard parameterized by k .*

Proof. We know that COUNTING DOMINATING SETS is #W[2]-hard. See, for example, Theorem 19 in [51].

We use the standard reduction from DOMINATING SET to SET COVER to show that the latter problem is #W[2]-hard, too.

Denote by G an instance of COUNTING DOMINATING SETS, that is, G is a normal graph. For all vertices $v \in V(G)$, we create a set $S_v := N_G[v]$ with the closed neighborhood of v in G . Then, the instance of #SETCOVER consists of the set family $\mathcal{F} := \{S_v \mid v \in V(G)\}$ and the universe $U := V(G)$.

The correctness follows immediately as there is a one-to-one correspondence between the dominating sets of size k and the set covers of size k . $\square$

In order to establish our lower bound for #ANTIFACTOR^{Rel}₁ from Main Lemma 7.8.4, we show a reduction from #SETCOVER to relation-weighted #ANTIFACTOR^{Rel}₁, that is, we prove Lemma 7.8.5.

Lemma 7.8.5. *There is a polynomial-time parsimonious reduction from #SETCOVER with a set family $\mathcal{F}$ of size n , a universe U of size m , and a parameter k to relation-weighted #ANTIFACTOR^{Rel}₁ with 3 different weights such that the size is bounded by $\mathcal{O}(k \cdot n \cdot m)$, the total arity is $\mathcal{O}(1)$, and the pathwidth is at most $k + \mathcal{O}(1)$.*

We postpone the proof of Lemma 7.8.5 and first show how we use this result.

We use two earlier results from Section 7.4 to reduce the relation-weighted instance of #ANTIFACTOR^{Rel}₁ from Lemma 7.8.5 to the unweighted version. Formally, we use Lemmas 7.4.4 and 7.4.5 which we restate here for convenience.

Note that k does *not* denote the number of literals that appear in the clauses of ϕ . This value was only given implicitly.

Lemma 7.4.4. *Let $X \subseteq \mathbb{N}$ be an arbitrary set (which may not be fixed but given as input).*

There is a pathwidth-preserving reduction from edge- and relation-weighted X -#ANTIFACTOR^{Rel} with r different weights to edge-weighted X -#ANTIFACTOR^{Rel} with at most r weights such that

- *the size increases by a multiplicative factor of $\mathcal{O}(r)$, and*
- *the total arity Δ^* increases to $\Delta^* + r$.*

Lemma 7.4.5. *Let $X \subseteq \mathbb{N}$ be a non-empty set (which may not be fixed but given as input).*

There is a pathwidth-preserving reduction from edge-weighted X -#ANTIFACTOR^{Rel} on instances of size n and with r different weights to unweighted X -#ANTIFACTOR^{Rel} operating in time $n^{\mathcal{O}(r)}$. The reduction is such that for each oracle call

- *the size increases by a multiplicative factor of $\mathcal{O}(\log^2(n))$,*
- *the degree of the simple vertices stays the same, and*
- *the total arity increases to at most $\Delta^* + \mathcal{O}(1)$.*

Then, the lower bound from Main Lemma 7.8.4 follows by Lemmas 7.8.2 and 7.8.3 combined with Lemmas 7.4.4 and 7.4.5.

Main Lemma 7.8.4. *There exists a constant c such that there is no $\mathcal{O}(n^{w-c})$ algorithm for #ANTIFACTOR^{Rel}₁ on n -vertex graphs with $\Delta^* \in \mathcal{O}(1)$, even if only one set is used and we are given a path decomposition of width w , unless #SETH fails.*

Moreover, #ANTIFACTOR^{Rel}₁ is #W[2]-hard parameterized by pathwidth.

Proof. Denote by H an instance of #SETCOVER. We apply Lemma 7.8.5 to obtain a relation-weighted #ANTIFACTOR^{Rel}₁ instance with 3 weights. Then, we use Lemma 7.4.4 to transform this into a single instance of edge-weighted #ANTIFACTOR^{Rel}₁ instance with 3 weights. Finally, we apply Lemma 7.4.5 to transform this into a polynomial number of unweighted #ANTIFACTOR^{Rel}₁ instances G_i .

For each instance G_i , we know that $n_{G_i} \leq \mathcal{O}(n \cdot m \cdot k \cdot \log^2(n \cdot m \cdot k))$ and $\text{pw}(G_i) = k + c_1$ for some constant $c_1 > 0$.

Now assume that, for all $c > 0$, there is an algorithm for #ANTIFACTOR^{Rel}₁ on instances of size N and pathwidth pw that operates in time $\mathcal{O}(N^{\text{pw}-c})$.

We apply this algorithm to all instances G_i and recover the solution for the instance H of #SETCOVER by the previous reductions. As the Turing-reduction from Lemma 7.4.5 adds an overhead of $\mathcal{O}((n \cdot m \cdot k)^{c_2})$ to the running time, for some constant $c_2 > 0$, the entire procedure takes time

$$\mathcal{O}\left((n \cdot m \cdot k)^{c_2} + \sum_i n_{G_i}^{\text{pw}(G_i)-c}\right) \leq \mathcal{O}\left((n \cdot m \cdot k)^{c_2} \cdot (n \cdot m \cdot k \cdot \log^2(n \cdot m \cdot k))^{k+c_1-c}\right)$$

and when we set $c_3 := c_1 + c_2$, this is bounded by

$$\mathcal{O}\left((n \cdot m \cdot k \cdot \log^2(n \cdot m \cdot k))^{k+c_3-c}\right) = \mathcal{O}\left((n \cdot m \cdot k)^{k+c_3-c} \cdot \log^{2(k+c_3-c)}(n \cdot m \cdot k)\right).$$

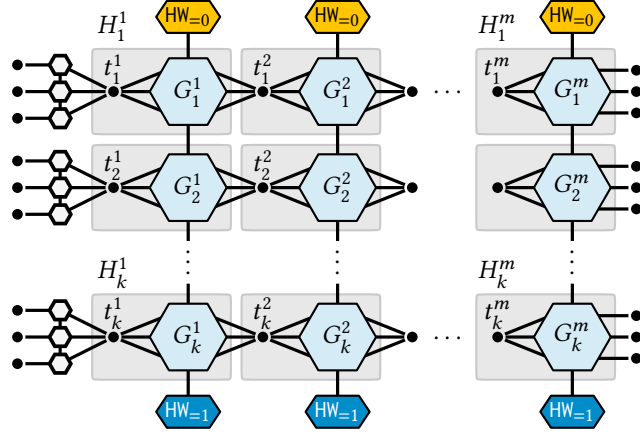


Figure 7.18: The high-level construction of the $\#ANTI-FACTOR^{\text{Rel}}_1$ instance from Lemma 7.8.5. The boxes containing G_i^j and t_i^j describe the gadgets H_i^j . The hexagons on the left are assigned the relation M . The details of the gadgets G_i^j are shown in Figure 7.19.

For large enough N , we know that $\log^a N \leq N^b$. Thus, there is some small constant $c_4 > 0$ to bound the running time by

$$\begin{aligned} \mathcal{O}((n \cdot m \cdot k)^{k+c_3-c} \cdot (n \cdot m \cdot k)^{c_4}) &\leq \mathcal{O}(n^{k+c_5-c} \cdot m^{k+c_5-c} \cdot k^{k+c_5-c}) \\ &\leq \mathcal{O}(n^{k+c_5-c} (\log^d n)^{k+c_5-c} \cdot k^{k+c_5-c}) \end{aligned}$$

where we use that $c_5 := c_3 + c_4$ and $m \in \log^{\mathcal{O}(1)} n$ by assumption.

When we use $\log^d(n) \leq n$ for sufficiently large n and the assumption that k is fixed as it is part of the problem definition, the running time is bounded by

$$\mathcal{O}(n^{k+c_5-c} n) = \mathcal{O}(n^{k+c_5+1-c}).$$

When we chose the constant c such that $c > c_5 + 1$, then the algorithm solves $\#SETCOVER$ in time $\mathcal{O}(n^{k-\varepsilon})$, for some constant $\varepsilon > 0$. However, this contradicts $\#SETH$ by Lemma 7.8.2 and thus, finishes the proof together with the observation that $\#SETCOVER$ is $\#W[2]$ -hard by Lemma 7.8.3. $\square$

In the remainder we prove Lemma 7.8.5.

Idea of the Construction. Our goal is to reduce $\#SETCOVER$ to the variant of $\#ANTI-FACTOR^{\text{Rel}}_1$ where we additionally use weighted relations. From now on, we denote by n the size of the set family $\mathcal{F}$ and by m the size of the universe U . Without loss of generality, we assume that $U = [1..m]$ and that $\mathcal{F} = \{F_1, \dots, F_n\}$.

We start the construction from a $k \times m$ grid. The i th row corresponds to the i th set that we select from $\mathcal{F}$ (note that this is not identical to set F_i) and the j th column to the element $j \in U$. We refer to Figure 7.18 for an illustration of the construction.

At the crossing point of the i th row and the j th column, the gadget H_i^j checks whether the i th “selected” set from $\mathcal{F}$ hits the element j . For the encoding of the sets, we connect H_i^j to the gadget H_i^{j+1} from the next column by $n - 1$ edges such that a selection of ℓ edges encodes the selection of the set $F_{\ell+1}$. Moreover, the gadget H_i^j is connected to the gadget H_{i+1}^j from the next row by a single edge. This edge indicates whether j was hit by one of the first $i - 1$ selected sets.

As a next step we describe the intuitive behavior of the gadget H_i^j . For this we fix some gadget H_i^j . Denote the $n - 1$ edges coming from gadget H_i^{j-1} by $in_1, \dots, in_{n-1}$, denote the $n - 1$ edges going to gadget H_i^{j+1} by $out_1, \dots, out_{n-1}$, denote the (vertical) edge coming from gadget H_{i-1}^j by in_v , and denote the (vertical) edge going to gadget H_{i+1}^j by out_v . We later construct the gadget such that it satisfies the following three constraints.

(P1) The selection of the outgoing edges must be monotone.

Formally, there must be an $\ell \in [1 .. n]$ such that the $n - \ell$ edges $out_1, \dots, out_{n-\ell}$ are not selected but the $\ell - 1$ edges $out_{n-\ell+1}, \dots, out_{n-1}$ are selected.

(P2) The information about the selected element is transferred consistently.

Formally, for all $\ell \in [1 .. n]$, the edge in_ℓ is selected if and only if the edge out_ℓ is selected.

(P3) The gadget H_i^j checks whether the selected set hits the element j and propagates this information to the edge between H_i^j and H_{i+1}^j .

Formally, if the edge in_v is selected, then the edge out_v must also be selected. Otherwise, the edge out_v is selected if and only if there is an $\ell \in [1 .. n]$ such that exactly $\ell - 1$ edges from $in_1, \dots, in_{n-1}$ are selected and j is contained in F_ℓ .

In contrast to the earlier construction, the arity of the gadgets H_i^j depends on the input size n . Hence, we cannot simply define a single relation to satisfy these constraints. Indeed, such a relation would increase the size and pathwidth too much when we replace the relations by suitable gadgets. Instead, we build the gadgets H_i^j from smaller relations which have constant arity. Moreover, we have to do this carefully such that the pathwidth of the construction is not too large, that is, the pathwidth of the entire construction is $k + \mathcal{O}(1)$.

We take care of these details in the formal proof, which we consider next.

Lemma 7.8.5. *There is a polynomial-time parsimonious reduction from #SETCOVER with a set family $\mathcal{F}$ of size n , a universe U of size m , and a parameter k to relation-weighted #ANTIFACTOR $^{Rel}_1$ with 3 different weights such that the size is bounded by $\mathcal{O}(k \cdot n \cdot m)$, the total arity is $\mathcal{O}(1)$, and the pathwidth is at most $k + \mathcal{O}(1)$.*

Proof. For all simple vertices that we introduce in the construction, we choose $X := \{n - 2\}$ as the set of forbidden degrees, that is, the degree $n - 2$ is the only forbidden degree. This is possible as only the size of X is fixed and not the set X itself and thus, is allowed to depend on the input. Hence, we do not explicitly mention this whenever we create a simple vertex.

The graph is formally defined as follows.

- For all $i \in [1..k]$ and all $j \in [1..m]$, we create a vertex t_i^j .
- For all $i \in [1..k]$ and all $j \in [1..m]$, we create a (weighted) gadget G_i^j whose formal description we postpone for now. We connect G_i^j to the two vertices t_i^j and t_i^{j+1} by $n-1$ edges each. Moreover, we connect G_i^j to the two gadgets G_{i-1}^j and G_{i+1}^j by a single edge each.
Intuitively, for all $i \in [1..k]$ and all $j \in [1..m]$, the vertex t_i^j together with the gadget G_i^j constitute the gadget H_i^j from the informal description above.
- Assume we have a monotonicity gadget M which is defined shortly. For all $i \in [1..k]$, we create a copy M_i of this gadget and attach it to t_i^1 by $n-1$ edges.
- For all $i \in [1..k]$, we attach $n-1$ fresh simple vertices to G_i^m .
Similarly, we attach $n-1$ fresh simple vertices to M_i .
- For all $j \in [1..m]$, we create a complex vertex with relation $\text{HW}_{=0}^{(1)}$ and make it adjacent to G_1^j .
- For all $j \in [1..m]$, we create a complex vertex with relation $\text{HW}_{=1}^{(1)}$ and make it adjacent to G_k^j .

It remains to define the gadgets G_i^j and the monotonicity gadget M . We start with the latter as this is used inside G_i^j .

In order to finish the construction, it remains to define the gadgets G_i^j (and the monotonicity gadget M). We decompose each such gadget into four stages and refer to Figure 7.19 for an illustration thereof.

We first describe the purpose of the stages and then define the relations formally and provide the construction of G_i^j afterward. In light of the later construction, we refer to the edges of the relations as left, top, bottom, and right edges and abbreviate them by left, top, bottom, right. If there are more than one edge on one side, we enumerate them from left to right or from top to bottom.

1. Check whether the incoming edges are selected monotonically starting from the top. While doing this, select the ℓ th edge going to the next stage if and only if the first $\ell-1$ edges are selected. The vertical edges encode whether all incoming edges of the blocks above have been selected.

Formally, for all $\ell \in [0..n-1]$, there is a vertex x_ℓ . We connect x_ℓ to t_i^j if $\ell < n-1$ and refer to this edge as the left edge of x_ℓ . For x_{n-1} there is no such left edge, that is, we can assume an edge that is always unselected.

We connect the vertices $x_0, \dots, x_{n-1}$ to form a path from x_0 at the top to x_{n-1} at the bottom where we refer to the top and bottom edges in the natural way. We add a new vertex with relation $\text{HW}_{=1}$ to x_0 by the top edge and add one vertex with relation $\text{HW}_{=0}$ to x_{n-1} by the bottom edge.

We define the relation M as $M := \{(\text{left}, \text{top}, \text{bottom}), (\text{top}, \text{right}), \emptyset\}$ and assign it to all vertices x_ℓ .

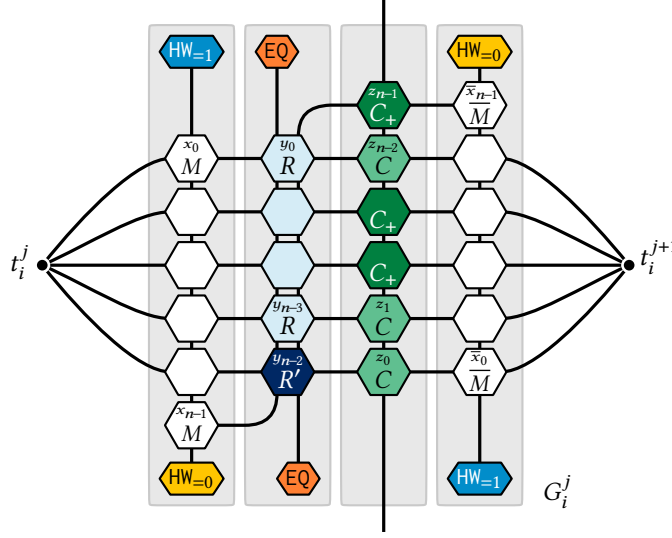


Figure 7.19: The gadget G_i^j from Lemma 7.8.5 checks whether the i th set covers the element j . The gadget ensures together with t_i^j that the information about the selected set is consistent in the whole graph. The vertices of the four different stages are grouped together. The example illustrates the case where F_3 , F_4 , and F_6 cover the element j .

2. This stage assigns the different weights to the solutions depending on the number of selected ingoing and outgoing edges. By the first and last stage (monotonicity gadgets), this information is encoded by the position of the selected edge coming from the first stage and the position of the selected edge going to the third stage. The first and second vertical edge encodes whether $n - 1$ ingoing edges and $n - 1$ outgoing edges are selected, respectively.

Formally, for all $\ell \in [0 \dots n - 2]$, we create a vertex y_ℓ and connect x_ℓ and y_ℓ by one edge to which we refer to as the right edge of x_ℓ and as the left edge of y_ℓ . Additionally, we connect y_{n-2} by the first bottom edge to x_{n-1} , this edge is the right edge of x_{n-1} .

We connect the vertices $y_0, \dots, y_{n-2}$ by two edges to form a path with parallel edges from y_0 at the top to y_{n-2} at the bottom. We add one vertex with relation EQ_1 to y_0 by the first top edge and one vertex with relation EQ_1 to y_{n-2} by the second bottom edge.

We define the relation R as

$$R := \{(\text{left}, \text{right}), (\text{left}, \text{top}_2, \text{bottom}_2), (\text{top}_1, \text{bottom}_1, \text{right}), \\ \emptyset, (\text{top}_1, \text{bottom}_1), (\text{top}_2, \text{bottom}_2), (\text{top}_1, \text{top}_2, \text{bottom}_1, \text{bottom}_2)\}$$

and assign weight -1 to the input $(\text{left}, \text{right})$ and weight 1 to all other accepted inputs. The relation R' completely agrees with R except that we assign weight $-(n - 2)$ to the input $(\text{top}_1, \text{top}_2, \text{bottom}_1, \text{bottom}_2)$.

We assign relation R' to vertex y_{n-2} and relation R to all other vertices y_ℓ .

3. This stage checks whether an element is hit. If the ℓ th edge from the bottom is selected, then this corresponds to set F_ℓ to be selected. We propagate this information to the next stage. If $j \in F_\ell$, then we assign relation C_+ to the ℓ th block from the bottom and otherwise the relation C . The vertical edges are selected if the set has been hit by any element (possibly also by some element from some earlier set $F_{\ell'}$).

Formally, for all $\ell \in [0 \dots n-2]$, we create a vertex z_ℓ , and connect z_ℓ to $y_{n-2-\ell}$ and refer to this edge as the right edge of $y_{n-2-\ell}$ and as the left edge of z_ℓ . Additionally, we create a vertex z_{n-1} and connect it to y_0 and refer to this edge as the second top edge of y_0 and the left edge of z_{n-1} .

We connect the vertices $z_0, \dots, z_{n-1}$ by a single edge each to form a path from z_{n-1} at the top to z_0 at the bottom. We make vertex z_{n-1} incident to the top edge of the gadget G_i^j which goes to gadget G_{i-1}^j . Similarly, we make vertex z_0 incident to the bottom edge of the gadget G_i^j which goes to gadget G_{i+1}^j .

We define the relation C as $C := \{\emptyset, (\text{left}, \text{right}), (\text{top}, \text{bottom}), (\text{left}, \text{top}, \text{bottom}, \text{right})\}$ and $C_+ := C \cup \{(\text{left}, \text{bottom}, \text{right})\}$. If $j \in F_{\ell+1}$, then we assign relation C_+ to vertex z_ℓ . Otherwise, we assign relation C to vertex z_ℓ .

4. This stage is a rotated version of the first stage. That is, if the ℓ th edge from the bottom is selected, then we select $\ell - 1$ outgoing edges. The vertical edges now correspond to the fact whether all outgoing edges from below have been selected.

For all $\ell \in [0 \dots n-1]$, there is a vertex $\bar{x}_\ell$. We connect $\bar{x}_\ell$ to z_ℓ and refer to this edge as the left edge of $\bar{x}_\ell$ and the right edge of z_ℓ . Moreover, we connect $\bar{x}_\ell$ to t_i^{j+1} if $\ell < n-1$, and refer to this edge as the right edge of $\bar{x}_\ell$. For the vertex $\bar{x}_{n-1}$ we can assume that there is a right edge that is always unselected.

We connect the vertices $\bar{x}_0, \dots, \bar{x}_{n-1}$ to form a path from $\bar{x}_{n-1}$ at the top to x_0 at the bottom. We add a new vertex with relation $\text{HW}_{=1}$ to x_0 by the bottom edge and add one vertex with relation $\text{HW}_{=0}$ to x_{n-1} by the top edge.

We define the relation $\bar{M}$ as $\bar{M} := \{(\text{top}, \text{bottom}, \text{right}), (\text{left}, \text{bottom}), \emptyset\}$ and assign it to all vertices $\bar{x}_\ell$.

This concludes the construction. We denote the resulting instance of $\# \text{ANTIFACTOR}_1$ by G . In the following we first show the correctness and then analyze the pathwidth.

Proof of Correctness. We first focus on a fixed H_i^j and thus, gadget G_i^j . By the first and second stage of the gadgets G_i^j , we know that we can describe the signature of H_i^j by a function $\hat{h}(\alpha, \text{in}_v, \gamma, \text{out}_v)$, where α denotes the *number* of selected edges from $\{\text{in}_\ell\}_\ell$ and γ denotes the number of selected edges from $\{\text{out}_\ell\}_\ell$. We focus on the function $h(\alpha, \gamma)$, with the same meaning for α and γ as for $\hat{h}$, where we have

$$\hat{h}(\alpha, \text{in}_v, \gamma, \text{out}_v) = \begin{cases} 0 & \text{if } \text{in}_v = 1 \wedge \text{out}_v = 0, \\ h(\alpha, \gamma) & \text{if } \text{in}_v = \text{out}_v, \\ h(\alpha, \gamma) & \text{if } \text{in}_v = 0 \wedge \text{out}_v = 1 \wedge j \in F_{\gamma+1}. \end{cases}$$

By the definition of H_i^j , we need to show that $h(\alpha, \alpha) = 1$ for all α , and that $h(\alpha, \gamma) = 0$ whenever $\gamma \neq \alpha$.

Now, consider the signature of G_i^j and let β denote the number of selected edges between t_i^j and G_i^j . By the vertices x_ℓ with the imposed monotonicity constraint, we can use the same arguments to denote this signature by $g(\beta, \gamma)$. From the definition of X we get

$$h(\alpha, \gamma) = \sum_{\substack{\beta=0 \\ \alpha+\beta \neq n-2}}^{n-1} g(\beta, \gamma).$$

From the construction of the instance, we know that

- for all $\gamma \neq n-1$, we have $g(n-2-\gamma, \gamma) = -1$,
- for all $\delta \neq n-1$, we have $g(n-1, \delta) = g(\delta, n-1) = 1$,
- we have $g(n-1, n-1) = -(n-2)$, and
- for all other cases, we have $g(\beta, \gamma) = 0$.

It is straightforward to check that these choices satisfy the constraints for h .

We conclude that the gadget G_i^j together with vertex t_i^j satisfy all properties of H_i^j . For this we observe that the fourth stage of G_i^j ensures property (P1) of H_i^j . The third stage ensures the property (P3) and the first and second stage together with t_i^j ensure property (P2) by the choice of the weights as we outlined above.

With these observations it is clear that there is a one-to-one correspondence between the solutions for the #SETCOVER instance and the solutions for the X -#ANTIFACTOR^{Rel} instance. For this we crucially depend on the vertices with relations $\text{HW}_{=0}^{(1)}$ and $\text{HW}_{=1}^{(1)}$. The $\text{HW}_{=0}^{(1)}$ -relations ensure that every element j is initially not hit. Since the gadgets G_i^j propagate the selection of the top edge to the bottom edge whenever the set is not hit by the selected element, there must be some $G_{i'}^j$ such that some selected $F_{\ell_{i'}}$ hits j . Otherwise, the relation $\text{HW}_{=1}^{(1)}$ cannot be satisfied; but this is forbidden.

Claim 7.8.5.1. *The pathwidth of the final graph is $k + \mathcal{O}(1)$.*

Proof of Claim. We use a node search strategy to clean the graph from left to right. For this we place searchers on the vertices t_1^j , for all j , and clean the monotonicity gadgets going from top to bottom.

The remaining graph is cleaned in m rounds. For the j th round we clean the gadgets G_i^j from top to bottom. Before cleaning each G_i^j we place one searcher on t_i^{j+1} and afterward we remove the searcher from t_i^j . Then, we continue with the next row until all G_i^j are cleaned—in this case we proceed to the $j+1$ th round. $\triangleleft$

This concludes the proof of Lemma 7.8.5 $\square$

7.9 Special Case: Counting Edge Covers

In the previous sections we consider the conditional lower bounds for most cases of Λ -FACTOR when the set Λ is finite or cofinite. The notable exception—besides the polynomial-time solvable cases—is the problem #EDGECOVER which is equal to the problem $(\mathbb{N}_{\geq 1})$ -#FACTOR which is identical to $\{0\}$ -#ANTIFACTOR.

In the following we show the conditional lower bound for #EDGECOVER by a self-contained reduction that is not based on any of the previous results. Formally, we prove Theorem 7.9.6.

Theorem 7.9.6 (Lower Bound for Counting Edge Covers). *If there is an $\varepsilon > 0$ and an algorithm that solves #EDGECOVER on a given graph G with a given path decomposition of width w in time $(2 - \varepsilon)^w \cdot |G|^{O(1)}$, then #SETH is false.*

We prove the lower bound in multiple steps. The main step toward this goal is to establish a lower bound for #MAXINDSET.

Definition 7.9.1 (Counting Maximum Independent Sets (#MAXINDSET)). *Given a graph $G = (V, E)$. Let $\mathcal{I}$ denote the set of all independent set of G , that is, we set $\mathcal{I} := \{I \subseteq V \mid \forall \{u, v\} \in E : u \notin I \vee v \notin I\}$.*

The task of #MAXINDSET is to compute the number of maximum independent sets of a graph G . That is, the task is to compute $|\{I \in \mathcal{I} \mid |I| = M\}|$ where $M := \max_{I \in \mathcal{I}} |I|$ denotes the size of the largest independent set.

Now, we can split the lower bound for #EDGECOVER into the following four steps.

1. We first give a self-reduction from the classical problem #SAT to the variant where we add the requirement that the given formula is satisfiable. We formalize this reduction in Lemma 7.9.2.
2. As the first main step, we provide a lower bound for #MAXINDSET on graphs of low degree. To show this, we reduce from #SAT after applying the previous reduction. By the promise that the given SAT-formula is satisfiable, we ensure that there is some independent set of a specified size. We formalize this lower bound in Lemma 7.9.3.
3. As a next step, we reduce #MAXINDSET on graphs with small degree to #MAXINDSET on regular graphs by attaching small gadgets to each vertex such that the graph becomes regular. We formalize this reduction in Lemma 7.9.4.
4. As the last step, we reduce from #MAXINDSET on regular graphs to #EDGECOVER. We use ideas from [15, 17] to prove Lemma 7.9.5.

Finally, we combine the results from Lemmas 7.9.3 to 7.9.5 to prove Theorem 7.9.6.

Step 1: Modifying #SAT. We start with the reduction from #SAT to the mentioned variant of #SAT where the given formula is satisfiable.

Lemma 7.9.2. *Given a k -CNF formula ϕ with n variables and m clauses, we can construct in time linear in the output size a $(k+1)$ -CNF formula ψ on $n+1$ variables and $m+n$ clauses such that $\#SAT(\phi) + 1 = \#SAT(\psi)$.*

Proof. Denote by $x_1, \dots, x_n$ the variables of ϕ and let x_0 be a free variable.

We define ψ such that $\psi \equiv (x_0 \vee \phi) \wedge (\neg x_0 \vee (x_1 \wedge \dots \wedge x_n))$ after we transform the right-hand side into a CNF. Indeed, we can add x_0 to each clause of ϕ and additionally, add, for all $i \in [1..n]$, the clause $(\neg x_0 \vee x_i)$ to the formula to obtain ψ .

By the definition of ψ , the assignment $x_0 = x_1 = \dots = x_n = \text{true}$ is satisfying for ψ . Further, for $x_0 = \text{true}$ this is the only satisfying assignment. For $x_0 = \text{false}$ the satisfying assignments of ϕ directly transfer to ψ which concludes the proof. $\square$

Step 2: #MAXINDSET with Small Degree. Lokshtanov, Marx, and Saurabh [97] prove, under the SETH, a lower bound of the form $(2 - \varepsilon)^{pw} \cdot |G|^{\mathcal{O}(1)}$ for *finding some* maximum independent set in a graph G given a path decomposition of width pw . Unfortunately their reduction is not parsimonious and therefore, does not hold under #SETH. We strengthen this result by proving a lower bound for #MAXINDSET based on the weaker assumption of #SETH. Additionally, we show that this lower bound even holds when the resulting graph has low degree.

Note that Main Theorem III.4 applies for counting independent sets but does not imply a bound on the degree of the vertices and includes all independent sets in the count.

Lemma 7.9.3. *For every constant $\varepsilon > 0$, there is an $r > 0$ such that #MAXINDSET on a graph G with maximum degree r cannot be solved in time $(2 - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, even if G is given with a path decomposition of width w , unless #SETH fails.*

Proof. We provide a reduction from #SAT to #MAXINDSET. Intuitively the graph consists of two parts or rather two types of gadgets. The first type of gadget represents the encoded assignment to the variables and ensures that this assignment is transferred consistently to other parts of the graph. The second type of gadget, the so-called *clause gadgets*, ensure that each clause contains some satisfied literal.

A crucial idea in the construction is to repeat certain parts of these gadgets. By this repetition, we ensure that the size of the maximum independent set decreases significantly when the gadget does not behave as expected, that is, when the gadget does not satisfy the intended purpose.

Construction of the Graph. Let ϕ be a given SAT instance. We first apply Lemma 7.9.2 to ensure that ϕ is satisfiable. For ease of notation, we denote the resulting formula again by ϕ and assume that ϕ contains n variables and m clauses. We construct the graph G_ϕ as follows.

Encoding Assignments. We partition the variables of ϕ into $t := \lceil n/g \rceil$ groups of size at most g , where g is chosen later. We denote the groups by $V_1, \dots, V_t$.

The idea is to encode each assignment to group V_i by a unique subset $S \subseteq [1..d]$ of size $d/2$ where d is chosen later to satisfy $\binom{d}{d/2} \geq 2^g$.

Formally, the graph consists of a grid-like structure with the following vertices and edge. Intuitively, there are $t := \lceil n/g \rceil$ rows, that is, one row for each group of variables, and m columns, that is, one column for each clause. We refer to Figure 7.20 for an illustration of the construction.

- For all $i \in [1..t]$ and all $j \in [1..m]$, we create d new vertices $v_{i,1}^j, \dots, v_{i,d}^j$.
- For all $i \in [1..t]$, we pick an arbitrary injective function $\sigma_i: \{0,1\}^{V_i} \rightarrow \binom{[1..d]}{d/2}$ which associates with each partial assignment for the variables in V_i a subset of $[1..d]$ of size $d/2$.

For different values of i , we might use different mappings.

We say that a set $S \subseteq [1..d]$ with $|S| = d/2$ is *valid* for i if there is some partial assignment $\pi_i: V_i \rightarrow \{0,1\}$ such that $\sigma_i(\pi_i) = S$. Otherwise, the set is invalid.

We say that a valid set S satisfies clause C_j if the assignment $\pi_i: V_i \rightarrow \{0,1\}$ with $\sigma_i(\pi_i) = S$ satisfies the clause C_j . Otherwise, we say that the set S does not satisfy C_j .

- For all $i \in [1..t]$ and all $j \in [1..m]$, we create d cliques on $\binom{d}{d/2}$ vertices and identify each vertex with a set $S \subseteq [1..d]$ where $|S| = d/2$

Formally, for all $\ell \in [1..d]$ and all $S \subseteq [1..d]$ with $|S| = d/2$, we create a vertex $w_{i,S,\ell}^j$. Then, we connect the vertices $w_{i,\star,\ell}^j$ to form a clique with $\binom{d}{d/2}$ vertices.

- For all $i \in [1..t]$, all $j \in [1..m]$, all $\ell \in [1..d]$, and all sets $S \subseteq [1..d]$ with $|S| = d/2$, we connect the vertex $w_{i,S,\ell}^j$ to the vertices $v_{i,k}^j$ and $v_{i,k}^{j+1}$ for all $k \in [1..d] \setminus S$. Note that we use the complement of the set S to define the adjacency.

For $j = m$, the vertices are connected only to the first set of vertices, as the vertices $v_{i,k}^{m+1}$ do not exist.

- For all $i \in [1..t]$ and all $j \in [1..m]$, we say that the vertices $\{v_{i,\ell}^j, w_{i,S,\ell}^j\}_{S,\ell}$ form the gadget D_i^j .

The main observation of this construction is the following. If the vertices $\{v_{i,k}^j \mid k \in S\}$ are selected, where $S \subseteq [1..d]$ with $|S| = d/2$, then we can also select the vertices $\{w_{i,S,\ell}^j \mid \ell \in [1..d]\}$ by the definition of the graph.

Before we continue with the construction of the clause gadgets, we first prove two properties of the previous construction.

Claim 7.9.3.1. *For all i and j , the maximum independent sets of D_i^j have size exactly $d/2 + d$.*

Proof of Claim. We first construct an independent set of the claimed size. Pick an arbitrary set $S \subseteq [1..d]$ with $|S| = d/2$. We select the $d/2$ vertices $v_{i,k}^j$ for all $k \in S$ and select the d vertices $w_{i,S,\ell}^j$ for all $\ell \in [1..d]$.

Now, assume that there is an independent set with more than $d/2 + d$ vertices. Naturally, each of the d cliques can contribute at most one vertex to the independent set. Hence, there must be more than $d/2$ vertices $v_{i,k}^j$ selected, denote the set of indices k by S .

We observe that all sets $S' \subseteq [1..d]$ with $|S'| = d/2$ intersect S . Thus, all vertices of the cliques are adjacent to at least one selected vertex $v_{i,k}^j$. As a consequence this is not an independent set which concludes the proof. $\triangleleft$

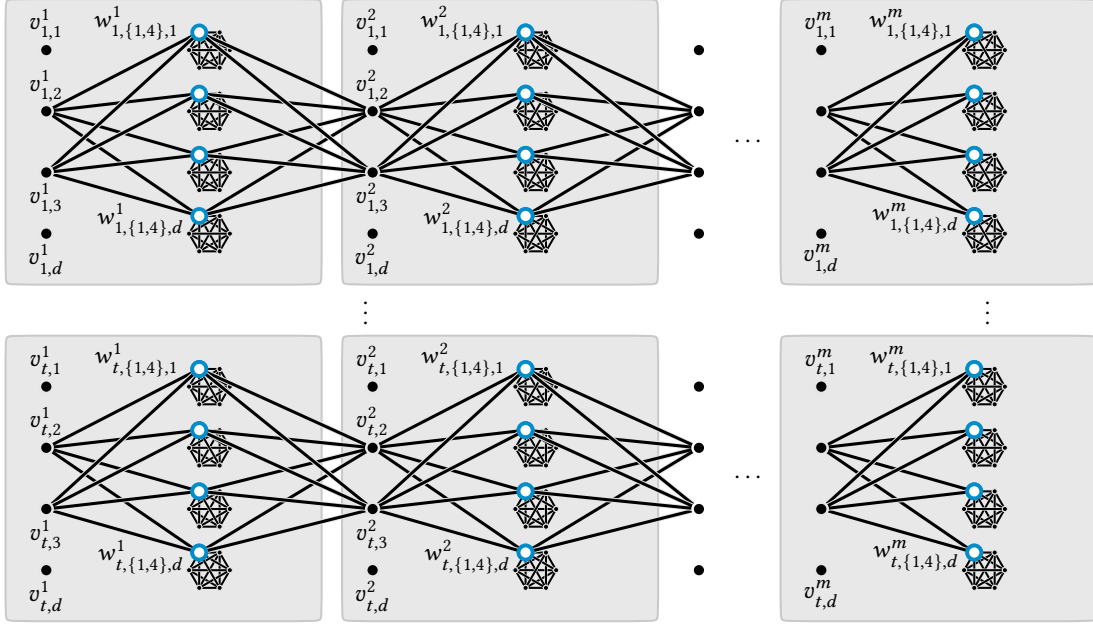


Figure 7.20: The part of the graph from the construction in Lemma 7.9.3 that encodes the assignment. Here shown for $d = 4$.

The highlighted vertex of the clique corresponds to the set $S = \{1, 4\}$.

As a next step, we consider, for a fixed $i \in [1 \dots d]$, all gadgets $D_i^1, \dots, D_i^m$ together.

Claim 7.9.3.2. *Fix an index $i \in [1 \dots t]$. The independent sets in the graph induced by $D_i^1, \dots, D_i^m$ are of size exactly $(d/2 + d) \cdot m$ if and only if there is a set $S \subseteq [1 \dots d]$ with $|S| = d/2$ such that the vertices $v_{i,k}^j$ for all $k \in S$ and the vertices $w_{i,S,\ell}^j$ for all $\ell \in [1 \dots d]$ are selected. Otherwise, the size of the independent set is smaller.*

Proof of Claim. The “if” direction follows directly from the definition of the graph and the size bound for each D_i^j using Claim 7.9.3.1.

Assume for the “only if” direction that we are given a maximum independent set of size $(d/2 + d) \cdot m$. Further, assume there is some $j \in [1 \dots m - 1]$ and there are two sets $S, S' \subseteq [1 \dots d]$ with $|S| = |S'| = d/2$ and $S \neq S'$ such that the vertices $\{v_{i,k}^j \mid k \in S\}$ and the vertices $\{v_{i,k}^{j+1} \mid k \in S'\}$ are selected for this independent set

Thus, there is some index $k \in S' \setminus S$ such that the vertex $w_{i,S,\ell}^j$ is connected to the vertex $v_{i,k}^{j+1}$ by the design of the graph. As we are given an independent set, either none of the vertices $w_{i,S,\ell}^j$ is selected or vertex $v_{i,k}^{j+1}$ is not selected.

In both cases the size of the independent set can be at most $(d/2 + d) \cdot m - 1$ because of the upper bound from Claim 7.9.3.1 for the remaining gadgets. $\triangleleft$

The intuitive meaning of Claim 7.9.3.2 is that in any maximum independent set, for each i , the

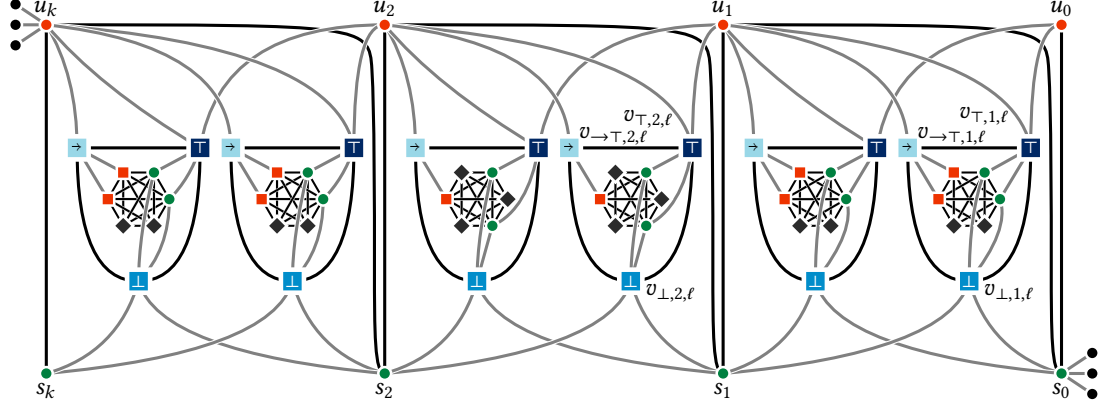


Figure 7.21: The clause gadget from the lower bound for #MAXINDSET with $\kappa = 3$ and $d = 4$ (but only two repetitions are shown).

The diamonds of the clique represent invalid sets, the round vertices represent satisfying assignments, and the square vertices represent assignments that do not satisfy.

The vertices $v_{\rightarrow, i, \ell}$ are labelled with $\rightarrow$, the vertices $v_{\top, i, \ell}$ are labelled with $\top$, and the vertices $v_{\perp, i, \ell}$ are labelled with $\perp$.

Black and gray edges connect vertices of the same and different type, respectively.

encoded assignment is identical for each D_i^j . We say that the partial assignment is *consistently encoded* if the maximum independent set has the size from Claim 7.9.3.2.

As a next step, we define the clause gadgets.

Encoding the Clauses. The clause gadget checks whether a specific clause is satisfied by the encoded partial assignments. We repeat the following construction for each clause C_j of ϕ .

We refer to Figure 7.21 for an illustration of the gadget H_j which we construct next.

First, denote by $i_1, \dots, i_\kappa \in [1 \dots t]$ the indices of the groups of those variables that appear in clause C_j . Then, we proceed as follows.

1. For all $i \in [0 \dots \kappa]$, we create vertices u_i^j and s_i^j and connect them by an edge.

Moreover, for all $i \in [1 \dots \kappa]$, we connect the vertices s_{i-1}^j and u_i^j .

Intuitively, the selection of vertex s_i^j represent that the clause C_j is satisfied by some variable from $V_{i_1}, \dots, V_{i_i}$. In contrast, a selection of the vertex u_i^j represent that the clause C_j is not satisfied by any variable from $V_{i_1}, \dots, V_{i_i}$.

2. For all $i \in [1 \dots \kappa]$, we create the three vertices $v_{\top, i, \ell}^j$, $v_{\perp, i, \ell}^j$, and $v_{\rightarrow, i, \ell}^j$ which are pairwise connected for each $\ell \in [1 \dots d]$, that is, for each ℓ the three vertices form a triangle.

Additionally, we make vertex $v_{\top, i, \ell}^j$ adjacent to u_{i-1}^j and u_i^j . Likewise, we connect $v_{\perp, i, \ell}^j$ to s_{i-1}^j and s_i^j . Additionally, we connect $v_{\rightarrow, i, \ell}^j$ to u_i^j .

For all $\ell \in [1 \dots d]$, and for all $S \subseteq [1 \dots d]$ with $|S| = d/2$ that are not valid for i , we connect the three vertices $v_{\top, i, \ell}^j$, $v_{\perp, i, \ell}^j$, and $v_{\rightarrow, i, \ell}^j$ to vertex $w_{i, S, \ell}^j$.

For all $\ell \in [1 \dots d]$ and for all $S \subseteq [1 \dots d]$ with $|S| = d/2$ that satisfy clause C_j , we connect the vertices $v_{\perp, \ell}^j$ and $v_{\top, \ell}^j$ to vertex $w_{i, S, \ell}^j$.

Conversely, for all $\ell \in [1 \dots d]$ and for all $S \subseteq [1 \dots d]$ with $|S| = d/2$ that do not satisfy clause C_j , we connect the vertex $v_{\rightarrow \top, \ell}^j$ to vertex $w_{i, S, \ell}^j$.

The intuitive idea is as follows. The vertex $v_{\perp, \ell}^j$ is selected if the clause C_j is not yet satisfied and also not satisfied by the assignment for group V_{i_ℓ} . The vertex $v_{\top, \ell}^j$ is selected if the clause C_j is already satisfied but is not satisfied by the assignment for group V_{i_ℓ} . The vertex $v_{\rightarrow \top, \ell}^j$ is selected if the clause is satisfied by the assignment for group V_{i_ℓ} independent of whether the clause is already satisfied.

3. We make the two vertices s_0^j and u_κ^j adjacent to d new vertices, each.

Intuitively, this ensures that in a maximum independent set the vertices s_0^j and u_κ^j are not selected but the vertices u_0^j and s_κ^j should be selected. Hence, the clause is initially not satisfied but eventually must be satisfied.

For the proof of correctness, we only count the vertices that are newly introduced. That is, we do not count the vertices $w_{i, S, \ell}^j$.

Claim 7.9.3.3. *If the assignment is consistently encoded and satisfies the clause, then there is a unique maximal independent set for H_j with $(\kappa + 1) + (\kappa \cdot d) + (2d)$ vertices.*

Proof of Claim. By the construction of the gadget H_j , the independent set must have the following form to obtain size $\kappa + 1 + \kappa \cdot d + 2d$.

- For all $\iota \in [0 \dots \kappa]$, we either select s_ι^j or u_ι^j .
- For all $\iota \in [1 \dots \kappa]$, we select exactly one of the vertex sets $\{v_{\top, \ell}^j\}_\ell$, $\{v_{\perp, \ell}^j\}_\ell$, or $\{v_{\rightarrow \top, \ell}^j\}_\ell$.
- We select the d vertices adjacent to s_0^j and the d vertices adjacent to u_κ^j .

By the construction of the gadget, we directly get that no independent set can be of larger size as otherwise two selected vertices would be adjacent.

As all vertices adjacent to s_0^j are selected, the vertex u_0^j must be selected. We select the other vertices in rounds.

If vertex $u_{\iota-1}^j$ is selected, we cannot select vertex $v_{\top, \ell}^j$ for the independent set. By the above observation and since we want to construct an independent set, we must select either the vertices $v_{\perp, \ell}^j$ for all $\ell \in [1 \dots d]$, or the vertices $v_{\rightarrow \top, \ell}^j$ for all $\ell \in [1 \dots d]$.

Observe that we can only select the vertices $v_{\rightarrow \top, \ell}^j$ if the assignment for group V_{i_ℓ} is satisfying (and thus, also valid) because these vertices are connected to the vertices that represent an assignment that is *not satisfying*. Conversely, the vertices $v_{\perp, \ell}^j$ can only be selected if the assignment for group V_{i_ℓ} is not satisfying (but still valid), as the vertices $v_{\perp, \ell}^j$ are connected to vertices that represent a *satisfying* assignment.

If vertex $s_{\iota-1}^j$ is selected, we know that s_ι^j must be selected as u_ι cannot be selected. Moreover, we cannot select $v_{\perp, \ell}^j$ for the independent set because this vertex is connected to $s_{\iota-1}^j$. Therefore, either all vertices $v_{\perp, \ell}^j$ for $\ell \in [1 \dots d]$ are selected or all vertices $v_{\rightarrow \top, \ell}^j$ for $\ell \in [1 \dots d]$ are

selected. Depending on the assignment which we are given, we can apply the same argument as before.

By this procedure, we eventually arrive at a point, where either u_κ^j or s_κ^j must be selected. Since the clause C_j is satisfied by the encoded assignment, there is some $\bar{i} \in [1 \dots \kappa]$ such that $u_{\bar{i}-1}^j$ and $s_{\bar{i}}^j$ are selected. Thus, by the previous argument, the vertex s_κ^j is selected. $\triangleleft$

Now we prove the converse direction.

Claim 7.9.3.4. *Assume the assignment is consistently encoded. If the maximum independent set of H_j has size $(\kappa + 1) + (\kappa \cdot d) + (2d)$, then the encoded assignment satisfies the clause C_j .*

Proof of Claim. From the definition of the graph, we get that the independent set must have the following form.

- For each $\iota \in [1 \dots \kappa]$, either vertex u_ι^j or vertex s_ι^j is selected.
- For all $\iota \in [1 \dots \kappa]$, exactly one of the vertex sets $\{v_{\top, \iota, \ell}^j\}_\ell$, $\{v_{\perp, \iota, \ell}^j\}_\ell$, or $\{v_{\rightarrow, \iota, \ell}^j\}_\ell$ is selected.
- All d pendant vertices of s_0^j and all d pendant vertices of u_κ^j are selected.

Observe that the second properties especially implies that the encoded assignment is valid.

We first observe that u_0^j and s_κ^j must be selected. Moreover, by the first observation and the fact that the vertices $s_{\bar{i}-1}^j$ and $u_{\bar{i}}^j$ are adjacent for all $\bar{i} \in [1 \dots \kappa]$, there must be some $\bar{i} \in [1 \dots \kappa]$ such that the vertices $u_0^j, \dots, u_{\bar{i}-1}^j$ and the vertices $s_{\bar{i}}^j, \dots, s_\kappa^j$ are selected. This especially implies that, for all $\ell \in [1 \dots d]$, none of the vertices $v_{\perp, \bar{i}, \ell}^j$ and $v_{\top, \bar{i}, \ell}^j$ can be selected as they are connected to $s_{\bar{i}}^j$ and $u_{\bar{i}-1}^j$, respectively.

Therefore, the vertices $v_{\rightarrow, \bar{i}, \ell}^j$ must be selected for all $\ell \in [1 \dots d]$. This is only possible if, for some set $S \subseteq [1 \dots d]$ which corresponds to a satisfying assignment, the vertices $w_{\bar{i}, S, \ell}^j$ are selected. Thus, the assignment is valid and satisfies the clause C_j . $\triangleleft$

Analysis of the Construction. As the last intermediate step, we analyze the pathwidth, size and maximum degree of the graph G_ϕ .

As the values for d and g are not set yet, we use them for the bounds.

Claim 7.9.3.5. *The size of G_ϕ is $\mathcal{O}(m \cdot n \cdot \binom{d}{d/2} \cdot d)$.*

The maximum degree of G_ϕ is $\binom{d}{d/2} + 2d + \mathcal{O}(1)$.

The graph G_ϕ has pathwidth at most $\lceil n/g \rceil \cdot d + \binom{d}{d/2} + d + \mathcal{O}(1)$.

Proof of Claim. By the construction of G_ϕ , we get a size bound of

$$m \cdot \left\lceil \frac{n}{g} \right\rceil \cdot \left(\binom{d}{d/2} \cdot d + d \right) + m \cdot k \cdot \mathcal{O}(d)$$

assuming the given formula is in k -CNF. The claimed bound then follows by a straightforward simplification.

To bound the degree, we observe that the vertices $v_{i,1}^j, \dots, v_{i,d}^j$ have degree at most $2 \cdot \binom{d-1}{d/2} \leq \binom{d}{d/2}$. The vertices $w_{i,S,\ell}^j$ have degree at most $\binom{d}{d/2} + 2 \cdot d/2 + \mathcal{O}(1)$. The vertices in the clause gadget have degree at most $d + \mathcal{O}(1)$. Hence, the maximum degree of a vertex is $\binom{d}{d/2} + d + \mathcal{O}(1)$.

To bound the pathwidth of the construction, we provide a node-search strategy. For this reason, we iterate over all clauses and clean the clause gadget and the corresponding columns starting from the first row (which corresponds to the first group V_1 of variables). We place d searchers on the vertices $v_{i,1}^{j+1}, \dots, v_{i,d}^{j+1}$ of the next gadget D_i^{j+1} and then clean the cliques and clause gadget sequentially by cleaning repetition by repetition. As the cliques are not connected to each other, this requires only $\binom{d}{d/2} + \mathcal{O}(1)$ searchers. Moreover, the cliques are only connected to vertices that already have a searcher on them, that is, the vertices $v_{i,1}^j, \dots, v_{i,d}^j$ of the gadget D_i^j and the vertices $v_{\tau,i,\ell}^j, v_{\perp,i,\ell}^j$, and $v_{\rightarrow\tau,i,\ell}^j$ of the clause gadget H_j for a fixed ℓ , as we place searcher simultaneously on them. Thus, we can clean the clause gadgets with $\mathcal{O}(1)$ searchers. $\triangleleft$

Now we have everything ready to complete the proof.

Lower Bound. Assume, for some $\varepsilon > 0$, a given algorithm solves #MAXINDSET for an instance G with maximum degree $r = \binom{d}{d/2} + d + \mathcal{O}(1)$ that is given with a path decomposition of width pw_G in time $(2 - \varepsilon)^{\text{pw}_G} \cdot |G|^{\mathcal{O}(1)}$. The precise value for r is given by Claim 7.9.3.5 and only depends on ε as we show in the following.

We use this algorithm for #MAXINDSET to count the number of satisfying assignments of a SAT formula ϕ with n variables in time $(2 - \delta)^n \cdot |\phi|^{\mathcal{O}(1)}$ for some $\delta > 0$.

For ease of notation, we define $\varepsilon' := \log(2 - \varepsilon)$. We choose some $\alpha > 1$ such that $\alpha \cdot \varepsilon' = \delta' := \log(2 - \delta) < 1$ for some $\delta > 0$.

We set d to be an even integer that is large enough such that $d \leq \alpha \cdot \lfloor d - \log d \rfloor$. Then, we set $g := \lfloor d - \log d \rfloor$ and claim that $2^g \leq \binom{d}{d/2}$, that is, we can actually encode all assignments. Indeed, by the properties of the binomial coefficients, we get that $\binom{d}{d/2} \geq 2^d/d$ from which the claim follows.

As the next step, we construct the graph G_ϕ from the given formula ϕ by the procedure above. We use the algorithm for #MAXINDSET to count the number of maximum independent sets. Since we know that ϕ is satisfiable, the number of maximum independent sets corresponds to the number of satisfying assignments of ϕ .

The entire procedure operates in time

$$\begin{aligned} (2 - \varepsilon)^{\text{pw}_{G_\phi}} \cdot |G_\phi|^{\mathcal{O}(1)} &\leq (2 - \varepsilon)^{\lceil \frac{n}{g} \rceil \cdot d + \binom{d}{d/2} + d + \mathcal{O}(1)} \cdot |G_\phi|^{\mathcal{O}(1)} \\ &\leq (2 - \varepsilon)^{\frac{n}{g} \cdot d + \binom{d}{d/2} + 2d + \mathcal{O}(1)} \cdot |\phi|^{\mathcal{O}(1)}. \end{aligned}$$

Observe that d does not depend on the input formula but just on ε . Thus, the term $(2 - \varepsilon)^{\binom{d}{d/2} + 2d + \mathcal{O}(1)}$ contributes only a constant factor to the overall running time. Thus, the running time is bounded by

$$(2 - \varepsilon)^{\frac{n}{g} \cdot d} \cdot |\phi|^{\mathcal{O}(1)} \leq 2^{\varepsilon' \cdot \frac{n}{\lfloor d - \log d \rfloor} \cdot d} \cdot |\phi|^{\mathcal{O}(1)}.$$

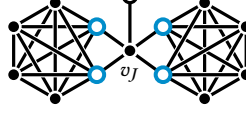


Figure 7.22: The gadget J for $\bar{r} = 5$ from Claim 7.9.4.1 which we attach to the existing vertices by the dangling edge such that every vertex has degree $\bar{r}$. The four marked vertices form the unique maximum independent set.

By our choice of d and α , this is bounded by

$$2^{\epsilon' \cdot \alpha \cdot n} \cdot |\phi|^{\mathcal{O}(1)} \leq 2^{\delta' \cdot n} \cdot |\phi|^{\mathcal{O}(1)} \leq (2 - \delta)^n \cdot |\phi|^{\mathcal{O}(1)}.$$

As this running time now immediately contradicts #SETH, this concludes the proof. $\square$

Step 3: #MAXINDSET on Regular Graphs. For the next step we provide the reduction from #MAXINDSET on instances with low degree to #MAXINDSET on regular graphs (of low degree).

The basic idea is to add small gadgets to each vertex that does not have full degree yet. This gadget is defined such that it does not alter any maximum independent set in the original graph besides increasing the size.

Lemma 7.9.4. *For all $r \geq 4$, there is a parsimonious many-one reduction from #MAXINDSET on graphs with degree at most r to #MAXINDSET on $r + (1 + r \bmod 2)$ -regular graphs such that pathwidth increases by at most r .*

Proof. Assume we are given an instance G of #MAXINDSET with degree at most r . We modify G such that we get a $\bar{r}$ -regular graph H where $\bar{r}$ is the odd number of r and $r + 1$, that is, we set $\bar{r} := r + (1 + r \bmod 2)$.

We first construct a gadget J which we later add to all vertices with degree smaller than $\bar{r}$. See Figure 7.22 for an illustration of this gadget.

Claim 7.9.4.1. *There is a gadget J of size at most $\bar{r}^2$ and pathwidth $r + 2$ such that all vertices except for one distinguished vertex v_J have degree $\bar{r}$ and vertex v_J has degree $\bar{r} - 1$. Moreover, there is a unique maximum independent set of size $\bar{r} - 1$ and vertex v_J is not part of that independent set.*

Proof of Claim. We first create the distinguished vertex v_J . Then, we create a clique with $\bar{r} + 1$ vertices and remove an edge between two arbitrary vertices, say between v_1 and v_2 . We add an edge from v_1 to v_J and an edge from v_2 to v_J .

We repeat this process such that we obtain $(\bar{r} - 1)/2$ modified cliques in total. This completes the construction of the gadget J .

Observe that vertex v_J has degree $\bar{r} - 1$ whereas all other vertices in the gadget have degree $\bar{r}$.

Moreover, the gadget J has a unique maximum independent set of size $r - 1$ for which we select all copies of the vertices v_1 and v_2 from each of the $(\bar{r} - 1)/2$ modified cliques. Especially, vertex v_J is not part of this independent set. Indeed, since $\bar{r} > 4$, we know that $1 + (\bar{r} - 1)/2 < \bar{r} - 1$ and thus, selecting vertex v_J results in a smaller independent set.

From the construction of the gadget J , we know that there are at most $1 + (\bar{r} - 1)/2 \cdot (\bar{r} + 1)$ vertices. Hence, the size is bounded by $\bar{r}^2$. $\triangleleft$

Now, for each vertex v in G , we create $r - \deg(v)$ copies of the gadget J from Claim 7.9.4.1 and make all copies of v_J adjacent to v .

Let H denote the resulting r -regular graph. Observe that H retains all the maximum independent sets from G as the portal vertex v_J is not selected in any maximal independent set of the gadget J .

The number of vertices of the graph H is bounded by $\mathcal{O}(n \cdot r^3)$ and the pathwidth increases by an additive term of $r + 2$. $\square$

Step 4: From #MAXINDSET to #EDGECOVER. For the last step, we use Fact 2.1.7 which allows us to recover multivariate polynomials. For convenience, we restate the result here.

Fact 2.1.7 ([33, Lemma 2.5]). *Let $P \in \mathbb{Q}[x_1, \dots, x_q]$ be a multivariate polynomial such that, for all $i \in [1..q]$, the degree of x_i in P is bounded by $d_i \in \mathbb{N}$. Furthermore, assume we are given sets $\Xi_i \subseteq \mathbb{Q}$ for $i \in [1..q]$ such that $|\Xi_i| = d_i + 1$ for all $i \in [1..q]$. Consider the Cartesian product of these sets, that is,*

$$\Xi := \Xi_1 \times \dots \times \Xi_q.$$

Then, we can compute the coefficients of P with $\mathcal{O}(|\Xi|^3)$ arithmetic operations when given as input the set $\{(\xi, P(\xi)) \mid \xi \in \Xi\}$.

Now we have everything ready for the last reduction where we reduce from #MAXINDSET on regular graphs to #EDGECOVER.

Lemma 7.9.5. *For all $r > 0$, there is a pathwidth-preserving reduction from #MAXINDSET on r -regular graphs to #EDGECOVER.*

Proof. For some arbitrary value r , denote by $H = (V, E)$ an r -regular instance of #MAXINDSET. We follow the ideas from the #P-completeness proof of #EDGECOVER by Bordewich, Dyer, and Karpinski [15] to obtain a Turing-reduction from #MAXINDSET to #EDGECOVER. For ease of notation we write $n = |V|$ in the following.

For an integer $j \in [0..n]$, we denote by $I_j(H)$ the number of independent sets of size exactly j in H .

Based on the graph H , we define a new graph G where we subdivide all edges of H by placing a new vertex on each edge. We denote by U the set of these new vertices in G .

For an integer $i \in [0..n]$, we denote by $N_i(G)$ be the number of edge subsets $E' \subseteq E(G)$ such that i vertices of V are not covered but all vertices of U are covered. We follow a similar argument as in [15, 17] to count the number of subgraphs that contribute to $N_i(G)$.

Claim 7.9.5.1. *For all $j \in [0..|V|]$, it holds that*

$$3^{r \cdot n/2 - j \cdot r} \cdot I_j(H) = \sum_{i=j}^n \binom{i}{j} \cdot N_i(G).$$

Proof of Claim. We prove the claim by analyzing how independent sets for H can be transformed into edge covers of G . The first formulation covers with the left-hand side and the second corresponds to the right-hand side.

Fix an independent set $S \subseteq V(H)$ of size j in H and consider some vertex $v \in S$. Denote by $(v, v_1), \dots, (v, v_r)$ the incident edges of v . Suppose these edges are subdivided to get the edges $(v, u_1), (u_1, v_1), \dots, (v, u_r), (u_r, v_r)$ in G for some $u_1, \dots, u_r \in U$. We select none of the edges from $(v, u_1), \dots, (v, u_r)$ and select all edges $(u_1, v_1), \dots, (u_r, v_r)$ in the solution for G . We repeat the previous selection procedure for every $v \in S$.

Now, observe that there are exactly $r \cdot n/2 - j \cdot r$ edges in H that are not incident to a vertex from S . Indeed, there are $r \cdot n/2$ edges in total, and the vertices in the independent set S are incident to $j \cdot r$ of them. For each of these edges, there is a unique vertex $u \in U$ subdividing this edge in G . We select at least one of the two subdividing edges incident to u ; there are three possible ways to do this. The number of possible ways for this selection forms the left-hand side of the equation.

For the right-hand side, observe that the above process creates every subset of edges in G where S is not covered and all of U is covered. If an edge set $E' \subseteq E(G)$ leaves exactly i vertices, say $S' \subseteq V$, uncovered in G , then the above process generates E' only if $S \subseteq S'$. Thus, there are exactly $\binom{i}{j}$ sets S of size j for which the process creates E' . $\triangleleft$

By Claim 7.9.5.1, we can recover the values of $I_j(H)$ once we know the values of $N_i(G)$. Thus, it remains to compute the values of $N_i(G)$.

For all $i \in [1..n]$ and all $j \in [1..|U|]$, we denote by $N_{ij}(G)$ be the number of edge subsets of G that leave exactly i vertices in V and exactly j vertices in U uncovered.

For two positive integers k, ℓ , we construct a new graph $G_{k,\ell}$ based on the graph G as follows. For all vertices $v \in V$, we attach a path with k edges to vertex v , and, for all vertices $u \in U$, we attach a path with ℓ edges to vertex u .

For a path with ℓ edges, we denote by M_ℓ the number of edge covers of such a path. Observe that $M_\ell := M_{\ell-1} + M_{\ell-2}$ with $M_1 := M_2 := 1$, which is precisely the definition of the Fibonacci numbers.

Now, consider an edge subset E' of G that leaves exactly i vertices in V and exactly j vertices in U uncovered. That is, the set E' contributes toward $N_{ij}(G)$. We analyze how E' can be extended to an edge cover of $G_{k,\ell}$. For a vertex $v \in V \cup U$ that is already covered by some edge in E' , we can choose an edge cover of the path attached to v that either covers v or does not cover v . For a vertex $v \in V \cup U$ that is *not* covered by any edge in E' , we must choose an edge cover of the path attached to v that also covers v .

Denote by $\#EC(G_{k,\ell})$ the number of edge covers of $G_{k,\ell}$. Then, the above observations yields

$$\begin{aligned} \#EC(G_{k,\ell}) &= \sum_{i=0}^n \sum_{j=0}^{r \cdot n/2} (M_k)^i \cdot (M_k + M_{k-1})^{n-i} \cdot (M_\ell)^j \cdot (M_\ell + M_{\ell-1})^{r \cdot n/2 - j} \cdot N_{ij}(G) \\ &= (M_k)^n \cdot (M_\ell)^{r \cdot n/2} \cdot \sum_{i=0}^n \left(1 + \frac{M_{k-1}}{M_k}\right)^{n-i} \cdot \sum_{j=0}^{r \cdot n/2} \left(1 + \frac{M_{\ell-1}}{M_\ell}\right)^{r \cdot n/2 - j} \cdot N_{ij}(G). \end{aligned}$$

We interpret $\#EC(G_{k,\ell})$ as a polynomial in the variables x_1 and x_2 that is evaluated at $x_1 = 1 + M_{k-1}/M_k$ and $x_2 = 1 + M_{\ell-1}/M_\ell$. The degree of x_1 is at most n and the degree of x_2 at most $r \cdot n/2$.

Now the goal is to evaluate the unknown polynomial at sufficiently many points to recover the coefficients and thus, the values of $N_{i,j}(G)$.

To evaluate the polynomial, we use the oracle-access to $\#EDGECOVER$ for varying choices of k and ℓ . Since the values of M_ℓ correspond to Fibonacci numbers, we know that $M_\ell/M_{\ell-1}$ takes infinitely many distinct values, although the sequence is converging.

Hence, we can pick $n+1$ values for k and $r \cdot n/2 + 1$ values for ℓ and compute $\#EC(G_{k,\ell})$ for these values. Then, we can use Fact 2.1.7 to recover the polynomial and especially the values of $N_{i,j}(G)$ for all values of $i \in [0..n]$ and $j \in [0..r \cdot n/2]$.

By the definition of $N_i(G)$ and $N_{i,j}(G)$, we immediately get that $N_i(G) = N_{i,0}(G)$. Hence, we can recover the number of maximum independent sets of H , that is, we return $I_{j'}(H)$ where $j' := \max\{j \in [0..r \cdot n/2] \mid I_j(H) \neq 0\}$.

It remains to analyze the pathwidth of the new instance. Observe that we only subdivided edges and attached graphs of constant pathwidth to vertices. As both of these steps do not increase pathwidth by more than an additive constant, this concludes the proof. $\square$

Finalizing the Proof. Now we can combine the previous results from Lemmas 7.9.3 to 7.9.5 to prove the lower bound for $\#EDGECOVER$, that is, prove Theorem 7.9.6.

Theorem 7.9.6 (Lower Bound for Counting Edge Covers). *If there is an $\varepsilon > 0$ and an algorithm that solves $\#EDGECOVER$ on a given graph G with a given path decomposition of width w in time $(2 - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then $\#SETH$ is false.*

Proof. Assume for some $\varepsilon > 0$, that there is an algorithm that solves $\#EDGECOVER$ in time $(2 - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$ for an instance G that is given with a path decomposition of width w .

Let r denote the value from Lemma 7.9.3 such that $\#MAXINDSET$ cannot be solved in time $(2 - \varepsilon)^{w_H} \cdot |H|^{\mathcal{O}(1)}$ for an instance H of degree at most r that is given along with a path decomposition of width w_H .

Now, consider an instance H of $\#MAXINDSET$ with maximum degree r . We apply the many-one reduction from Lemma 7.9.4 to H . We apply the Turing-reduction from Lemma 7.9.5 to the resulting instances to obtain a polynomial number of instances of $\#EDGECOVER$. We denote these instances by G_i .

Now, we apply the faster algorithm for $\#EDGECOVER$ to each of these instances G_i and recover the solution for the original instance H of $\#MAXINDSET$.

We know that the reduction changes the pathwidth of the instances only by a small constant, that is, we get $\text{pw}(G_i) \leq w_H + \mathcal{O}(1)$. Similarly, the size of the new instances is polynomial in the size of the instance H .

We conclude that the entire procedure operates in time

$$|H|^{\mathcal{O}(1)} + \sum_i (2 - \varepsilon)^{\text{pw}(G_i)} \cdot |G_i|^{\mathcal{O}(1)} = (2 - \varepsilon)^{w_H} \cdot |H|^{\mathcal{O}(1)}.$$

However, by Lemma 7.9.3, this is a contradiction to $\#SETH$, which concludes the proof. $\square$

Part III

General Domination

Introduction to General Domination

8

In this part of the thesis we consider the generalization of vertex selection problems such as DOMINATING SET or INDEPENDENT SET. Formally, for two sets σ and ρ of non-negative integers, we define the problem (σ, ρ) -DOMSET as follows.

Definition 8.1.1 ((σ, ρ) -DOMINATING SET ((σ, ρ) -DOMSET)). *Fix two sets σ and ρ of non-negative integers, that is, $\sigma, \rho \subseteq \mathbb{N}$.*

For a graph G , we say that a set of vertices $S \subseteq V(G)$ is a (σ, ρ) -set (or a solution)

1. *if for every $u \in S$, we have $|N(u) \cap S| \in \sigma$, and*
2. *if for every $v \in V(G) \setminus S$, we have $|N(v) \cap S| \in \rho$.*

We refer to the first requirement as the σ -constraint and to the second as the ρ -constraint.

The decision problem (σ, ρ) -DOMSET takes as input a graph G , and asks whether G has a (σ, ρ) -set $S \subseteq V(G)$.

Similarly, we can define the counting variant (σ, ρ) -#DOMSET where, for a given graph G , the task is to determine the *number* of (σ, ρ) -sets $S \subseteq V(G)$.

As DOMINATING SET corresponds to (σ, ρ) -DOMSET where $\sigma = \mathbb{N}$ and $\rho = \mathbb{N}_{\geq 1}$ there is an algorithm for this case parameterized by treewidth by van Rooij, Bodlaender, and Rossmanith [132]. Similarly, Independent Set corresponds to (σ, ρ) -DOMSET where $\sigma = \{0\}$ and $\rho = \mathbb{N}$ and there is a straightforward algorithm for this case (see [36, Section 7.3.1] for example).

Besides these individual results, van Rooij provides algorithms parameterized by treewidth for the broad class of finite or cofinite sets [130, 131]. Naturally, the running time of these algorithms should depend on σ and ρ and, more precisely, on the largest reasonable element in σ and ρ .

Definition 2.1.3 (The Top Element). *Consider a set $\tau \subseteq \mathbb{N}$ that is either finite or cofinite. Based on the set τ , we define the number $\text{top } \tau \in \tau$ as*

$$\text{top } \tau := \begin{cases} \max \tau & \text{if } \tau \text{ is finite,} \\ 1 + \max(\mathbb{Z} \setminus \tau) & \text{if } \tau \text{ is cofinite.} \end{cases}$$

To shorten notation, we associate the following parameters with (σ, ρ) . We define

$$s_{\text{top}} := \text{top } \sigma \quad \text{and} \quad r_{\text{top}} := \text{top } \rho \quad \text{and} \quad t_{\text{top}} := \max\{s_{\text{top}}, r_{\text{top}}\}. \quad (8.1.1)$$

Then, we can state the general algorithm by van Rooij as follows.

Theorem 8.1.2 ([131, Theorem 1]). *Fix two sets $\sigma, \rho \subseteq \mathbb{N}$ that are finite or cofinite. Given a graph G with a tree decomposition of width w and an integer s , we can count the number of (σ, ρ) -sets of size exactly s in time $(s_{\text{top}} + r_{\text{top}} + 2)^w \cdot n^{\mathcal{O}(1)}$.*

It is a natural question to ask whether this algorithm is optimal. In light of the matching bounds from Theorem 5.1.3 for Λ -FACTOR, the first intuitive answer might be “yes, this algorithm is optimal”. However, this answer is actually far from the truth. Although the algorithm from Theorem 8.1.2 provides a good running time, there is an infinite class of pairs (σ, ρ) where significant improvement are possible. To characterize these sets, we introduce some notation.

Definition 8.1.3 (m-Structured Sets). *Fix an integer $m \geq 1$. We say that a set $\tau \subseteq \mathbb{N}$ is m-structured if there is some number $t^* \in \mathbb{N}$ such that*

$$t \equiv_m t^*$$

for all $t \in \tau$.

We say that a pair (σ, ρ) is m-structured if both σ and ρ are m-structured. Observe that (σ, ρ) is always 1-structured. Hence, we are usually interested in the largest value m such that (σ, ρ) is still m-structured.

Our Results. With this definition, we can now formally define the coefficients $c_{\sigma, \rho}$ which describe the running time of our improved algorithm.

Definition 8.1.4. *Consider two finite or cofinite sets σ and ρ of non-negative integers. We define $c_{\sigma, \rho}$ by setting*

- $c_{\sigma, \rho} := s_{\text{top}} + r_{\text{top}} + 2$ if (σ, ρ) is not m-structured for any $m \geq 2$,
- $c_{\sigma, \rho} := \max\{s_{\text{top}}, r_{\text{top}}\} + 2$ if (σ, ρ) is 2-structured, but not m-structured for any $m \geq 3$, and $s_{\text{top}} = r_{\text{top}}$ is even, and
- $c_{\sigma, \rho} := \max\{s_{\text{top}}, r_{\text{top}}\} + 1$, otherwise.

With this definition, we can formally state our first main result for (σ, ρ) -DOMSET which provides a faster algorithm for (σ, ρ) -DOMSET when (σ, ρ) are structured.

Main Theorem III.1. *Fix two finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$. Given a graph G with a tree decomposition of width w and an integer s , we can count the number of (σ, ρ) -sets of size exactly s in time $(c_{\sigma, \rho})^w \cdot |G|^{\mathcal{O}(1)}$.*

We remark that this algorithm solves the decision and the counting version simultaneously.

In contrast to the algorithm by van Rooij, we prove that our improved algorithm from Main Theorem III.1 is essentially optimal (for the counting version).

Main Theorem III.4. *Fix two non-empty finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$ with $\rho \neq \{0\}$ and $(\sigma, \rho) \neq (\mathbb{N}, \mathbb{N})$.*

If there is an $\varepsilon > 0$ and an algorithm that counts the number of (σ, ρ) -sets in a given graph G with a given path decomposition of width w in time $(c_{\sigma, \rho} - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-#SETH fails.

Note that the lower bound is stated for the larger parameter pathwidth and thus, immediately implies the corresponding result for treewidth. Hence, the lower bound matches the upper bound except for the two explicitly excluded cases. We briefly argue that this restriction is actually needed as otherwise the problem is easy to solve.

Theorem 8.1.5. *Fix two sets $\sigma, \rho \subseteq \mathbb{N}$ such that $\rho = \{0\}$ or $(\sigma, \rho) = (\mathbb{N}, \mathbb{N})$.*

Then, (σ, ρ) -#DOMSET can be solved in polynomial time.

Proof. If $\rho = \{0\}$, then every connected component of a given graph can be unselected. Moreover, each such component can also be selected if the degree of every vertex is contained in σ . This property can be checked easily as σ is given. Hence, denote by c the number of connected components of the given graph that satisfy this property. Then, the number of (σ, ρ) -sets is 2^c .

If $(\sigma, \rho) = (\mathbb{N}, \mathbb{N})$, then the situation is even simpler. Indeed, the vertices do not impose any constraint and every subset of the vertices is a solution. That is, the number of (σ, ρ) -sets is $2^{|V(G)|}$ for a given graph G . $\square$

Based on Theorem 8.1.5, we say that (σ, ρ) is *trivial* or *easy* if $\rho = \{0\}$ or $(\sigma, \rho) = (\mathbb{N}, \mathbb{N})$. Otherwise, we say that the pair is non-trivial.

Decision Version. While the lower bound from Main Theorem III.4 focuses on the counting version, we also prove a similar result for the decision version. However, here the complexity is less understood which forces us to restrict to finite sets.

Main Theorem III.3. *Fix two finite sets $\sigma, \rho \subseteq \mathbb{N}$ with $0 \notin \rho$. If there is an $\varepsilon > 0$ and an algorithm that decides whether there is a (σ, ρ) -set in a given graph G with a given path decomposition of width w in time $(c_{\sigma, \rho} - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-SETH fails.*

We again observe that we strengthen the lower bound by stating it for the larger parameter pathwidth as this immediately implies the corresponding bound for treewidth.

Moreover, we again remark that we need to exclude the case when $0 \in \rho$ as otherwise the problem is trivial to solve.

Theorem 8.1.6. *Fix two sets $\sigma, \rho \subseteq \mathbb{N}$ with $0 \in \rho$.*

Then, (σ, ρ) -DOMSET can be solved in constant time.

Proof. Observe that if $0 \in \rho$, then the empty set is a trivial solution and (σ, ρ) -DOMSET becomes trivial independently of the precise definition of σ and ρ . $\square$

The restriction of the lower bounds from Main Theorem III.3 leaves open the question of faster algorithms when σ or ρ are cofinite. We (partially) answer this question with an improved algorithm where the running time does not depend on s_{top} and r_{top} but the number of forbidden degrees. We formalize this idea by the *cost* of a set.

Definition 8.1.7 (Cost of a Set). *Consider a set $\tau \subseteq \mathbb{N}$ that is either finite or cofinite. Based on the set τ , we define the number $\text{cost } \tau$ as*

$$\text{cost } \tau := \begin{cases} \max \tau & \text{if } \tau \text{ is finite,} \\ |\mathbb{N} \setminus \tau| & \text{if } \tau \text{ is cofinite.} \end{cases}$$

Our improved algorithm uses fast matrix multiplication as a subroutine. Therefore, we recall that $\omega \leq 2.371552$ denotes the matrix multiplication exponent [135].

Main Theorem III.2. Fix two finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$. Based on these sets we define $t_{\text{cost}} := \max\{\text{cost } \sigma, \text{cost } \rho\}$.

Given a graph G with a tree decomposition of width w , and an integer s , we can decide whether G has a (σ, ρ) -set of size exactly s in time

$$2^w \cdot (t_{\text{cost}} + 1)^{w \cdot (\omega + 1)} \cdot |G|^{\mathcal{O}(1)}.$$

By the algorithm from Main Theorem III.2, it is clear that we need to restrict the lower bound from Main Theorem III.3 to finite sets. Otherwise, the combination of the two results would contradict pw-SETH.

Such algorithms are
frequently called
combinatorial.

As the algorithm in Main Theorem III.2 depends on matrix multiplication, this leaves open the quest for faster algorithms (that do not use matrix multiplication). However, while a first approach could improve the running time to $(t_{\text{cost}} + 1)^{tw}$ it is unclear whether this running time is optimal.

At this point we are at a similar situation as for the algorithm by van Rooij before our results; although the algorithm seemed optimal, non-trivial techniques provide a significantly better bound in many cases. For the case of cofinite sets it is quite likely that, for the decision version, similar improvements are possible for the algorithm in Main Theorem III.2.

Organization. Before we give the formal proofs, we provide some useful definitions and concepts that we use for the upper and lower bounds in Section 8.1. Then, we prove the algorithmic results from Main Theorems III.1 and III.2 in Chapter 9, and we prove the lower bounds from Main Theorems III.3 and III.4 in Chapter 10.

8.1 Preliminaries

In the following, we denote by $\sigma, \rho \subseteq \mathbb{N}$ two sets that are finite or cofinite.

Analogously to the definition of s_{top} and r_{top} , we also define $s_{\text{min}} := \min \sigma$ and $r_{\text{min}} := \min \rho$.

Partial Solutions and States. For our algorithmic results, a key ingredient is the description of *partial solutions*. We formalize this based on graphs with *portals*.

A *graph with portals* is a pair (G, U) , where G is a graph and $U \subseteq V(G)$. If $U = \{u_1, \dots, u_k\}$, then we also write $(G, u_1, \dots, u_k)$ instead of (G, U) . Intuitively speaking, the idea of this notion is that G may be part of some larger graph that interacts with G only via vertices from U .

In particular, in the context of the (σ, ρ) -DOMSET problem, vertices in U do not necessarily need to satisfy the σ - and ρ -constraints imposed by a (σ, ρ) -set since they may receive further selected neighbors from outside G . That is, we only have a *partial* solution.

Definition 8.1.8 (Partial Solution). Fix a graph with portals (G, U) . A set $S \subseteq V(G)$ is a partial solution (with respect to U) if

(PS1) for each $v \in S \setminus U$, we have $|N(v) \cap S| \in \sigma$, and

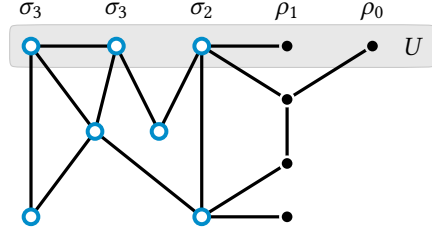


Figure 8.1: A graph G with the portals U and a set of selected vertices S that are represented by highlighted blue circles.

When $\sigma = \{2, 4\}$ and $\rho = \{1\}$, the set S is a partial solution (with respect to U), as every selected vertex in $S \setminus U$ satisfies $|N(s) \cap S| \in \{2, 4\} = \sigma$ and every unselected vertex in $V(G) \setminus (S \cup U)$ satisfies $|N(v) \cap S| \in \{1\} = \rho$. Moreover, set S corresponds to the compatible vector $\sigma_3\sigma_3\sigma_2\rho_1\rho_0$.

(PS2) for each $v \in V(G) \setminus (S \cup U)$, we have $|N(v) \cap S| \in \rho$.

To describe whether vertices from U are selected into partial solutions and how many selected neighbors they already have inside G , we associate a state with every vertex from U .

Formally, we write $\mathbb{S}_{\text{full}} := \{\sigma_i \mid i \in \mathbb{N}\}$ for the set of potential σ -states, and we write $\mathbb{R}_{\text{full}} := \{\rho_i \mid i \in \mathbb{N}\}$ for the set of potential ρ -states. We also write $\mathbb{A}_{\text{full}} := \mathbb{S}_{\text{full}} \cup \mathbb{R}_{\text{full}}$ for the set of all potential states.

With this notation, we can define the set of state-vectors that are compatible with a graph with portals. We refer to Figure 8.1 for a visualization of an example of a partial solution and its corresponding compatible vectors.

Definition 8.1.9 (Compatible State-Vectors). Fix a graph with portals (G, U) . A vector $x \in \mathbb{A}_{\text{full}}^U$ is compatible with (G, U) if there is a partial solution $S_x \subseteq V(G)$ such that

1. for each $v \in U \cap S_x$, we have $x[v] = \sigma_s$, where $s = |N(v) \cap S_x|$, and
2. for each $v \in U \setminus S_x$, we have $x[v] = \rho_r$, where $r = |N(v) \cap S_x|$.

We also refer to the vertices in S_x as being selected and say that S_x is a (partial) solution, selection, or witness that witnesses x .

Observe that, although $\mathbb{A}_{\text{full}}$ is an infinite alphabet, for every graph with portals (G, U) , only finitely many vectors x can be realized. Indeed, if $|V(G)| = n$, then every compatible vector can have only states from $\mathbb{A}_n = \mathbb{S}_n \cup \mathbb{R}_n$, where $\mathbb{S}_n := \{\sigma_i \mid i \in [0..n]\}$ and $\mathbb{R}_n := \{\rho_i \mid i \in [0..n]\}$.

We refer to the set of all vectors that are compatible with a graph with portals as the realized language.

Definition 8.1.10 (Realized Language and L -Providers). For a graph with portals (G, U) , we define realized language as

$$L(G, U) := \{x \in \mathbb{A}_{\text{full}}^U \mid x \text{ is compatible with } (G, U)\}.$$

For a language $L \subseteq \mathbb{A}_{\text{full}}^U$, we say that (G, U) is an L -realizer if $L = L(G, U)$, and we say that (G, U) is an L -provider if $L \subseteq L(G, U)$.

Again, we observe that $L(G, U) \subseteq \mathbb{A}_n^U$, where $n = |V(G)|$.

In fact, for most of our applications, it makes sense to restrict the alphabet even further. Recall the definition of s_{top} and r_{top} from Equation (8.1.1). Suppose that σ is finite. Then, we are usually not interested in partial solutions S where some vertex from U is selected and already has more than $s_{\text{top}} = \max \sigma$ selected neighbors as it is impossible to extend this partial solution into a full solution. Also, if σ is infinite, it is usually irrelevant to us whether a selected vertex has exactly s_{top} selected neighbors, or more than s_{top} selected neighbors, since both options lead to the same outcome for all possible extensions of a partial solution. For this reason, we typically restrict ourselves to the alphabets

$$\mathbb{S} := \{\sigma_0, \dots, \sigma_{s_{\text{top}}}\} \quad \text{and} \quad \mathbb{R} := \{\rho_0, \dots, \rho_{r_{\text{top}}}\}.$$

As before, we define $\mathbb{A} := \mathbb{S} \cup \mathbb{R}$.

Sometimes, it turns out to be more convenient to work with the more general variants. We clearly mark these rare occurrences of $\mathbb{A}_{\text{full}}$ and $\mathbb{A}_n$.

Solving General Domination

In the following, we prove the algorithmic results for (σ, ρ) -DOMSET parameterized by treewidth. Instead of providing different algorithms for the decision and counting version, we present unified algorithms.

For the first result, with Theorem 8.1.2 in mind, we can restrict ourselves to the case where (σ, ρ) is m -structured for some $m \geq 2$. In particular, in our case this implies that both σ and ρ are finite. Note that in general this is not true, as, for example, the even numbers are 2-structured. Formally, we prove Theorem 9.1.24.

Theorem 9.1.24. *Fix two finite non-empty sets $\sigma, \rho \subseteq \mathbb{N}$ such that (σ, ρ) is m -structured for some $m \geq 2$. Then, there is an algorithm $\mathcal{A}$ that, given a graph G , a nice tree decomposition of G of width w , and an integer s , computes the number of (σ, ρ) -sets of G of size exactly s .*

If $m \geq 3$ or t_{top} is odd or $s_{\text{top}} \neq r_{\text{top}}$, then algorithm $\mathcal{A}$ runs in time

$$(t_{\text{top}} + 1)^w \cdot 2^{(t_{\text{top}})^{\mathcal{O}(1)}} \cdot |G|^{\mathcal{O}(1)}.$$

If $m = 2$, t_{top} is even, and $s_{\text{top}} = r_{\text{top}} = t_{\text{top}}$, then algorithm $\mathcal{A}$ runs in time

$$(t_{\text{top}} + 2)^w \cdot 2^{(t_{\text{top}})^{\mathcal{O}(1)}} \cdot |G|^{\mathcal{O}(1)}.$$

Together with the known algorithms by van Rooij for the general case as stated in Theorem 8.1.2, this proves Main Theorem III.1.

Main Theorem III.1. *Fix two finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$. Given a graph G with a tree decomposition of width w and an integer s , we can count the number of (σ, ρ) -sets of size exactly s in time $(c_{\sigma, \rho})^w \cdot |G|^{\mathcal{O}(1)}$.*

Proof. When (σ, ρ) are not m -structured for any $m \geq 2$, we use the algorithm by van Rooij from Theorem 8.1.2 (see also Theorem 1 in [131]). Otherwise, we use the algorithm from Theorem 9.1.24. The final running time follows from the definition of $c_{\sigma, \rho}$ in Definition 8.1.4. $\square$

As a second result, we give an improved algorithm for the case when at least one of σ and ρ is cofinite. In this case we can achieve a significant improvement over the general algorithm. Recall that, for a finite set τ , we define $\text{cost } \tau = \max \tau$ and, for a cofinite set τ , we define $\text{cost } \tau = \max(\mathbb{N} \setminus \tau) + 1$. See Definition 8.1.7 for the formal statement.

Main Theorem III.2. *Fix two finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$. Based on these sets we define $t_{\text{cost}} := \max\{\text{cost } \sigma, \text{cost } \rho\}$.*

Given a graph G with a tree decomposition of width w , and an integer s , we can decide whether G has a (σ, ρ) -set of size exactly s in time

$$2^w \cdot (t_{\text{cost}} + 1)^{w \cdot (\omega + 1)} \cdot |G|^{\mathcal{O}(1)}.$$

In contrast to the result from Theorem 9.1.24 we cannot extend this algorithm to the counting version of the problem. Note that this limitation is inherent to the design of the algorithm; the fundamental idea of our approach is to not store all the partial solutions, which would be necessary for the counting version. Although this limitation occurs because we use representative sets, Main Theorem III.4 implies that there is no algorithm counting (σ, ρ) -sets in time $(f(t_{\text{cost}}))^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ for any function f assuming #SETH.

Comparison to Previous Algorithms. Before we move toward the proofs of these algorithms, let us briefly compare the running times from the new algorithm and the existing algorithm by van Rooij [131] (see also Theorem 8.1.2). To this end, we recall the definition of top τ for a finite or cofinite set $\tau \subseteq \mathbb{N}$. If τ is finite, then we define $\text{top } \tau := \max \tau$. Otherwise, if τ is cofinite, then we define $\text{top } \tau := \max(\mathbb{Z} \setminus \tau) + 1$. Observe that the definition of cost and top differ for the case of cofinite sets.

Note that $\text{cost } \tau \leq \text{top } \tau$ for all finite or cofinite sets $\tau \subseteq \mathbb{N}$. Moreover, for a cofinite set $\tau \subseteq \mathbb{N}$, we have that $\text{cost } \tau = \text{top } \tau$ if and only if $\tau = \{c, c+1, c+2, \dots\}$ for some number $c \in \mathbb{N}$.

The running time of the algorithm by van Rooij from Theorem 8.1.2 is

$$\left(\text{top } \sigma + \text{top } \rho + 2 \right)^{\text{tw}} \cdot n^{\mathcal{O}(1)}.$$

On the other hand, the algorithm from Main Theorem III.2 runs in time

$$\left(2 \cdot (\max\{\text{cost } \sigma, \text{cost } \rho\} + 1)^{\omega+1} \right)^{\text{tw}} \cdot n^{\mathcal{O}(1)}.$$

If σ and ρ are both finite, then the algorithm by van Rooij is clearly faster using that $\text{top } \sigma = \text{cost } \sigma$ and $\text{top } \rho = \text{cost } \rho$ (but, in this case, Theorem 9.1.24 provides an improved algorithm for structured sets).

However, if at least one of the sets σ, ρ is cofinite, then our algorithm may be substantially faster since $\text{cost } \tau$ can be arbitrarily smaller than $\text{top } \tau$ for a cofinite set τ .

As a concrete example, suppose that $\rho := \mathbb{N} \setminus \{c\}$ and $\sigma := \mathbb{N} \setminus \{d\}$. Then, $\text{cost } \rho = \text{cost } \sigma = 1$, but $\text{top } \rho = c+1$ and $\text{top } \sigma = d+1$. Hence, van Rooij's algorithm runs in time $(c+d+4)^{\text{tw}} \cdot n^{\mathcal{O}(1)}$, where the algorithm from Main Theorem III.2 takes time $2^{\text{tw} \cdot (\omega+2)} \cdot n^{\mathcal{O}(1)}$ which is at most $20.7^{\text{tw}} \cdot n^{\mathcal{O}(1)}$. Observe that the second running time is independent of c and d and thus, can yield significant improvements.

Outline. In the following, we first prove Theorem 9.1.24 in Section 9.1. Afterward, we prove Main Theorem III.2 in Section 9.2.

9.1 Parameterizing by the Largest Degree

The goal of this section is to provide a faster algorithm for (σ, ρ) -DOMSET when the sets σ and ρ are structured, that is, we prove Theorem 9.1.24.

As we assume that one set is cofinite, our algorithm from Theorem 9.1.24 does not apply.

Theorem 9.1.24. *Fix two finite non-empty sets $\sigma, \rho \subseteq \mathbb{N}$ such that (σ, ρ) is m -structured for some $m \geq 2$. Then, there is an algorithm $\mathcal{A}$ that, given a graph G , a nice tree decomposition of G of width w , and an integer s , computes the number of (σ, ρ) -sets of G of size exactly s .*

If $m \geq 3$ or t_{top} is odd or $s_{\text{top}} \neq r_{\text{top}}$, then algorithm $\mathcal{A}$ runs in time

$$(t_{\text{top}} + 1)^w \cdot 2^{(t_{\text{top}})^{\mathcal{O}(1)}} \cdot |G|^{\mathcal{O}(1)}.$$

If $m = 2$, t_{top} is even, and $s_{\text{top}} = r_{\text{top}} = t_{\text{top}}$, then algorithm $\mathcal{A}$ runs in time

$$(t_{\text{top}} + 2)^w \cdot 2^{(t_{\text{top}})^{\mathcal{O}(1)}} \cdot |G|^{\mathcal{O}(1)}.$$

We observe that Theorem 9.1.24 solves the decision, optimization and counting version of the problem simultaneously. If we restrict our attention to the decision version, then we could improve the algorithm and achieve a running time with smaller polynomial dependence. However, to have a unified and versatile algorithm, we give a combined algorithm which solves the more general exact counting problem although having slightly larger dependence on the size of the input graph.

Finally, note that t_{top} only depends on the two sets σ and ρ , and thus, the term $2^{(t_{\text{top}})^{\mathcal{O}(1)}}$ is a fixed constant for any specific (σ, ρ) -DOMSET problem.

Idea of the Algorithm. As for most algorithms parameterized by treewidth, our approach uses a dynamic programming method that operates on the given tree decomposition. For each node of the tree decomposition, we store a set of state-vectors that correspond to those partial solutions that can be achieved for the corresponding part of the graph.

For a more detailed description, first suppose for simplicity that both σ and ρ are finite. Now, in our case, a partial solution at a node t is a set $S \subseteq V_t$ such that $|N(v) \cap S| \in \sigma$ for every $v \in S \setminus \beta(t)$, and $|N(v) \cap S| \in \rho$ for every $v \in V_t \setminus (S \cup \beta(t))$, that is, all vertices of V_t outside $\beta(t)$ satisfy the σ -constraints and ρ -constraints, but the vertices in $\beta(t)$ may not. In particular, it may happen that a vertex $v \in \beta(t) \setminus S$ or $u \in \beta(t) \cap S$ does not yet have a correct number of selected neighbors, that is, $|N(v) \cap S| \notin \rho$ and $|N(u) \cap S| \notin \sigma$ is possible, since vertices v and u may receive additional selected neighbors that lie outside V_t .

Based on precisely these vertices in $\beta(t)$, we define the type of a partial solution S . More precisely, for each $v \in \beta(t)$, we specify whether $v \in S$ and how many selected neighbors v has. Hence, we can describe such a type by a state-vector $y \in \mathbb{A}^{\beta(t)}$, where $\mathbb{A} = \{\sigma_0, \dots, \sigma_{s_{\text{top}}}, \rho_0, \dots, \rho_{r_{\text{top}}}\}$, that associates with every $v \in \beta(t)$ a state $y[v] \in \mathbb{A}$,

- where state σ_i means that $v \in S$ and v has i selected neighbors in S , and
- where state ρ_j means that $v \notin S$ and v has j selected neighbors in S .

Observe that we can immediately dismiss partial solutions that assign too many selected neighbors to a vertex, that is, for instance, a state σ_i for $i > s_{\text{top}}$, since such partial solutions can never be extended to a full solution. This gives a trivial upper bound of $(s_{\text{top}} + r_{\text{top}} + 2)^{tw+1}$ for the number of types; and yields essentially the known algorithms.

As we assume that the sets are finite or cofinite, this is always the case when (σ, ρ) is m -structured for some $m \geq 2$.

For cofinite sets the states σ_i with $i \geq s_{\text{top}}$ are treated equivalent instead.

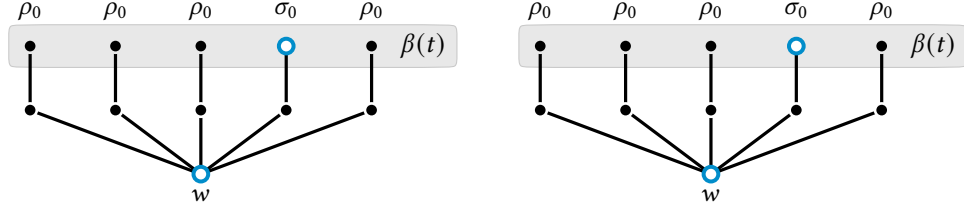


Figure 9.1: An example for DOMINATING SET (that is, (σ, ρ) -DOMSET with $\sigma := \{0, 1, 2, \dots\}$ and $\rho := \{1, 2, \dots\}$), where every vector $y \in \mathbb{A}^{\beta(t)} = \{\sigma_0, \rho_0, \rho_1\}^{\beta(t)}$ is compatible with t . Here every selection of the vertices is valid after selecting vertex w , we show two possibilities where the selected vertices are highlighted. Note that here the states σ_1 and σ_0 are identical.

However, it was long unknown if the trivial upper bound is tight. Phrased differently, is it really possible that for every vector $y \in \mathbb{A}^{\beta(t)}$ there is some partial solution S_y that corresponds to y ?

Note that here the sets are cofinite, but the ideas from above transfer directly.

In general, this turns out to be indeed the case. Consider the classical DOMINATING SET problem, that is, (σ, ρ) -DOMSET where $\sigma := \{0, 1, 2, \dots\}$ and $\rho := \{1, 2, \dots\}$, and the following example. Suppose that for some bag $\beta(t)$, each of its vertices $v_i \in \beta(t)$ has a single neighbor $v'_i \in V_t \setminus \beta(t)$, and suppose that all vertices v'_i share a common neighbor $w \in V_t \setminus X_t$. Consult Figure 9.1 for a visualization of this example.

Now, for any vector $y \in \mathbb{A}^{\beta(t)} = \{\sigma_0, \rho_0, \rho_1\}^{\beta(t)}$, there is indeed a partial solution; once we select w , any selection of v'_i or v_i is a valid partial solution. That is, each vector $y \in \mathbb{A}^{\beta(t)}$ is indeed *compatible* with t . In particular, there are $(s_{\text{top}} + r_{\text{top}} + 2)^{|\beta(t)|} = (0 + 1 + 2)^{|\beta(t)|} = 3^{|\beta(t)|}$ vectors compatible with t . Thus, when $|\beta(t)| = tw + 1$ the trivial upper bound on the number of types is indeed tight.

In stark contrast to this situation, we show that for m -structured (σ, ρ) where $m \geq 2$, not all $(s_{\text{top}} + r_{\text{top}} + 2)^{tw+1}$ different vectors can be compatible with t . Vastly simplified, we observe in multiple steps, that for each vertex of $\beta(t)$, we can either choose whether it is selected or how many selected neighbors this vertex has. The other part of the state is then determined by the other positions of the vector.

In Section 9.1.1 we formally prove the structural insights and prove an upper bound on the number of states that need to be maintained by the dynamic programming algorithm.

Fast Join Operations. At this point, we face a second challenge. When performing dynamic programming along a tree decomposition, the most expensive step is to perform a join operation. In this situation, the current node t has exactly two children t_1 and t_2 with $\beta(t) = \beta(t_1) = \beta(t_2)$.

In Section 3.1 we provide various convolution-based methods to efficiently compute the set of compatible vectors for t given the sets of compatible vectors for t_1 and t_2 . We wish to apply these methods to obtain faster algorithms. Although this is possible, a direct application does not yield an improved algorithm. The main reason is that the convolution-based methods are not designed to handle restrictions of the input space.

To circumvent this issue, we design a specialized compression method that again exploits the structure described above. We base this compression on the same observations that lead

to the better bounds on the number of states, which we encounter at each node. However, the crux is that we need to design the compression in such a way that it agrees with the join operation: two compatible vectors y_1 and y_2 for t_1 and t_2 can be joined into a compatible vector y for t if and only if the compressed vectors y'_1 and y'_2 can be joined into the compressed vector y' .

We formally prove these results in Section 9.1.2, and we show how to efficiently merge such states using a fast convolution-based algorithm.

Finalizing the Algorithm. Finally, we combine all previous results to obtain the faster algorithm for (σ, ρ) -DOMSET when the sets are structured, that is, we prove Theorem 9.1.24 in Section 9.1.3.

9.1.1 Structural Insights

In the following we provide some structural insight into the language realized by a graph with portals to eventually bound the size of the language. Phrased slightly more formal, for a graph with portals (G, U) , we prove a bound on the size of the realized language $L(G, U)$ in terms of s_{top} and r_{top} . Although we do not provide a *lower* bound for this size bound, it seems plausible that our bound is optimal (that is, tight). Indeed, an even better bound on the size of the realized language would yield an even faster algorithm which is not possible assuming the SETH (see Main Theorem III.3 for a precise statement).

By the design of our algorithm and the requirement of (σ, ρ) to be m-structured, we consider only languages which we refer to as *quasi-orthogonal*; see Definition 9.1.3 for the formalization. With this notation, we prove Main Lemma 9.1.11 as one of the main results.

Main Lemma 9.1.11. *Let $L \subseteq \mathbb{A}^n$ denote a quasi-orthogonal language.*

If $m \geq 3$ or t_{top} is odd or $\min\{s_{\text{top}}, r_{\text{top}}\} < t_{\text{top}}$, then $|L| \leq (t_{\text{top}} + 1)^n$.

If $m = 2$, t_{top} is even, and $s_{\text{top}} = r_{\text{top}} = t_{\text{top}}$, then $|L| \leq (t_{\text{top}} + 2)^n$.

We first observe that we can restrict ourselves to the states from the alphabet $\mathbb{A} := \mathbb{S} \dot{\cup} \mathbb{R}$ where we recall that $\mathbb{S} := \{\sigma_0, \dots, \sigma_{s_{\text{top}}}\}$ and $\mathbb{R} := \{\rho_0, \dots, \rho_{r_{\text{top}}}\}$.

Thereby, we also bound the number of states that are required in our dynamic programming based approach for computing a solution for a given graph G . To that end, we first define three types of vectors associated with each state-vector $x \in \mathbb{A}^n$, which essentially represent a decomposition of the state-vector into its σ/ρ -component and its “weight”-component.

To be able to reuse the definition in later sections, we state it for the full alphabet $\mathbb{A}_{\text{full}}$.

Definition 9.1.1. *For a state-vector $x \in \mathbb{A}_{\text{full}}^n$, we define*

- *the σ -vector of x as $\vec{\sigma}(x) \in \{0, 1\}^n$ with*

$$\vec{\sigma}(x)[i] := \begin{cases} 1 & \text{if } x[i] \in \mathbb{S}, \\ 0 & \text{if } x[i] \in \mathbb{R}. \end{cases}$$

- *the weight-vector of x as $\vec{w}(x) \in \mathbb{N}^n$ with*

$$\vec{w}(x)[i] := c, \quad \text{where } x[i] \in \{\sigma_c, \rho_c\}.$$

- the m -weight-vector of x as $\vec{w}_m(x) \in [0 \dots m]^n$ with

$$\vec{w}_m(x)[i] := \vec{w}(x)[i] \bmod m.$$

For a language $L \subseteq \mathbb{A}_{\text{full}}^n$, we write $\vec{\sigma}(L) := \{\vec{\sigma}(x) \mid x \in L\}$ for the set of all σ -vectors of L , we write $\vec{w}(L) := \{\vec{w}(x) \mid x \in L\}$ for the set of all weight-vectors of L , and we write $\vec{w}_m(L) := \{\vec{w}_m(x) \mid x \in L\}$ for the set of all m -weight-vectors of L .

Finally, for a vector $\vec{s} \in \{0, 1\}^n$, we define the capacity-vector of $\vec{s}$ as $\text{cap}(\vec{s}) \in [0 \dots t_{\text{top}}]^n$ with

$$\text{cap}(\vec{s})[i] := \begin{cases} s_{\text{top}} & \text{if } \vec{s}[i] = 1, \\ r_{\text{top}} & \text{if } \vec{s}[i] = 0. \end{cases}$$

To bound the size of realized languages, we proceed as follows. First, we compare two different partial solutions with respect to a fixed set U of portal vertices to obtain certain frequency properties of the characters of the corresponding state-vectors in $\mathbb{A}^U$ (expressed as σ -vectors and m -weight-vectors). In a second step, we then show that there is only a moderate number of state-vectors with said structure.

Our first and main structural insight reads as follows.

Lemma 9.1.2. *Let (G, U) be a graph with portals and let $L := L(G, U) \subseteq \mathbb{A}^U$ denote its realized language. Also let $x, y \in L$ denote two state-vectors with witnesses $S_x, S_y \subseteq V(G)$ such that $|S_x \setminus U| \equiv_m |S_y \setminus U|$. Then, we have $\vec{\sigma}(x) \cdot \vec{w}_m(y) \equiv_m \vec{\sigma}(y) \cdot \vec{w}_m(x)$.*

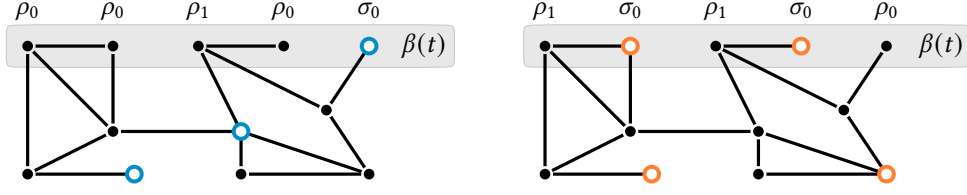
For an intuition for Lemma 9.1.2 let us move to the EXACT INDEPENDENT DOMINATING SET problem (also known as PERFECT CODE) as a specific example which corresponds to $\sigma = \{0\}$ and $\rho = \{1\}$. We refer to Figure 9.2a for a visualization of an example. Observe that $(\{0\}, \{1\})$ is indeed 2-structured.

Consider two partial solutions $S_x, S_y \subseteq V_t$. Note that both S_x and S_y are independent sets in G since $\sigma = \{0\}$. Now, let us count the edges between S_x and S_y . To that end, we define X_0 as the set of vertices from $V_t \setminus S_x$ that have no selected neighbor (that is, no neighbor in S_x), and we define X_1 as the set of vertices from $V_t \setminus S_x$ that have *one* selected neighbor (observe that $X_0 \subseteq \beta(t)$ since $\rho = \{1\}$). Observe that (S_x, X_0, X_1) forms a partition of V_t . We define Y_0 and Y_1 analogously for the partial solution S_y .

Now, every vertex in $S_y \cap X_0$ has no neighbor in S_x , while every vertex in $S_y \cap X_1$ has exactly one neighbor in S_x . Recalling that vertices from $S_y \cap S_x$ have no neighbor in S_x (since S_x is an independent set), the number of edges from S_y to S_x equals $|S_y \cap X_1|$. Repeating the same argument with reversed roles, we get that the number of edges from S_x to S_y equals $|S_x \cap Y_1|$. So $|S_y \cap X_1| = |S_x \cap Y_1|$. Assuming that $\beta(t) = V_t$, this condition is equivalent to $\vec{\sigma}(x) \cdot \vec{w}(y) = \vec{\sigma}(y) \cdot \vec{w}(x)$. To obtain the conclusion of the lemma also for $\beta(t) \neq V_t$, we additionally use the assumption that $|S_x \setminus \beta(t)| \equiv_2 |S_y \setminus \beta(t)|$ and $X_0, Y_0 \subseteq \beta(t)$. We refer to Figure 9.2b for a visualization of this proof sketch for the example from Figure 9.2a.

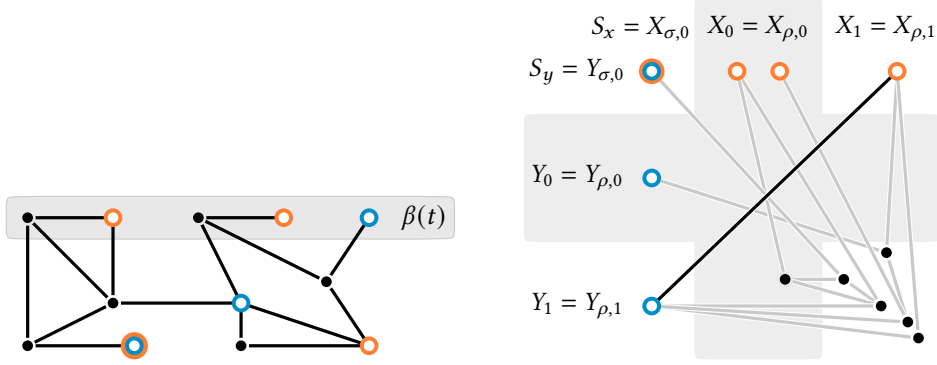
With this intuition, we move to the formal proof of Lemma 9.1.2.

Lemma 9.1.2. *Let (G, U) be a graph with portals and let $L := L(G, U) \subseteq \mathbb{A}^U$ denote its realized language. Also let $x, y \in L$ denote two state-vectors with witnesses $S_x, S_y \subseteq V(G)$ such that $|S_x \setminus U| \equiv_m |S_y \setminus U|$. Then, we have $\vec{\sigma}(x) \cdot \vec{w}_m(y) \equiv_m \vec{\sigma}(y) \cdot \vec{w}_m(x)$.*



(a) A bag $\beta(t)$ together with the vertices in V_t . The vectors $x = \rho_0\rho_0\rho_1\rho_0\sigma_0$ and $y = \rho_1\sigma_0\rho_1\sigma_0\rho_0$ are compatible with t ; the corresponding partial solutions S_x and S_y are shown on the left and right where the selected vertices are encircled in blue and orange, respectively.

We have $|S_x \setminus \beta(t)| = |S_y \setminus \beta(t)| = 2$ and $\vec{\sigma}(x) \cdot \vec{w}(y) = (0, 0, 0, 0, 1) \cdot (1, 0, 1, 0, 0) = 0 \equiv_2 0 = (0, 1, 0, 1, 0) \cdot (0, 0, 1, 0, 0) = \vec{\sigma}(y) \cdot \vec{w}(x)$.



(b) The partial solutions S_x and S_y depicted together. Observe that edges between a vertex $v_x \in S_x$ and $v_y \in S_y$ are possible only if v_x and v_y have exactly one selected neighbor in the corresponding other partial solution.

Figure 9.2: An example for partial solutions and edges between them for PERFECT CODE as an illustration for Lemma 9.1.2. As this corresponds to (σ, ρ) -DOMSET with $\sigma := \{0\}$ and $\rho := \{1\}$, the sets (σ, ρ) are 2-structured.

Proof. Fix a graph G , a subset of its vertices $U \subseteq V(G)$, and the realized language $L := L(G, U) \subseteq \mathbb{A}^U$. For a state-vector $x \in L$ and an integer $m \geq 2$, we define the ordered partition

$$P_m(x) := (X_{\sigma,0}, \dots, X_{\sigma,m-1}, X_{\rho,0}, \dots, X_{\rho,m-1}) \text{ of } U$$

with

$$X_{\sigma,i} := \{v \in U \mid \vec{\sigma}(x)[v] = 1 \text{ and } \vec{w}_m(x)[v] = i\} \quad \text{for all } i \in [0 \dots m),$$

and

$$X_{\rho,i} := \{v \in U \mid \vec{\sigma}(x)[v] = 0 \text{ and } \vec{w}_m(x)[v] = i\} \quad \text{for all } i \in [0 \dots m).$$

In a first step, we count the number of edges between S_x and S_y in two different ways to obtain the following result.

Claim 9.1.2.1. *We have*

$$\sum_{i \in [1 \dots m)} \sum_{j \in [0 \dots m)} i \cdot (|X_{\sigma,j} \cap Y_{\rho,i}| + |X_{\sigma,j} \cap Y_{\sigma,i}|) \equiv_m \sum_{i \in [1 \dots m)} \sum_{j \in [0 \dots m)} i \cdot (|Y_{\sigma,j} \cap X_{\rho,i}| + |Y_{\sigma,j} \cap X_{\sigma,i}|).$$

Proof of Claim. The corresponding ordered partitions are

$$P_m(x) = (X_{\sigma,0}, \dots, X_{\sigma,m-1}, X_{\rho,0}, \dots, X_{\rho,m-1}) \text{ and } P_m(y) = (Y_{\sigma,0}, \dots, Y_{\sigma,m-1}, Y_{\rho,0}, \dots, Y_{\rho,m-1}).$$

Recall the integers $B_\sigma, B_\rho \in [0..m]$ with $s \equiv_m B_\sigma$ for every $s \in \sigma$ and $r \equiv_m B_\rho$ for every $r \in \rho$. Observe that by (PS2) from Definition 8.1.8, every (unselected) vertex in $V(G) \setminus (U \cup S_y)$ has $(B_\rho + m \cdot \ell)$ neighbors in S_y for some non-negative integer ℓ . In particular, this holds for all vertices in $S_x \setminus (U \cup S_y)$.

Similarly, observe that by (PS1), every (selected) vertex in $(S_x \cap S_y) \setminus U$ has $(B_\sigma + m \cdot \ell')$ neighbors in S_y for some non-negative integer ℓ' . Finally, a vertex v in $S_x \cap U = \bigcup_{j \in [0..m)} X_{\sigma,j}$ has exactly $i + m \cdot \ell''$ neighbors in S_y for some non-negative integer ℓ'' if and only if v is in one of $Y_{\rho,i}$ or $Y_{\sigma,i}$.

When we write $\vec{E}(X, Y) := \{(v, w) \in E(G) \mid v \in X, w \in Y\}$ for the edges between the vertex sets X and Y , we obtain

$$\begin{aligned} |\vec{E}(S_x, S_y)| &\equiv_m B_\rho \cdot |S_x \setminus (U \cup S_y)| \\ &\quad + B_\sigma \cdot |(S_x \cap S_y) \setminus U| \\ &\quad + \sum_{i \in [1..m)} \sum_{j \in [0..m)} i \cdot (|X_{\sigma,j} \cap Y_{\rho,i}| + |X_{\sigma,j} \cap Y_{\sigma,i}|) \\ &\equiv_m B_\rho \cdot (|S_x \setminus U| - |S_x \cap S_y| + |S_x \cap S_y \cap U|) \\ &\quad + B_\sigma \cdot |(S_x \cap S_y) \setminus U| \\ &\quad + \sum_{i \in [1..m)} \sum_{j \in [0..m)} i \cdot (|X_{\sigma,j} \cap Y_{\rho,i}| + |X_{\sigma,j} \cap Y_{\sigma,i}|). \end{aligned}$$

Symmetrically, we count the edges from S_y to S_x and get

$$\begin{aligned} |\vec{E}(S_y, S_x)| &\equiv_m B_\rho \cdot (|S_y \setminus U| - |S_y \cap S_x| + |S_y \cap S_x \cap U|) \\ &\quad + B_\sigma \cdot |(S_y \cap S_x) \setminus U| \\ &\quad + \sum_{i \in [1..m)} \sum_{j \in [0..0)m} i \cdot (|Y_{\sigma,j} \cap X_{\rho,i}| + |Y_{\sigma,j} \cap X_{\sigma,i}|). \end{aligned}$$

Now, we combine the previous equations and use the assumption that $|S_x \setminus U| \equiv_m |S_y \setminus U|$ to obtain

$$\sum_{i \in [1..m)} \sum_{j \in [0..m)} i \cdot (|X_{\sigma,j} \cap Y_{\rho,i}| + |X_{\sigma,j} \cap Y_{\sigma,i}|) \equiv_m \sum_{i \in [1..m)} \sum_{j \in [0..m)} i \cdot (|Y_{\sigma,j} \cap X_{\rho,i}| + |Y_{\sigma,j} \cap X_{\sigma,i}|)$$

which concludes the proof. $\triangleleft$

Next, we unfold the definitions for $X_{\star,\star}, Y_{\star,\star}$ and observe that, for every $i \in [1..m)$, it holds that

$$\begin{aligned} &\sum_{j \in [0..m)} i \cdot (|X_{\sigma,j} \cap Y_{\rho,i}| + |X_{\sigma,j} \cap Y_{\sigma,i}|) \\ &\equiv_m \sum_{j \in [0..m)} i \cdot |\{k \in [1..n] \mid \vec{\sigma}(x)[k] = 1, \vec{w}_m(x)[k] = j, \text{ and } \vec{w}_m(y)[k] = i\}| \\ &\equiv_m i \cdot |\{k \in [1..n] \mid \vec{\sigma}(x)[k] = 1 \text{ and } \vec{w}_m(y)[k] = i\}|. \end{aligned}$$

It follows that

$$\begin{aligned}
 & \sum_{i \in [1 \dots m]} \sum_{j \in [0 \dots m]} i \cdot (|X_{\sigma,j} \cap Y_{\rho,i}| + |X_{\sigma,j} \cap Y_{\sigma,i}|) \\
 & \equiv_m \sum_{i \in [1 \dots m]} i \cdot |\{k \in [1 \dots n] \mid \vec{\sigma}(x)[k] = 1 \text{ and } \vec{w}_m(y)[k] = i\}| \\
 & \equiv_m \vec{\sigma}(x) \cdot \vec{w}_m(y).
 \end{aligned}$$

Hence, by using the symmetrical argument for the other sum, we obtain the claimed equivalence $\vec{\sigma}(x) \cdot \vec{w}_m(y) \equiv_m \vec{\sigma}(y) \cdot \vec{w}_m(x)$, which completes the proof. $\square$

From now on, we focus on those languages that satisfy the equivalence stated in Lemma 9.1.2. For these languages we show a bound on the size of a realized language $L \subseteq \mathbb{A}^n$.

In order to simplify notation, we define a relation that precisely captures the property from Lemma 9.1.2.

Definition 9.1.3 (Quasi-Orthogonal Languages). *For an integer n , we define*

$$\mathcal{R}^n := \{(x, y) \in \mathbb{A}^n \times \mathbb{A}^n \mid \vec{\sigma}(x) \cdot \vec{w}_m(y) \equiv_m \vec{\sigma}(y) \cdot \vec{w}_m(x)\}. \quad (9.1.1)$$

We say that a language $L \subseteq \mathbb{A}^n$ is quasi-orthogonal if $L \times L \subseteq \mathcal{R}^n$.

Observe that the relation $\mathcal{R}^n$ is reflexive and symmetric and hence, the definition of being quasi-orthogonal is well-defined. Note however, that the relation $\mathcal{R}^n$ is *not* transitive.

We proceed to exploit the quasi-orthogonality to obtain size bounds for realized languages in Section 9.1.1.1. Afterward, in Section 9.1.1.2, we investigate how these size bounds behave when we combine realized languages.

9.1.1.1 A Single Realized Language

The goal of this section is to show the following result.

Main Lemma 9.1.11. *Let $L \subseteq \mathbb{A}^n$ denote a quasi-orthogonal language.*

If $m \geq 3$ or t_{top} is odd or $\min\{s_{\text{top}}, r_{\text{top}}\} < t_{\text{top}}$, then $|L| \leq (t_{\text{top}} + 1)^n$.

If $m = 2$, t_{top} is even, and $s_{\text{top}} = r_{\text{top}} = t_{\text{top}}$, then $|L| \leq (t_{\text{top}} + 2)^n$.

Note that the bounds of Main Lemma 9.1.11 are essentially optimal. We consider an example to see why this is true.

Consider the three languages

- $L_1 := \mathbb{R}^n = \{v \in \mathbb{A}^n \mid \vec{\sigma}(v)[\ell] = 0 \text{ for all } \ell \in [1 \dots n]\},$
- $L_2 := \{v \in \mathbb{S}^n \mid \sum_{\ell \in [1 \dots n]} \vec{w}(v)[\ell] \equiv_m 0\},$ and
- $L_3 := \{v \in \mathbb{A}^n \mid \vec{w}(v)[\ell] \equiv_m 0 \text{ for all } \ell \in [1 \dots n]\}.$

It is straightforward to see that L_i is quasi-orthogonal for all $i \in \{1, 2, 3\}$. We have that $|L_1| = (r_{\text{top}} + 1)^n$, $|L_2| \geq (s_{\text{top}} + 1)^{n-1}$ and

$$|L_3| = \left(\left\lceil \frac{s_{\text{top}} + 1}{m} \right\rceil + \left\lceil \frac{r_{\text{top}} + 1}{m} \right\rceil \right)^n.$$

Observe that $|L_3| = (t_{\text{top}} + 2)^n$ if $m = 2$, t_{top} is even, and $r_{\text{top}} = s_{\text{top}} = t_{\text{top}}$. In all other cases, $|L_3| \leq (t_{\text{top}} + 1)^n$. In particular, the language L_3 indicates why the case $m = 2$ with even $s_{\text{top}} = r_{\text{top}} = t_{\text{top}}$ stands out.

Toward the proof of Main Lemma 9.1.11, we first show that, for state-vectors with the same σ -vector, the difference of their m -weight-vectors is “orthogonal” to the σ -vector of any other state-vector.

Lemma 9.1.4. *Let $L \subseteq \mathbb{A}^n$ denote a quasi-orthogonal language. For three arbitrary state-vectors $v, w, z \in L$ with $\vec{\sigma}(v) = \vec{\sigma}(w)$, we have*

$$(\vec{w}_m(v) - \vec{w}_m(w)) \cdot \vec{\sigma}(z) \equiv_m 0.$$

Proof. Fix three state-vectors $v, w, z \in L$. By the definition of quasi-orthogonality, we have

$$\vec{\sigma}(v) \cdot \vec{w}_m(z) \equiv_m \vec{\sigma}(z) \cdot \vec{w}_m(v) \quad \text{and} \quad \vec{\sigma}(w) \cdot \vec{w}_m(z) \equiv_m \vec{\sigma}(z) \cdot \vec{w}_m(w).$$

Using the assumption that the two state-vectors v and w share one σ -vector, that is, $\vec{\sigma}(v) = \vec{\sigma}(w)$, we conclude that

$$\vec{\sigma}(z) \cdot \vec{w}_m(v) \equiv_m \vec{\sigma}(z) \cdot \vec{w}_m(w),$$

which finishes the proof after rearranging. $\square$

Next, we explore the implications of Lemma 9.1.4. Intuitively, we show that, for a language L of state-vectors of length n , each of the n positions contributes either to the σ -vectors from $\vec{\sigma}(L)$ or to weight-vectors from $\vec{w}_m(L)$. Formally, let us start with the notion of a σ -defining set.

Definition 9.1.5 (σ -Defining Set). *Let $L \subseteq \mathbb{A}^n$ denote an arbitrary language. A set $S \subseteq [1..n]$ is σ -defining for $\vec{\sigma}(L)$ if S is an inclusion-minimal set of positions that uniquely characterize the σ -vectors of the state-vectors in L , that is, for all $u, v \in L$, we have*

$$\vec{\sigma}(u)[S] = \vec{\sigma}(v)[S] \quad \text{implies} \quad \vec{\sigma}(u) = \vec{\sigma}(v). \quad (9.1.2)$$

As a σ -defining set S is (inclusion-)minimal, there are “witnesses” that testify that each position in S is needed. We formalize this in the following definition.

Definition 9.1.6. *Let S denote a σ -defining set of a language $L \subseteq \mathbb{A}^n$.*

For a position $i \in S$, two σ -vectors $w_{1,i}, w_{0,i} \in \vec{\sigma}(L)$ are witness-vectors for position i if the two vectors differ on S only at position i , with $w_{1,i}[i] = 1$, that is,

- $w_{1,i}[S \setminus i] = w_{0,i}[S \setminus i]$,
- $w_{1,i}[i] = 1$, and

- $w_{0,i}[i] = 0$.

We write $\mathcal{W}_S := \{w_{1,i}, w_{0,i} \mid i \in S\}$ for a set of witness-vectors for $\vec{\sigma}(L)$.

Note that, similar as the σ -defining set S itself, the witness-vectors $\mathcal{W}_S$ do not directly depend on the state-vectors in L , but only on the σ -vectors of L .

Before we focus on the “dual” result which considers weight-defining sets, we first show that a σ -defining set S (as well as witness-vectors that certify that S is indeed a minimal set) can be computed efficiently.

Lemma 9.1.7. *Given a language $L \subseteq \mathbb{A}^n$, we can compute a σ -defining set S for $\vec{\sigma}(L)$, as well as a set of witness-vectors $\mathcal{W}_S$ for S , in time $\mathcal{O}(|L| \cdot n^4)$.*

Proof. Given a language $L \subseteq \mathbb{A}^n$, we first compute the set $\vec{\sigma}(L)$ of all σ -vectors. Then, starting with $S := [1 \dots n]$, we repeatedly iterate over the positions 1 to n ; for each position i , we check if i can be removed from S , that is, whether removing the i th position from each vector in $\vec{\sigma}(L)$ does not decrease the size of $\vec{\sigma}(L)$. If removing the i th position would decrease the size of $\vec{\sigma}(L)$, we also store two witness-vectors that differ only at position i , but not at the remaining positions in S . We stop when no further positions can be removed; we return the resulting set S , as well as the corresponding pairs of witness-vectors.

We can check if a position $i \in S$ can be removed by checking if the (multi-)set

$$\vec{\sigma}(L)_{S,i} := \{v[S \setminus \{i\}] \mid v \in \vec{\sigma}(L)\}$$

(where position i is removed) contains a duplicate element. This we can do by a linear scan over $\vec{\sigma}(L)_{S,i}$ to construct $\vec{\sigma}(L)_{S,i}$, sorting $\vec{\sigma}(L)_i$, and then another linear scan over $\vec{\sigma}(L)_{S,i}$. Observe that if we indeed detect a duplicate, we have also found the required witness.

For the correctness, observe that during the algorithm we maintain that the positions in S uniquely identify the vectors in $\vec{\sigma}(L)$; as we maintain the size of $\vec{\sigma}(L)$, the resulting set does indeed also uniquely identify the vectors in the initial set $\vec{\sigma}(L)$. Further, our algorithm trivially ensures that S is minimal, and hence, the returned set S is indeed σ -defining for $\vec{\sigma}(L)$.

For the running time, we can compute $\vec{\sigma}(L)$ in time $\mathcal{O}(|L| \cdot \log |\vec{\sigma}(L)| \cdot n) = \mathcal{O}(|L| \cdot n^2)$ by iterating over L and computing each σ -vector separately; filtering out duplicates by using an appropriate data structure. Next, observe that we iterate over all positions in S at most n times; in each iteration, we check for at most n positions whether they can be removed from S . The check if we can remove a position from S runs in the time it takes to sort $\vec{\sigma}(L)$, which we can bound by $\mathcal{O}(|\vec{\sigma}(L)| \cdot \log |\vec{\sigma}(L)| \cdot n) = \mathcal{O}(|L| \cdot n^2)$. Hence, in total the algorithm runs in the claimed running time of $\mathcal{O}(|L| \cdot n^4)$, which completes the proof. $\square$

As a next result, we show that those positions that are not included in the σ -defining set S can be considered as “weight-defining”. Phrased differently, the entries of the weight-vector for the positions not in S determine the entries of the weight-vector at all other positions as well. Formally, we prove the following result.

Lemma 9.1.8. *Let $L \subseteq \mathbb{A}^n$ denote a quasi-orthogonal language and let S denote a σ -defining set for $\vec{\sigma}(L)$.*

Then, for any two state-vectors $u, v \in L$ with $\vec{\sigma}(u) = \vec{\sigma}(v)$, the remaining positions $\bar{S} := [1 \dots n] \setminus S$ uniquely characterize the m -weight-vectors of u and v , that is, we have

$$\vec{w}_m(u)[\bar{S}] = \vec{w}_m(v)[\bar{S}] \quad \text{implies} \quad \vec{w}_m(u) = \vec{w}_m(v). \quad (9.1.3)$$

Proof. Let $S \subseteq [1 \dots n]$ denote a σ -defining set for $\vec{\sigma}(L)$ with witness-vectors $\mathcal{W}_S$ (see Definition 9.1.6), and consider the set $\bar{S} := [1 \dots n] \setminus S$. We proceed to show that Equation (9.1.3) is satisfied. To that end, let $u, v \in L$ denote two state-vectors with $\vec{\sigma}(u) = \vec{\sigma}(v)$ and $\vec{w}_m(u)[\bar{S}] = \vec{w}_m(v)[\bar{S}]$. We need to argue that $\vec{w}_m(u) = \vec{w}_m(v)$, and, in particular, that $\vec{w}_m(u)[S] = \vec{w}_m(v)[S]$. Hence, we proceed to show that, for every $i \in S$, we have $\vec{w}_m(u)[i] = \vec{w}_m(v)[i]$.

Now, fix a position $i \in S$ and fix two corresponding witness-vectors $w_{1,i}, w_{0,i} \in \mathcal{W}_S$. We proceed by showing two immediate equalities.

Claim 9.1.8.1. *The vectors $u, v, w_{1,i}$, and $w_{0,i}$ satisfy*

$$(\vec{w}_m(u) - \vec{w}_m(v)) \cdot (w_{1,i} - w_{0,i}) \equiv_m 0.$$

Proof of Claim. By Lemma 9.1.4, we have that

$$\begin{aligned} (\vec{w}_m(u) - \vec{w}_m(v)) \cdot (w_{1,i} - w_{0,i}) &\equiv_m (\vec{w}_m(u) - \vec{w}_m(v)) \cdot w_{1,i} - (\vec{w}_m(u) - \vec{w}_m(v)) \cdot w_{0,i} \\ &\equiv_m 0 - 0 \\ &\equiv_m 0, \end{aligned}$$

which completes the proof. $\triangleleft$

Claim 9.1.8.2. *The vectors $u, v, w_{1,i}$, and $w_{0,i}$ satisfy*

$$(\vec{w}_m(u) - \vec{w}_m(v)) \cdot (w_{1,i} - w_{0,i}) \equiv_m (\vec{w}_m(u)[i] - \vec{w}_m(v)[i]) \cdot (w_{1,i}[i] - w_{0,i}[i]).$$

Proof of Claim. Observe that, by assumption, for every position $j \in \bar{S}$, we have $\vec{w}_m(u)[j] = \vec{w}_m(v)[j]$. Further, by the definitions of i and S , for every position $j \in S \setminus i$, we have $w_{1,i}[j] = w_{0,i}[j]$, which yields the claim. $\triangleleft$

Now, combining Claims 9.1.8.1 and 9.1.8.2 yields

$$\begin{aligned} 0 &\equiv_m (\vec{w}_m(u) - \vec{w}_m(v)) \cdot (w_{1,i} - w_{0,i}) \\ &\equiv_m (\vec{w}_m(u)[i] - \vec{w}_m(v)[i]) \cdot (w_{1,i}[i] - w_{0,i}[i]) \\ &\equiv_m (\vec{w}_m(u)[i] - \vec{w}_m(v)[i]) \cdot (1 - 0) \\ &\equiv_m \vec{w}_m(u)[i] - \vec{w}_m(v)[i]. \end{aligned}$$

In other words, $\vec{w}_m(u)[i] = \vec{w}_m(v)[i]$ for all $i \in S$, which yields Equation (9.1.3), and hence, the claim. $\square$

As a direct consequence of Lemma 9.1.8, we obtain a first upper bound on the size of quasi-orthogonal languages.

Lemma 9.1.9. *Let $L \subseteq \mathbb{A}^n$ denote a quasi-orthogonal language, and let S denote a σ -defining set for $\vec{\sigma}(L)$. Then, we have*

$$|L| \leq (t_{\text{top}} + 1)^{n-|S|} \cdot \sum_{k=0}^{|S|} \binom{|S|}{k} \cdot \left\lceil \frac{s_{\text{top}} + 1}{m} \right\rceil^k \cdot \left\lceil \frac{r_{\text{top}} + 1}{m} \right\rceil^{|S|-k}.$$

Proof. For an integer $k \in [0 \dots |S|]$, we write L_k to denote the set of all state-vectors $x \in L$ such that $\vec{\sigma}(x)[S]$ has a Hamming-weight of exactly k , that is, $\vec{\sigma}(x)[S]$ contains exactly k entries equal to 1.

As L decomposes into the different sets L_k , we obtain $|L| = \sum_{k=0}^{|S|} |L_k|$. Hence, it suffices to show that, for each $k \in [0 \dots |S|]$, we have

$$|L_k| \leq (t_{\text{top}} + 1)^{n-|S|} \cdot \binom{|S|}{k} \cdot \left\lceil \frac{s_{\text{top}} + 1}{m} \right\rceil^k \cdot \left\lceil \frac{r_{\text{top}} + 1}{m} \right\rceil^{|S|-k}. \quad (9.1.4)$$

We proceed with the proof of Inequality (9.1.4). To that end, fix $k \in [0 \dots |S|]$, and observe that a state-vector $x \in L_k$ is uniquely determined by its σ -vector $\vec{\sigma}(x)$ and its weight-vector $\vec{w}(x)$ (where elements are not taken modulo m). Note that since L is quasi-orthogonal, not all pairs of σ -vectors and weight-vectors correspond to a state-vector in L_k . Hence, we write

$$|L_k| \leq \sum_{\vec{s} \in \vec{\sigma}(L_k)} |\{\vec{w}(x) \mid x \in L_k \text{ and } \vec{\sigma}(x) = \vec{s}\}|. \quad (9.1.5)$$

Now, as S is σ -defining for $\vec{\sigma}(L)$ (and hence, for $\vec{\sigma}(L_k) \subseteq \vec{\sigma}(L)$), for each σ -vector for L_k when restricted to the positions S , there is exactly one σ -vector for L_k (on all positions). In particular, the number of different σ -vectors of L_k is equal to the number of different σ -vectors on the positions S . Further, by construction of L_k , all σ -vectors on S have Hamming-weight exactly k . Hence, we obtain

$$|\vec{\sigma}(L_k)| = |\{\vec{\sigma}(x) \mid x \in L_k\}| = |\{\vec{\sigma}(x)[S] \mid x \in L_k\}| \leq \binom{|S|}{k}. \quad (9.1.6)$$

Now, fix a σ -vector $\vec{s} \in \vec{\sigma}(L_k)$, and write $L_{k,\vec{s}} := \{x \in L_k \mid \vec{\sigma}(x) = \vec{s}\}$ for all state-vectors in L_k with σ -vector $\vec{s}$. We show the following bound.

Claim 9.1.9.1. *For every σ -vector $\vec{s} \in \vec{\sigma}(L_k)$, we have*

$$|\vec{w}(L_{k,\vec{s}})| \leq (t_{\text{top}} + 1)^{n-|S|} \cdot \left\lceil \frac{s_{\text{top}} + 1}{m} \right\rceil^k \cdot \left\lceil \frac{r_{\text{top}} + 1}{m} \right\rceil^{|S|-k}.$$

Proof of Claim. We start with an easy observation that allows us to decompose the bound into two parts, one considering the positions from S and one considering the positions from $\bar{S} := [1 \dots n] \setminus S$. For ease of notation, we write u_m for the m -weight-vector corresponding to a weight-vector u . Based on the “weight-defining” property from Lemma 9.1.8, we obtain

$$\begin{aligned} |\vec{w}(L_{k,\vec{s}})| &\leq \sum_{u \in \{w[\bar{S}] \mid w \in \vec{w}(L_{k,\vec{s}})\}} |\{\vec{w}(x)[S] \mid x \in L_{k,\vec{s}} \wedge \vec{w}(x)[\bar{S}] = u\}| \\ &\leq \sum_{u \in \{w[\bar{S}] \mid w \in \vec{w}(L_{k,\vec{s}})\}} |\{\vec{w}(x)[S] \mid x \in L_{k,\vec{s}} \wedge \vec{w}_m(x)[\bar{S}] = u_m\}|. \end{aligned} \quad (9.1.7)$$

We first restrict ourselves to the positions from $\bar{S}$ and obtain a naive bound on the number of weight-vectors (restricted to these positions). As there are no restriction for these positions, we directly get

$$|\{w[\bar{S}] \mid w \in \vec{w}(L_{k,\vec{s}})\}| \leq (t_{\text{top}} + 1)^{|\bar{S}|} = (t_{\text{top}} + 1)^{n-|S|}. \quad (9.1.8)$$

Thus, for the remainder of this proof, we focus on the positions from S and prove a bound for $|\{\vec{w}(x)[S] \mid x \in L_{k,\vec{s}} \wedge \vec{w}_m(x)[\bar{S}] = u_m\}|$.

By Lemma 9.1.8, restricting a m -weight-vector from $L_{k,\vec{s}}$ to the positions from $\bar{S}$, uniquely determines the m -weight-vector for $L_{k,\vec{s}}$ when considering all positions. Formally, for a fixed m -weight-vector $u \in [0 \dots m]^{|\bar{S}|}$, we have

$$|\{\vec{w}_m(x) \mid x \in L_{k,\vec{s}} \wedge \vec{w}_m(x)[\bar{S}] = u\}| = 1. \quad (9.1.9)$$

Hence, we know that the m -weight-vectors are unique, and it suffices to bound the number of different weight-vectors (on S) that result in the same m -weight-vector (on S).

By construction, on the positions S , the state-vector x contains exactly k σ -states. For each such σ -position, there are at most $\lceil (s_{\text{top}} + 1)/m \rceil$ different characters having the same m -weight-vector. For each of the remaining $|S| - k$ ρ -positions, there are at most $\lceil (r_{\text{top}} + 1)/m \rceil$ different characters having the same m -weight-vector. This yields

$$|\{\vec{w}(x)[S] \mid x \in L_{k,\vec{s}} \wedge \vec{w}_m(x)[\bar{S}] = u_m\}| \leq \left\lceil \frac{s_{\text{top}} + 1}{m} \right\rceil^k \cdot \left\lceil \frac{r_{\text{top}} + 1}{m} \right\rceil^{|S|-k}. \quad (9.1.10)$$

As this bound does not depend on the actual choice of $\vec{s}$ but only on the number of 1-entries, we can multiply the bounds from Equations (9.1.8) and (9.1.10) to conclude the proof by the observation from Equation (9.1.7). $\triangleleft$

As the bound from Claim 9.1.9.1 is independent of the actual choice for $\vec{s}$ but only depends on the number of σ -position on S , we obtain the claimed Inequality (9.1.4) from Equations (9.1.5) and (9.1.6) together with Claim 9.1.9.1:

$$\begin{aligned} |L_k| &\leq \sum_{\vec{s} \in \vec{\sigma}(L_k)} |\vec{w}(L_{k,\vec{s}})| \\ &\leq (t_{\text{top}} + 1)^{n-|S|} \cdot \binom{|S|}{k} \cdot \left\lceil \frac{s_{\text{top}} + 1}{m} \right\rceil^k \cdot \left\lceil \frac{r_{\text{top}} + 1}{m} \right\rceil^{|S|-k}. \end{aligned}$$

Overall, this yields the desired bound. $\square$

In a final step before proving Main Lemma 9.1.11, we tidy up the unwieldy upper bound from Lemma 9.1.9.

Lemma 9.1.10. *For any non-negative integers n and $a \in [0 \dots n]$, we have*

$$\begin{aligned} (t_{\text{top}} + 1)^{n-a} \cdot \sum_{k=0}^a \binom{a}{k} \cdot \left\lceil \frac{s_{\text{top}} + 1}{m} \right\rceil^k \cdot \left\lceil \frac{r_{\text{top}} + 1}{m} \right\rceil^{a-k} \\ \leq \begin{cases} (t_{\text{top}} + 2)^n & \text{if } m = 2 \text{ and } t_{\text{top}} = s_{\text{top}} = r_{\text{top}} \text{ is even,} \\ (t_{\text{top}} + 1)^n & \text{otherwise.} \end{cases} \end{aligned}$$

Proof. In a first step, we apply $t_{\text{top}} = \max\{s_{\text{top}}, r_{\text{top}}\}$ and the Binomial Theorem which yields

$$\begin{aligned} & (t_{\text{top}} + 1)^{n-a} \cdot \sum_{k=0}^a \binom{a}{k} \cdot \left\lceil \frac{s_{\text{top}} + 1}{m} \right\rceil^k \cdot \left\lceil \frac{r_{\text{top}} + 1}{m} \right\rceil^{a-k} \\ & \leq (t_{\text{top}} + 1)^{n-a} \cdot \sum_{k=0}^a \binom{a}{k} \cdot \left\lceil \frac{t_{\text{top}} + 1}{m} \right\rceil^k \cdot \left\lceil \frac{t_{\text{top}} + 1}{m} \right\rceil^{a-k} \\ & = (t_{\text{top}} + 1)^{n-a} \cdot \left(2 \cdot \left\lceil \frac{t_{\text{top}} + 1}{m} \right\rceil \right)^a. \end{aligned}$$

In a next step, we investigate the term $\lceil (t_{\text{top}} + 1)/m \rceil$.

First, if $m \geq 3$ or if $m = 2$ and t_{top} is odd, we have

$$2 \cdot \left\lceil \frac{t_{\text{top}} + 1}{m} \right\rceil \leq t_{\text{top}} + 1.$$

Hence, in these cases, we directly obtain

$$(t_{\text{top}} + 1)^{n-a} \cdot \left(2 \cdot \left\lceil \frac{t_{\text{top}} + 1}{m} \right\rceil \right)^a \leq (t_{\text{top}} + 1)^n.$$

Next, if $m = 2$ and $t_{\text{top}} = s_{\text{top}} = r_{\text{top}}$ is even, we have

$$2 \cdot \left\lceil \frac{t_{\text{top}} + 1}{m} \right\rceil = t_{\text{top}} + 2.$$

Hence, in this case, we directly obtain

$$(t_{\text{top}} + 1)^{n-a} \cdot \left(2 \cdot \left\lceil \frac{t_{\text{top}} + 1}{m} \right\rceil \right)^a \leq (t_{\text{top}} + 2)^n.$$

Finally, if $m = 2$, t_{top} is even, and $\min\{r_{\text{top}}, s_{\text{top}}\} < t_{\text{top}}$, we need a more careful analysis. We restart from the initial term, apply the Binomial Theorem and obtain

$$\begin{aligned} & (t_{\text{top}} + 1)^{n-a} \cdot \sum_{k=0}^a \binom{a}{k} \cdot \left\lceil \frac{s_{\text{top}} + 1}{m} \right\rceil^k \cdot \left\lceil \frac{r_{\text{top}} + 1}{m} \right\rceil^{a-k} \\ & = (t_{\text{top}} + 1)^{n-a} \cdot \left(\left\lceil \frac{s_{\text{top}} + 1}{m} \right\rceil + \left\lceil \frac{r_{\text{top}} + 1}{m} \right\rceil \right)^a \\ & \leq (t_{\text{top}} + 1)^{n-a} \cdot \left(\left\lceil \frac{t_{\text{top}}}{m} \right\rceil + \left\lceil \frac{t_{\text{top}} + 1}{m} \right\rceil \right)^a \\ & = (t_{\text{top}} + 1)^{n-a} \cdot \left(\frac{t_{\text{top}}}{2} + \frac{t_{\text{top}} + 2}{2} \right)^a \\ & \leq (t_{\text{top}} + 1)^n. \end{aligned}$$

This completes the proof. □

Finally, combining Lemmas 9.1.9 and 9.1.10 directly yields Main Lemma 9.1.11, which we restate here for convenience.

Main Lemma 9.1.11. *Let $L \subseteq \mathbb{A}^n$ denote a quasi-orthogonal language.*

If $m \geq 3$ or t_{top} is odd or $\min\{s_{\text{top}}, r_{\text{top}}\} < t_{\text{top}}$, then $|L| \leq (t_{\text{top}} + 1)^n$.

If $m = 2$, t_{top} is even, and $s_{\text{top}} = r_{\text{top}} = t_{\text{top}}$, then $|L| \leq (t_{\text{top}} + 2)^n$.

Proof. Let $L \subseteq \mathbb{A}^n$ denote a quasi-orthogonal language, and let S denote a σ -defining set for L . By Lemma 9.1.9, we obtain

$$|L| \leq (t_{\text{top}} + 1)^{n-|S|} \cdot \sum_{k=0}^{|S|} \binom{|S|}{k} \cdot \left\lceil \frac{s_{\text{top}} + 1}{m} \right\rceil^k \cdot \left\lceil \frac{r_{\text{top}} + 1}{m} \right\rceil^{|S|-k}.$$

Applying Lemma 9.1.10 yields the claim. $\square$

9.1.1.2 Combinations of Realized Languages

Having understood a single realized language, we turn to *combinations* of realized languages next.

Definition 9.1.12 (Combination of States). *For two state-vectors $x, y \in \mathbb{A}^n$, we define their combination as the state-vector $x \oplus y \in (\mathbb{A} \cup \{\perp\})^n$ obtained via*

$$(x \oplus y)[\ell] := \begin{cases} \sigma_k & \text{if } x[\ell] = \sigma_i \text{ and } y[\ell] = \sigma_j \text{ and } i + j = k \leq s_{\text{top}}, \\ \rho_k & \text{if } x[\ell] = \rho_i \text{ and } y[\ell] = \rho_j \text{ and } i + j = k \leq r_{\text{top}}, \\ \perp & \text{otherwise.} \end{cases}$$

We say that two state-vectors x and y can be joined if, for each position $\ell \in [1..n]$, we have $(x \oplus y)[\ell] \neq \perp$.

For two languages $L_1, L_2 \subseteq \mathbb{A}^n$, we define their combination $L_1 \oplus L_2$ as the set of all combinations of state-vectors that can be joined, that is,

$$L_1 \oplus L_2 := \{x \oplus y \mid x \in L_1 \text{ and } y \in L_2 \text{ such that } x \text{ and } y \text{ can be joined}\}.$$

Observe that, for two state-vectors $x_1 \in L_1$ and $x_2 \in L_2$, their combination $x_1 \oplus x_2$ is in $L_1 \oplus L_2$ if and only if x_1 and x_2 share a common σ -vector and the sum of their weight-vectors does not “overflow”, that is, we have

$$L_1 \oplus L_2 = \{x \oplus y \mid x \in L_1, y \in L_2, \vec{\sigma}(x) = \vec{\sigma}(y), \text{ and } \vec{w}(x) + \vec{w}(y) \leq \text{cap}(\vec{\sigma}(x))\} \quad (9.1.11)$$

where, for ease of notation, we use “ $\leq$ ” component-wise on weight-vectors.

Finally, observe that, for state-vectors $x \in L_1$ and $y \in L_2$ that can be joined, we have

$$\vec{w}(x) + \vec{w}(y) = \vec{w}(x \oplus y). \quad (9.1.12)$$

We use the remainder of this section to show that Lemma 9.1.9 and Main Lemma 9.1.11 easily lift to the combinations of realized languages. Thereby, we show that the number of partial solutions does not significantly increase by combining realized languages. We start with a small collection of useful properties of combinations of realized languages. First, we discuss how σ -vectors behave under combinations of languages.

Lemma 9.1.13. *Let $L_1, L_2 \subseteq \mathbb{A}^n$ denote two arbitrary languages.*

Then, $(L_1 \oplus L_2)$ has the σ -vectors that appear for both L_1 and L_2 , that is,

$$\vec{\sigma}(L_1 \oplus L_2) \subseteq \vec{\sigma}(L_1) \cap \vec{\sigma}(L_2).$$

Proof. The proof follows immediately from Definition 9.1.12. Indeed, no vector of the language $L_1 \oplus L_2$ has a $\perp$ entry, as $(L_1 \oplus L_2)$ contains only combinations of vectors with the same σ -vectors. Hence, the vectors in $L_1 \oplus L_2$ may differ from vectors in L_1 or L_2 only in their weight-vectors, and $\vec{\sigma}(L_1 \oplus L_2)$ has only those σ -vectors that appear in both $\vec{\sigma}(L_1)$ and $\vec{\sigma}(L_2)$. Furthermore, note that due to “overflows” in the weight-vectors, a σ -vector in $\vec{\sigma}(L_1) \cap \vec{\sigma}(L_2)$ might not be in $\vec{\sigma}(L_1 \oplus L_2)$. $\square$

Next, we show that quasi-orthogonality transfers as well to the combination of two quasi-orthogonal languages.

Lemma 9.1.14. *Let $L_1, L_2 \subseteq \mathbb{A}^n$ denote two quasi-orthogonal languages.*

Then, we have that $(L_1 \oplus L_2)$ is quasi-orthogonal.

Proof. Fix two state-vectors $x, y \in L_1 \oplus L_2$. By Definition 9.1.12, this means that there are two state-vectors $x_1, y_1 \in L_1$ and two state-vectors $x_2, y_2 \in L_2$ such that $x = x_1 \oplus x_2$ and $y = y_1 \oplus y_2$. Expanding the definition of $\oplus$ yields

$$\vec{\sigma}(x) \cdot \vec{w}_m(y) = \vec{\sigma}(x) \cdot (\vec{w}_m(y_1) + \vec{w}_m(y_2)) = \vec{\sigma}(x) \cdot \vec{w}_m(y_1) + \vec{\sigma}(x) \cdot \vec{w}_m(y_2).$$

As $\oplus$ does not change the σ -vectors of the state-vectors that can be joined, we obtain

$$\vec{\sigma}(x) \cdot \vec{w}_m(y) = \vec{\sigma}(x_1) \cdot \vec{w}_m(y_1) + \vec{\sigma}(x_2) \cdot \vec{w}_m(y_2).$$

Next, we use the quasi-orthogonality of L_1 and L_2 for (x_1, y_1) and (x_2, y_2) to obtain

$$\vec{\sigma}(x) \cdot \vec{w}_m(y) \equiv_m \vec{\sigma}(y_1) \cdot \vec{w}_m(x_1) + \vec{\sigma}(y_2) \cdot \vec{w}_m(x_2).$$

Again, as $\oplus$ does not change the σ -vectors for state-vectors that can be joined, we obtain

$$\begin{aligned} \vec{\sigma}(x) \cdot \vec{w}_m(y) &\equiv_m \vec{\sigma}(y) \cdot \vec{w}_m(x_1) + \vec{\sigma}(y) \cdot \vec{w}_m(x_2) \\ &= \vec{\sigma}(y) \cdot (\vec{w}_m(x_1) + \vec{w}_m(x_2)) \\ &= \vec{\sigma}(y) \cdot \vec{w}_m(x), \end{aligned}$$

which completes the proof that $L_1 \oplus L_2$ is quasi-orthogonal. $\square$

Now, we directly obtain that Lemma 9.1.9 lifts to the combination of languages.

Corollary 9.1.15. *Let $L_1, L_2 \subseteq \mathbb{A}^n$ denote two quasi-orthogonal languages. Further, let S denote a σ -defining set for $\vec{\sigma}(L_1 \oplus L_2)$. Then, we have*

$$|L_1 \oplus L_2| \leq (t_{\text{top}} + 1)^{n-|S|} \cdot \sum_{k=0}^{|S|} \binom{|S|}{k} \cdot \left\lceil \frac{s_{\text{top}} + 1}{m} \right\rceil^k \cdot \left\lceil \frac{r_{\text{top}} + 1}{m} \right\rceil^{|S|-k}.$$

Proof. By Lemma 9.1.14, we can use Lemma 9.1.9 on the language $L_1 \oplus L_2$. $\square$

With our improved bounds at hand, we can move toward the improved dynamic programming algorithm. We start with the faster join operations as they need a special treatment.

9.1.2 Fast Join Operations

Recall that the bound in Main Lemma 9.1.11 yields an upper bound on the number of partial solutions for a graph G and a subset U of its vertices which we treat as portal vertices. Recall further that, in the end, our goal is to state a dynamic programming based algorithm on a given tree decomposition. Hence, we need to be able to efficiently compute possible partial solutions for a graph given the already computed partial solutions for some of its subgraphs. Our next goal is precisely this computation. In particular, we show how to use convolution techniques to compute the combination of realized languages. Formally, we prove Main Lemma 9.1.23.

Main Lemma 9.1.23. *Let $L_1, L_2 \subseteq \mathbb{A}^n$ denote two quasi-orthogonal languages. Given two functions $f_1: L_1 \rightarrow [1..U]$ and $f_2: L_2 \rightarrow [1..U]$ for some positive integer U as table representations.*

We can compute the language $L_1 \oplus L_2$ and the function $f: L_1 \oplus L_2 \rightarrow [1..U]$ where

$$f(x) = \sum_{\substack{x_1 \in L_1, x_2 \in L_2 \\ s.t. x_1 + x_2 = x}} f_1(x_1) \cdot f_2(x_2) \quad (9.1.15)$$

in time $(\log U + n + 2^{t_{\text{top}}})^{\mathcal{O}(1)} \cdot (t_{\text{top}} + 2)^n$ if $m = 2$ and $t_{\text{top}} = s_{\text{top}} = r_{\text{top}}$ is even, and in time $(\log U + n + 2^{t_{\text{top}}})^{\mathcal{O}(1)} \cdot (t_{\text{top}} + 1)^n$ otherwise.

First recall that two state-vectors $x_1 \in L_1$ and $x_2 \in L_2$ decompose into a σ -vector and a weight-vector each. Further, recall from the definition of a combination of two languages in Definition 9.1.12 that $x_1 \oplus x_2$ is in $L_1 \oplus L_2$ if and only if x_1 and x_2 share a common σ -vector and the sum of their weight-vectors does not “overflow”. This observation yields the following proof strategy.

1. For each different σ -vector $\vec{s} \in \vec{\sigma}(L_1) \cap \vec{\sigma}(L_2)$, we compute all possible sums of the weight-vectors of those state-vectors with σ -vector $\vec{s}$.
2. Afterward, we filter out the resulting vectors where an overflow occurs.

To implement this strategy, we intend to make use of the tools we introduced in Section 3.1 which are based on the techniques by van Rooij [130].

However, the naive approach, which would directly apply the convolution techniques to the weight-vectors, is not fast enough for our purposes: A single convolution takes time $(t_{\text{top}} + 1)^n \cdot \text{poly}(n)$ already, and so, using such a convolution for each of the up to $2^{|S|}$ different σ -vectors is far too slow.

Instead of computing the convolution directly, we remind ourselves of the results about σ -defining vectors from Lemma 9.1.8. That is, once we fix a σ -vector $\vec{s}$, the number of weight-vectors is much smaller than $(t_{\text{top}} + 1)^n$. The main reason for this improved bound is that we can infer information about certain positions from other positions of the weight-vectors.

We exploit this property to *compress* the weight-vectors to a smaller representation. Then, we compute the convolution of these compressed vectors and finally recover the original vectors from the compression.

Although, Lemma 9.1.8 allows us to recover some positions, we only remember the information modulo m . Moreover, the application of Lemma 9.1.8 requires that the vectors already agree on some positions. As this is not always the case for the following applications, we define some auxiliary vector which allows for a “normalization” of the vectors.

Definition 9.1.16 (Remainder-Vector). Let $L \subseteq \mathbb{A}^n$ denote a (non-empty) quasi-orthogonal language, let S denote a σ -defining set for $\vec{\sigma}(L)$, let $\mathcal{W}_S \subseteq \vec{\sigma}(L)$ denote a corresponding set of witness-vectors, and set $\bar{S} := [1..n] \setminus S$.

For two vectors $u, o \in [0..t_{\text{top}}]^n$ (representing two weight-vectors) and a position $\ell \in S$, we define the remainder $\text{rem}_{\mathcal{W}_S}(u, o)$ at position ℓ as

$$\text{rem}_{\mathcal{W}_S}(u, o)[\ell] := \sum_{i \in \bar{S}} (u[i] - o[i]) \cdot (w_{1,\ell}[i] - w_{0,\ell}[i]).$$

If u is a longer $(n + d)$ -dimensional vector for some $d \geq 1$, we slightly abuse notation and write $\text{rem}_{\mathcal{W}_S}(u, o)$ to denote the vector $\text{rem}_{\mathcal{W}_S}(u[1..n], o)$.

We start with some observation about the remainder-vector.

Lemma 9.1.17. Let $L \subseteq \mathbb{A}^n$ denote a (non-empty) quasi-orthogonal language, let S denote a σ -defining set for $\vec{\sigma}(L)$, and let $\mathcal{W}_S \subseteq \vec{\sigma}(L)$ denote a corresponding set of witness-vectors.

For two weight-vectors u, o that correspond to two state-vectors in L with a common σ -vector, that is, for all $u, o \in \{\vec{w}(x) \mid x \in L \text{ and } \vec{\sigma}(x) = \vec{s}\}$, we have, for all positions $\ell \in S$, that

$$u[\ell] - o[\ell] + \text{rem}_{\mathcal{W}_S}(u, o)[\ell] \equiv_m 0.$$

Proof. By Definition 9.1.6, the witness-vectors $w_{1,\ell}$ and $w_{0,\ell}$ agree on all positions from $S \setminus \{i\}$. Hence, together with Lemma 9.1.4, we directly get

$$\begin{aligned} u[\ell] - o[\ell] + \text{rem}_{\mathcal{W}_S}(u, o)[\ell] &= u[\ell] - o[\ell] + \sum_{i \in [1..n] \setminus S} (u[i] - o[i]) \cdot (w_{1,\ell}[i] - w_{0,\ell}[i]) \\ &= (u - o) \cdot (w_{1,\ell} - w_{0,\ell}) \\ &= (u - o) \cdot w_{1,\ell} - (u - o) \cdot w_{0,\ell} \\ &\equiv_m 0 - 0 \equiv_m 0 \end{aligned}$$

and this concludes the proof. $\square$

Now the idea is as follows. We pick an arbitrary weight-vector $o \in \{\vec{w}(x) \mid x \in L \wedge \vec{\sigma}(x) = \vec{s}\}$ to act as an “origin”. Then, we can shift all vectors in $\{\vec{w}(x) \mid x \in L \text{ and } \vec{\sigma}(x) = \vec{s}\}$ by this origin-vector and additionally based on the remainder-vector so that their coordinates on S become divisible by m . Then, we can exploit this to compress the coordinates on S . In other words, the reduced flexibility due to fixing the σ -vector $\vec{s}$ translates to a reduced flexibility in the choice of coordinates for the positions that define the possible weight-vectors.

Definition 9.1.18 (Compression). Let $L \subseteq \mathbb{A}^n$ denote a (non-empty) quasi-orthogonal language, let S denote a σ -defining set for $\vec{\sigma}(L)$, and let $\mathcal{W}_S \subseteq \vec{\sigma}(L)$ denote a corresponding set of witness-vectors. Fix a σ -vector $\vec{s} \in \vec{\sigma}(L)$.

We write $\mathcal{C}_{S,\vec{s}} := \times_{\ell \in [1..n+1]} N_\ell$ for the space containing all σ -compressed vectors for a σ -defining set S and a fixed σ -vector $\vec{s}$ where,

- for all $\ell \in [1..n] \setminus S$, we define $N_\ell := [0..t_{\text{top}}]$,
- for all $\ell \in S$, we define $N_\ell := \mathbb{Z}_{\bar{m}_\ell}$ with $\bar{m}_\ell := \lceil (\text{cap}(\vec{s})[\ell] + 1)/m \rceil$, and

- we define $N_{\ell+1} := [0 .. n \cdot t_{\text{top}}]$.

For a weight-vector $z \in \{\vec{w}(x) \mid x \in L \wedge \vec{\sigma}(x) = \vec{s}\}$, and an origin-vector $o \in \{\vec{w}(x) \mid x \in L \wedge \vec{\sigma}(x) = \vec{s}\}$, we define the σ -compression of z with origin o and type $\vec{s}$ as the following $(n+1)$ -dimensional vector $z \downarrow_o \in \mathcal{C}_{S, \vec{s}}$:

$$\begin{aligned} z \downarrow_o[\ell] &:= z[\ell], & \ell \in [1 .. n] \setminus S, \\ z \downarrow_o[\ell] &:= \frac{z[\ell] - o[\ell] + \text{rem}_{\mathcal{W}_S}(z, o)[\ell]}{m} \bmod \left\lceil \frac{\text{cap}(\vec{s})[\ell] + 1}{m} \right\rceil, & \ell \in S, \\ z \downarrow_o[n+1] &:= \sum_{\ell \in S} z[\ell]. \end{aligned}$$

We observe that, by Lemma 9.1.17, the definition of a compression is actually well-defined since the positions from S are indeed integers.

Moreover, by the definition of $\mathcal{C}_{S, \vec{s}}$, we define the addition of two compressed weight-vectors component-wise; we use addition modulo $\overline{m}_\ell$ for a dimension $\ell \in S$, and use the standard addition (without modulus) for all other dimension.

From the definition of the compression, we directly obtain a size bound for the space of all compressed vectors.

Lemma 9.1.19. *Let $L \subseteq \mathbb{A}^n$ denote a (non-empty) quasi-orthogonal language, let S denote a σ -defining set for $\vec{\sigma}(L)$, Let $\vec{s} \in \vec{\sigma}(L)$ denote a σ -vector where k positions from S are 1-positions and the remaining $|S| - k$ positions are 0-positions. Then, we have*

$$|\mathcal{C}_{S, \vec{s}}| \leq (t_{\text{top}} + 1)^{n-|S|} \cdot \left\lceil \frac{s_{\text{top}} + 1}{m} \right\rceil^k \cdot \left\lceil \frac{r_{\text{top}} + 1}{m} \right\rceil^{|S|-k} \cdot (n \cdot t_{\text{top}} + 1).$$

Proof. The bound follows directly from the definition of the set $\mathcal{C}_{S, \vec{s}}$ and the properties of the σ -vector $\vec{s}$. $\square$

Later we use the convolution procedure from Main Theorem I.1 on the σ -compressed vectors and directly get a significant speed-up over the direct application to the weight-vectors (whose domain has a size of $(t_{\text{top}} + 1)^n$).

Now we have everything ready to prove that the σ -compression preserves the combination of two state-vectors or rather weight-vectors.

Lemma 9.1.20. *Let $L_1, L_2 \subseteq \mathbb{A}^n$ denote two (non-empty) quasi-orthogonal languages.*

Further, fix a σ -vector $\vec{s} \in \vec{\sigma}(L_1) \cap \vec{\sigma}(L_2)$, as well as two weight-vectors

$$o_1 \in \{\vec{w}(x) \mid x \in L_1 \wedge \vec{\sigma}(x) = \vec{s}\} \quad \text{and} \quad o_2 \in \{\vec{w}(x) \mid x \in L_2 \wedge \vec{\sigma}(x) = \vec{s}\}.$$

For any two state-vectors $u_1 \in L_1$ and $u_2 \in L_2$ with $\vec{\sigma}(u_1) = \vec{\sigma}(u_2) = \vec{s}$, we have that u_1 and u_2 can be joined if and only if there is a state-vector $z \in L_1 \oplus L_2$ with

$$\vec{w}(u_1) \downarrow_{o_1} + \vec{w}(u_2) \downarrow_{o_2} = \vec{w}(z) \downarrow_{o_1+o_2}. \quad (9.1.13)$$

If z exists, we have $z = u_1 \oplus u_2$.

Proof. We denote by S a σ -defining set for $\vec{\sigma}(L_1) \cap \vec{\sigma}(L_2)$ and denote by $\mathcal{W}_S \subseteq \vec{\sigma}(L_1) \cap \vec{\sigma}(L_2)$ a corresponding set of witness-vectors. Moreover, we set $\bar{S} := [1 \dots n] \setminus S$ to simplify notation.

We first show the forward direction of the equivalence.

Claim 9.1.20.1. *If two state-vectors $u_1 \in L_1$ and $u_2 \in L_2$ with $\vec{\sigma}(u_1) = \vec{\sigma}(u_2) = \vec{s}$ can be joined, then $\vec{w}(u_1) \downarrow_{o_1} + \vec{w}(u_2) \downarrow_{o_2} = \vec{w}(z) \downarrow_{o_1+o_2}$ and $z = u_1 \oplus u_2$.*

Proof of Claim. Fix two state-vectors $u_1 \in L_1, u_2 \in L_2$ with $\vec{\sigma}(u_1) = \vec{\sigma}(u_2) = \vec{s}$, and suppose that they can be joined. In particular, for each position $\ell \in [1 \dots n]$, we know that

$$\vec{w}(u_1)[\ell] + \vec{w}(u_2)[\ell] = \vec{w}(u_1 \oplus u_2)[\ell] \leq \text{cap}(\vec{s})[\ell] \leq t_{\text{top}}.$$

Hence, we obtain $\vec{w}(u_1) + \vec{w}(u_2) = \vec{w}(u_1 \oplus u_2)$.

Now, for all positions $\ell \in \bar{S} \cup \{n+1\}$, the claim follows immediately as the entries remain unchanged due to the definition of the compression from Definition 9.1.18. Hence, it remains to consider positions from S .

As the entries of the remainder-vector only depend on the positions in $\bar{S}$ (which are not changed by the compression), we first observe that, for a position $\ell \in S$, we have

$$\begin{aligned} \text{rem}_{\mathcal{W}_S}(\vec{w}(u_1), o_1)[\ell] + \text{rem}_{\mathcal{W}_S}(\vec{w}(u_2), o_2)[\ell] &= \text{rem}_{\mathcal{W}_S}(\vec{w}(u_1) + \vec{w}(u_2), o_1 + o_2)[\ell] \\ &= \text{rem}_{\mathcal{W}_S}(\vec{w}(u_1 \oplus u_2), o_1 + o_2)[\ell]. \end{aligned}$$

Now, we obtain

$$\begin{aligned} &\vec{w}(u_1 \oplus u_2)[\ell] - (o_1 + o_2)[\ell] + \text{rem}_{\mathcal{W}_S}(\vec{w}(u_1 \oplus u_2), o_1 + o_2)[\ell] \\ &\equiv_m \left(\vec{w}(u_1)[\ell] - o_1[\ell] + \text{rem}_{\mathcal{W}_S}(\vec{w}(u_1), o_1)[\ell] \right) \\ &\quad + \left(\vec{w}(u_2)[\ell] - o_2[\ell] + \text{rem}_{\mathcal{W}_S}(\vec{w}(u_2), o_2)[\ell] \right) \\ &= 0, \end{aligned}$$

which implies that the compression of $u_1 \oplus u_2$ is actually well-defined. As the addition for the dimension $\ell \in S$ is computed modulo $\lceil (\text{cap}(\vec{s})[\ell] + 1)/m \rceil$, this directly proves the claim. $\blacktriangleleft$

Now we consider the other direction.

Claim 9.1.20.2. *Assume for two state-vectors $u_1 \in L_1, u_2 \in L_2$ with $\vec{\sigma}(u_1) = \vec{\sigma}(u_2) = \vec{s}$, there is a state-vector $z \in L_1 \oplus L_2$ with $\vec{w}(u_1) \downarrow_{o_1} + \vec{w}(u_2) \downarrow_{o_2} = \vec{w}(z) \downarrow_{o_1+o_2}$. Then, the state-vectors u_1 and u_2 can be joined and $z = u_1 \oplus u_2$.*

Proof of Claim. We fix two state-vectors $u_1 \in L_1$ and $u_2 \in L_2$ with $\vec{\sigma}(u_1) = \vec{\sigma}(u_2) = \vec{s}$. Suppose that there is a state-vector $z \in L_1 \oplus L_2$ with $\vec{w}(u_1) \downarrow_{o_1} + \vec{w}(u_2) \downarrow_{o_2} = \vec{w}(z) \downarrow_{o_1+o_2}$.

First, for an index $\ell \in \bar{S}$, we have

$$\begin{aligned} \vec{w}(z)[\ell] &= \vec{w}(z) \downarrow_{o_1+o_2}[\ell] \\ &= \vec{w}(u_1) \downarrow_{o_1}[\ell] + \vec{w}(u_2) \downarrow_{o_2}[\ell] \\ &= \vec{w}(u_1)[\ell] + \vec{w}(u_2)[\ell] \end{aligned}$$

by the definition of the compression. Since $0 \leq \vec{w}(z)[\ell] \leq \text{cap}(\vec{s})[\ell] \leq t_{\text{top}} + 1$, we obtain that the vectors u_1 and u_2 can be joined at position ℓ . Therefore, we have $\vec{w}(u_1 \oplus u_2)[\ell] = \vec{w}(u_1)[\ell] + \vec{w}(u_2)[\ell] = \vec{w}(z)[\ell]$.

Next, we consider the positions in the set S , and we fix an $\ell \in S$. In the following we write $m' := \lceil (\text{cap}(\vec{s})[\ell] + 1)/m \rceil$ to simplify notation. From the definition of the compression we know that

$$\vec{w}(z) \downarrow_{o_1+o_2}[\ell] \equiv_{m'} \frac{\vec{w}(z)[\ell] - (o_1 + o_2)[\ell] + \text{rem}_{\mathcal{W}_S}(\vec{w}(z), o_1 + o_2)[\ell]}{m}$$

which implies that

$$m \cdot \vec{w}(z) \downarrow_{o_1+o_2}[\ell] \equiv_{m \cdot m'} \vec{w}(z)[\ell] - (o_1 + o_2)[\ell] + \text{rem}_{\mathcal{W}_S}(\vec{w}(z), o_1 + o_2)[\ell]$$

and thus, by rearranging the terms, we get

$$\vec{w}(z)[\ell] \equiv_{m \cdot m'} m \cdot \vec{w}(z) \downarrow_{o_1+o_2}[\ell] + (o_1 + o_2)[\ell] - \text{rem}_{\mathcal{W}_S}(\vec{w}(z), o_1 + o_2)[\ell].$$

This naturally transfers to the analogous results for u_1 and u_2 .

With this observation and the fact that the values of the remainder-vector depend on the positions from $\bar{S}$ only (which we already considered in the previous step), we get

$$\begin{aligned} & \vec{w}(z)[\ell] \\ & \equiv_{m \cdot m'} m \cdot \vec{w}(z) \downarrow_{o_1+o_2}[\ell] + (o_1 + o_2)[\ell] - \text{rem}_{\mathcal{W}_S}(\vec{w}(z), o_1 + o_2)[\ell] \\ & \equiv_{m \cdot m'} m \cdot (\vec{w}(u_1) \downarrow_{o_1}[\ell] + \vec{w}(u_2) \downarrow_{o_2}[\ell]) + (o_1[\ell] + o_2[\ell]) - \text{rem}_{\mathcal{W}_S}(\vec{w}(z), o_1 + o_2)[\ell] \\ & \equiv_{m \cdot m'} (m \cdot \vec{w}(u_1) \downarrow_{o_1}[\ell] + o_1[\ell] - \text{rem}_{\mathcal{W}_S}(\vec{w}(u_1), o_1)[\ell]) \\ & \quad + (m \cdot \vec{w}(u_2) \downarrow_{o_2}[\ell] + o_2[\ell] - \text{rem}_{\mathcal{W}_S}(\vec{w}(u_2), o_2)[\ell]) \\ & \equiv_{m \cdot m'} \vec{w}(u_1)[\ell] + \vec{w}(u_2)[\ell]. \end{aligned}$$

In combination with $0 \leq \vec{w}(z)[\ell] \leq \text{cap}(\vec{s})[\ell] \leq m \cdot m'$ and $0 \leq \vec{w}(u_1)[\ell] + \vec{w}(u_2)[\ell]$, we obtain

$$\vec{w}(z)[\ell] \leq \vec{w}(u_1)[\ell] + \vec{w}(u_2)[\ell]. \quad (9.1.14)$$

By the definition of the check sum in the $n + 1$ th dimension of the compression, we obtain

$$\sum_{i \in S} \vec{w}(z)[i] = \sum_{i \in S} \vec{w}(u_1)[i] + \sum_{i \in S} \vec{w}(u_2)[i].$$

Hence, in combination with Equation (9.1.14), we obtain the desired

$$\vec{w}(u_1)[\ell] + \vec{w}(u_2)[\ell] = \vec{w}(z)[\ell] \leq \text{cap}(\vec{s})[\ell].$$

Therefore, we conclude that $\vec{w}(u_1 \oplus u_2)[\ell] = \vec{w}(u_1)[\ell] + \vec{w}(u_2)[\ell] = \vec{w}(z)[\ell]$.

Overall, we obtain that u_1 and u_2 can be joined and that $z = u_1 \oplus u_2$, which completes the proof. $\triangleleft$

From Claims 9.1.20.1 and 9.1.20.2, we conclude the proof that the compression preserves the combination of state-vectors. $\square$

Based on the previous results, we can combine two languages efficiently by using the appropriate convolution technique. However, we also need to recover the original vectors from their compression. For this we first define the decompression and then show that this actually reverses the compression.

Definition 9.1.21 (Decompression). *For a fixed origin-vector o , we define the mapping $\star\uparrow_o$ to recover a weight-vector $z\uparrow_o$ from a σ -compression z :*

Consider a vector $z \in \mathcal{C}_{S,\bar{S}}$. We define

$$\begin{aligned} z\uparrow_o[\ell] &:= z[\ell], & \ell \in \bar{S}, \\ z\uparrow_o[\ell] &:= (m \cdot z[\ell] + o[\ell] - \text{rem}_{\mathcal{W}_S}(z, o)[\ell]) \mod m \cdot \left\lceil \frac{\text{cap}(\vec{s})[\ell] + 1}{m} \right\rceil \\ &= (m \cdot z[\ell] + o[\ell] \\ &\quad - \sum_{i \in \bar{S}} (z[i] - o[i]) \cdot (w_{1,\ell}[i] - w_{0,\ell}[i])) \mod m \cdot \left\lceil \frac{\text{cap}(\vec{s})[\ell] + 1}{m} \right\rceil, \quad \ell \in S. \end{aligned}$$

Now we can formally prove the remaining properties about the compression of weight-vectors and their decompression.

Lemma 9.1.22. *For all fixed origin-vectors o , the following holds:*

1. *The mapping $\star\downarrow_o$ is injective.*
2. *For all weight-vectors u , we have that $u = (u\downarrow_o)\uparrow_o$.*

Further, for elements $x \in \mathcal{C}_{S,\bar{S}}$ that cannot be obtained from a σ -compression, we have that

- $\sum_{i \in S} z\uparrow_o \neq z[n+1]$, or
- $x[\ell] > \text{cap}(\vec{s})[\ell]$ for some $\ell \in [1..n]$.

Hence, given a subset of $\mathcal{C}_{S,\bar{S}}$, we can decide in time $n^{\mathcal{O}(1)}$ for each vector whether it is a σ -compressed weight-vector.

Proof. Observe that the second set of results, directly follows from the definition of the compression as stated in Definition 9.1.18. Hence, we consider the first set of results in the following.

1. Assume we are given two vectors u and v that have the same compression. As the compression function does not change the values on the positions from $\bar{S}$, the weight-vectors u and v must agree on these coordinates. Hence, their remainder is identical. This already implies that their values on the positions from S must also be identical as both vectors are shifted by the origin-vector and their remainder-vector only.
2. This follows by combining the definition of the decompression with the definition of the compression. Additionally, we exploit that the remainder-vector is neither changed by the compression nor by the decompression. Moreover, we use that $\text{cap}(\vec{s})[\ell] \leq m \cdot \lceil (\text{cap}(\vec{s})[\ell] + 1)/m \rceil$ for the proof. $\square$

To formally state our main result about the combination of two languages, we reuse Main Theorem I.1. As we do not require the full generality of this result, we use Lemma 3.1.8 which is a corollary of Main Theorem I.1 for $\tau = 0$.

Lemma 3.1.8. *Consider three domains of vectors $\mathbb{S}$, $\mathbb{A}$, and $\mathbb{L}$*

- *with $\mathbb{S} = \mathbb{Z}_{z_1} \times \cdots \times \mathbb{Z}_{z_\sigma}$ for the first domain,*
- *with $\mathbb{A} = [0 \dots q_1] \times \cdots \times [0 \dots q_\alpha]$ for the second domain, and set $A_+ = \sum_{i=1}^\alpha q_i$, and*
- *with $\mathbb{L} = [0 \dots x_1] \times \cdots \times [0 \dots x_\lambda]$ for the third domain.*

Denote by $f_1, f_2: \mathbb{S} \times \mathbb{A} \times \mathbb{L} \rightarrow \mathbb{N}$ two functions that are given as table representations.

We denote by $f: \mathbb{S} \times \mathbb{A} \times \mathbb{L} \rightarrow \mathbb{N}$ the convolution of f_1 and f_2 that satisfies, for all $s \in \mathbb{S}$, $a \in \mathbb{A}$, and $\ell \in \mathbb{L}$,

$$f(s, a, \ell) = \sum_{s_1+s_2=s} \sum_{a_1+a_2=a} \sum_{\ell_1+\ell_2=\ell} f_1(s_1, a_1, \ell_1) \cdot f_2(s_2, a_2, \ell_2). \quad (3.1.5)$$

If U denotes a non-negative integer such that all images of f_1 , f_2 and f are contained in $[0 \dots U]$, then the function f can be computed in time

$$X \cdot \log X \cdot (\log U + \overline{M})^{\mathcal{O}(1)} + (\log U + \exp(\overline{M}))^{\mathcal{O}(1)}$$

with $X = |\mathbb{S}| \cdot |\mathbb{A}| \cdot (A_+ \cdot |\mathbb{L}|)^{\log 3} \cdot 3^\lambda$ and $\overline{M} = \prod_{m \in M} m$ where $M = \{z_1, \dots, z_\sigma, q_1, \dots, q_\alpha, 3\}$.

Finally, we are ready to prove the main result, that is, the computation of the combination of two languages.

Main Lemma 9.1.23. *Let $L_1, L_2 \subseteq \mathbb{A}^n$ denote two quasi-orthogonal languages. Given two functions $f_1: L_1 \rightarrow [1 \dots U]$ and $f_2: L_2 \rightarrow [1 \dots U]$ for some positive integer U as table representations.*

We can compute the language $L_1 \oplus L_2$ and the function $f: L_1 \oplus L_2 \rightarrow [1 \dots U]$ where

$$f(x) = \sum_{\substack{x_1 \in L_1, x_2 \in L_2 \\ \text{s.t. } x_1+x_2=x}} f_1(x_1) \cdot f_2(x_2) \quad (9.1.15)$$

in time $(\log U + n + 2^{t_{\text{top}}})^{\mathcal{O}(1)} \cdot (t_{\text{top}} + 2)^n$ if $m = 2$ and $t_{\text{top}} = s_{\text{top}} = r_{\text{top}}$ is even, and in time $(\log U + n + 2^{t_{\text{top}}})^{\mathcal{O}(1)} \cdot (t_{\text{top}} + 1)^n$ otherwise.

Proof. We use the following algorithm to compute the combined language and the convolution of the two functions g_1 and g_2 . The basic idea is to compute the convolution of f_1 and f_2 . Then, the preimage of the non-0 values corresponds to the combined language.

Formally, the algorithm proceeds as follows.

1. Given the two languages L_1 and L_2 , we first compute $\vec{\sigma}(L_1) \cap \vec{\sigma}(L_2)$. Then, we remove every state-vector from L_1 and L_2 whose σ -vector is not in $\vec{\sigma}(L_1) \cap \vec{\sigma}(L_2)$.
2. Next, we use Lemma 9.1.7 to compute a σ -defining set S for $\vec{\sigma}(L_1) \cap \vec{\sigma}(L_2)$ as well as a set of witness-vectors $\mathcal{W}_S$.

3. Now, for each σ -vector $\vec{s} \in \vec{\sigma}(L_1) \cap \vec{\sigma}(L_2)$, we repeat the following steps.

a) We compute the two sets

$$L_{1,\vec{s}} := \{u_1 \mid u_1 \in L_1 \text{ and } \vec{\sigma}(u) = \vec{s}\}$$

and

$$L_{2,\vec{s}} := \{u_2 \mid u_2 \in L_2 \text{ and } \vec{\sigma}(u) = \vec{s}\}.$$

b) We pick two arbitrary state-vectors $q_{1,\vec{s}} \in L_{1,\vec{s}}$ and $q_{2,\vec{s}} \in L_{2,\vec{s}}$. Then, we define $o_{1,\vec{s}} := \vec{w}(q_{1,\vec{s}})$ and $o_{2,\vec{s}} := \vec{w}(q_{2,\vec{s}})$ as the two corresponding origin-vectors.

c) We compute two functions $g_{1,\vec{s}}, g_{2,\vec{s}}: \mathcal{C}_{S,\vec{s}} \rightarrow \mathbb{N}$ based on the functions f_1 and f_2 .

We initialize the functions $g_{1,\vec{s}}$ and $g_{2,\vec{s}}$ with 0, that is, for all values $x \in \mathcal{C}_{S,\vec{s}}$, we initially set $g_{1,\vec{s}}(x) = g_{2,\vec{s}}(x) = 0$.

Then, for all $u_1 \in L_{1,\vec{s}}$ and all $u_2 \in L_{2,\vec{s}}$, we define

$$g_{1,\vec{s}}(\vec{w}(u_1) \downarrow_{o_{1,\vec{s}}}) := f_1(u_1) \quad \text{and} \quad g_{2,\vec{s}}(\vec{w}(u_2) \downarrow_{o_{2,\vec{s}}}) := f_2(u_2). \quad (9.1.16)$$

d) We use Lemma 3.1.8 to compute the function $g_{\vec{s}}: \mathcal{C}_{S,\vec{s}} \rightarrow \mathbb{N}$ as the convolution of $g_{1,\vec{s}}$ and $g_{2,\vec{s}}$ where the dimensions in S form the cyclic domain $\mathbb{S}$, the dimensions in $[1 \dots n] \setminus S$ forms the domain $\mathbb{A}$ with standard addition, and the dimension $n+1$ forms the domain $\mathbb{L}$ with large addition.

e) We use Lemma 9.1.22 to compute the set $W_{\vec{s}}$ of all decompressed weight-vectors whose compression has a positive value for $h_{\vec{s}}$:

$$W_{\vec{s}} := \{z \uparrow_{o_1+o_2} \mid z \in \mathcal{C}_{S,\vec{s}}: g_{\vec{s}}(z) > 0 \text{ and } (z \uparrow_{o_1+o_2}) \downarrow_{o_1+o_2} = z\}.$$

Note that the last check ensures that we only consider vectors that correspond to a compressed weight-vector.

f) We iterate over the weight-vectors in $W_{\vec{s}}$ and reconstruct the state-vectors from the weight-vectors and $\vec{s}$ in the straightforward way. Formally, we define

$$L_{\vec{s}} := \{u \in \mathbb{A}^n \mid \vec{\sigma}(u) = \vec{s} \text{ and } \vec{w}(u) \in W_{\vec{s}}\}.$$

We define the function $h_{\vec{s}}: L_{\vec{s}} \rightarrow \mathbb{N}$ as

$$h_{\vec{s}}(u) := g_{\vec{s}}(\vec{w}(u) \downarrow_{o_1+o_2})$$

for all $u \in L_{\vec{s}}$.

4. Finally, we return the union L of all sets $L_{\vec{s}}$ which we computed in the previous step, that is, we set

$$L := \bigcup_{\vec{s} \in \vec{\sigma}(L_1) \cap \vec{\sigma}(L_2)} L_{\vec{s}}.$$

We define the function $h: L \rightarrow [0 \dots \ell]$ as

$$h(x) := h_{\vec{\sigma}(x)}(x)$$

for all $x \in L$.

We first show the correctness of the algorithm and then analyze the running time.

Claim 9.1.23.1. *The algorithm is correct, that is, $L = L_1 \oplus L_2$ and the function h satisfies Equation (9.1.15) as the convolution of f_1 and f_2*

Proof of Claim. We first observe that, by Lemma 9.1.13, for each state-vector $z \in L_1 \oplus L_2$ there must be a state-vector in L_1 and a state-vector in L_2 such that all three state-vectors share the same σ -vector.

Hence, we can indeed compute the state-vectors in $L_1 \oplus L_2$ for each σ -vector separately. To simplify notation, we define $L_{1,2,\vec{s}} := \{x \in L_1 \oplus L_2 \mid \vec{\sigma}(x) = \vec{s}\}$ as the restriction of $L_1 \oplus L_2$ to the state-vectors with σ -vector $\vec{s}$. Based on this, we define $f_{\vec{s}}: L_{1,2,\vec{s}} \rightarrow [0 \dots \ell]$ as the restriction of the function f to vectors from $L_{1,2,\vec{s}}$.

From now on, we fix a σ -vector $\vec{s} \in \vec{\sigma}(L_1) \cap \vec{\sigma}(L_2)$. In the remainder we prove that $L_{1,2,\vec{s}} = L_{\vec{s}}$ and $f_{\vec{s}} = h_{\vec{s}}$. For both results it suffices to focus on the third step of the algorithm. Thus, we refer to the substeps (of step three) just as steps of the algorithm.

Now consider a state-vector $x \in L_{\vec{s}}$. By the fifth and sixth step, there must be some compressed weight-vector $z \in \mathcal{C}_{S,\vec{s}}$ with $g_{\vec{s}}(z) > 0$ and $(z \uparrow_{o_1+o_2}) \downarrow_{o_1+o_2} = z$ such that $z \uparrow_{o_1+o_2} = \vec{w}(x)$. Hence, we immediately get that $\vec{w}(x) \downarrow_{o_1+o_2} = z$. Together with the computation of the convolution from Lemma 3.1.8 in the fourth step, we know that

$$\begin{aligned} h_{\vec{s}}(x) &= g_{\vec{s}}(\vec{w}(x) \downarrow_{o_1+o_2}) \\ &= g_{\vec{s}}(z) \\ &= \sum_{\substack{z_1, z_2 \in \mathcal{C}_{S,\vec{s}}: \\ z_1 + z_2 = z}} g_{1,\vec{s}}(z_1) \cdot g_{2,\vec{s}}(z_2). \end{aligned}$$

Since $g_{\vec{s}}$ and $g_{2,\vec{s}}$ are 0 for all vectors not corresponding to a compressed weight-vector, and since the compression is injective, we get

$$\begin{aligned} h_{\vec{s}}(x) &= \sum_{\substack{z_1, z_2 \in \mathcal{C}_{S,\vec{s}}: \\ \exists u_1 \in L_1: \vec{w}(u_1) \downarrow_{o_1} = z_1, \\ \exists u_2 \in L_2: \vec{w}(u_2) \downarrow_{o_2} = z_2, \\ z_1 + z_2 = z}} g_{1,\vec{s}}(z_1) \cdot g_{2,\vec{s}}(z_2) \\ &= \sum_{\substack{u_1 \in L_1, u_2 \in L_2: \\ \vec{w}(u_1) \downarrow_{o_1} + \vec{w}(u_2) \downarrow_{o_2} = z}} g_{1,\vec{s}}(\vec{w}(u_1) \downarrow_{o_1}) \cdot g_{2,\vec{s}}(\vec{w}(u_2) \downarrow_{o_2}). \end{aligned}$$

With $z = \vec{w}(x) \downarrow_{o_1+o_2}$ and Lemma 9.1.20, we can rewrite the condition in the sum to get

$$\begin{aligned} h_{\vec{s}}(x) &= \sum_{\substack{u_1 \in L_1, u_2 \in L_2: \\ u_1 \oplus u_2 = x}} g_{1,\vec{s}}(\vec{w}(u_1) \downarrow_{o_1}) \cdot g_{2,\vec{s}}(\vec{w}(u_2) \downarrow_{o_2}) \\ &= \sum_{\substack{u_1 \in L_1, u_2 \in L_2: \\ u_1 \oplus u_2 = x}} f_{1,\vec{s}}(u_1) \cdot g_{2,\vec{s}}(u_2) \end{aligned}$$

where the last equality follows from the definition of the functions $g_{1,\vec{s}}$ and $g_{2,\vec{s}}$ in the third step of the algorithm.

Since the previous argument proves that $L_{\vec{s}} \subseteq L_{1,2,\vec{s}}$ and that the functions $f_{\vec{s}}$ and $h_{\vec{s}}$ are equal for all state-vectors in $L_{\vec{s}}$, it remains to show that $L_{1,2,\vec{s}} \subseteq L_{\vec{s}}$.

Consider a state-vector $u \in L_{1,2,\vec{s}}$. Since we have $f_{\vec{s}}(u) > 0$, there must be two state-vectors $u_1 \in L_{1,\vec{s}}$ and $u_2 \in L_{2,\vec{s}}$ such that $u = u_1 \oplus u_2$ and $f_{1,\vec{s}}(u_1), f_{2,\vec{s}}(u_2) > 0$. Lemma 9.1.20 yields that $\vec{w}(u_1) \downarrow_{o_1} + \vec{w}(u_2) \downarrow_{o_2} = \vec{w}(u_1 \oplus u_2) \downarrow_{o_1+o_2}$. By the definition of the compression, we directly get that $\vec{w}(u_1) \downarrow_{o_1}, \vec{w}(u_2) \downarrow_{o_2} \in \mathcal{C}_{S,\vec{s}}$ which implies that

$$\begin{aligned} g_{\vec{s}}(\vec{w}(u) \downarrow_{o_1+o_2}) &= \sum_{\substack{z_1, z_2 \in \mathcal{C}_{S,\vec{s}}: \\ z_1 + z_2 = \vec{w}(u) \downarrow_{o_1+o_2}}} g_{1,\vec{s}}(z_1) \cdot g_{2,\vec{s}}(z_2) \\ &\geq g_{1,\vec{s}}(\vec{w}(u_1) \downarrow_{o_1}) \cdot g_{2,\vec{s}}(\vec{w}(u_2) \downarrow_{o_2}) \\ &= f_{1,\vec{s}}(u_1) \cdot f_{2,\vec{s}}(u_2) > 0. \end{aligned}$$

Therefore, the vector $(\vec{w}(u) \downarrow_{o_1+o_2}) \uparrow_{o_1+o_2} = \vec{w}(u)$ is contained in the set $W_{\vec{s}}$ by Lemma 9.1.22 and the definition of $W_{\vec{s}}$. Hence, by the definition of the set $L_{\vec{s}}$, we conclude that $u \in L_{\vec{s}}$.

This concludes the proof of correctness of the algorithm. $\triangleleft$

Next, we consider the running time of the algorithm.

Claim 9.1.23.2. *The algorithm operates in time $(\log \ell + n + 2^{t_{\text{top}}})^{\mathcal{O}(1)} \cdot (t_{\text{top}} + 1)^n$ if $m \geq 3$ or $s_{\text{top}} \neq r_{\text{top}}$ or t_{top} is odd. Otherwise, the algorithm operates in time $(\log \ell + n + 2^{t_{\text{top}}})^{\mathcal{O}(1)} \cdot (t_{\text{top}} + 2)^n$.*

Proof of Claim. We can compute the sets $\vec{\sigma}(L_1)$ and $\vec{\sigma}(L_2)$ in total time

$$\mathcal{O}((|L_1| + |L_2|) \cdot \log(|\vec{\sigma}(L_1)| + |\vec{\sigma}(L_2)|) \cdot n) = \mathcal{O}(\max\{|L_1|, |L_2|\} \cdot n^2)$$

by iterating over L_1 or L_2 and computing each σ -vector separately; and additionally filtering out duplicates by using an appropriate data structure (note that $|\vec{\sigma}(L_1)|, |\vec{\sigma}(L_2)| \leq 2^n$). Afterward, we can compute $\vec{\sigma}(L_1) \cap \vec{\sigma}(L_2)$ in the same running time by using standard algorithms for merging sets.

Using the algorithm from Lemma 9.1.7, we can compute the σ -defining set S (as well as the corresponding witness-vectors) in time $\mathcal{O}(\max\{|L_1|, |L_2|\} \cdot n^4)$. As S is a σ -defining set for $\vec{\sigma}(L_1) \cap \vec{\sigma}(L_2)$, we have $|\vec{\sigma}(L_1) \cap \vec{\sigma}(L_2)| \leq 2^{|S|}$.

Now, fix a σ -vector $\vec{s} \in \vec{\sigma}(L_1) \cap \vec{\sigma}(L_2)$. The computation of the two sets $L_{1,\vec{s}}$ and $L_{2,\vec{s}}$, and the functions $g_{1,\vec{s}}$ and $g_{2,\vec{s}}$ takes time linear in L and $|\mathcal{C}_{S,\vec{s}}|$. By the definition of $\mathcal{C}_{S,\vec{s}}$, the application of Lemma 3.1.8 to compute the convolution $g_{\vec{s}}$ takes time

$$X \log X \cdot (\log \ell + \overline{H})^{\mathcal{O}(1)} + (\log \ell + \exp(\overline{H}))^{\mathcal{O}(1)}$$

where

$$X = |\mathcal{C}_{S,\vec{s}}| \cdot ((n - |S|) \cdot (t_{\text{top}} + 1))^{\log 3} \cdot (n \cdot t_{\text{top}} + 1)^{\log(3/2)} \cdot 3$$

and $\overline{H} = (t_{\text{top}} + 1) \cdot \lceil (s_{\text{top}} + 1)/m \rceil \cdot \lceil (r_{\text{top}} + 1)/m \rceil \cdot 3$. We immediately get that $\overline{H} \in t_{\text{top}}^{\mathcal{O}(1)}$. When bounding $\log |\mathcal{C}_{S,\vec{s}}|$ by $\mathcal{O}(n \cdot \log t_{\text{top}})$ according to Lemma 9.1.19 and applying some further simplifications, we obtain a running time of

$$|\mathcal{C}_{S,\vec{s}}| \cdot (\log \ell + t_{\text{top}} + n)^{\mathcal{O}(1)} + (\log \ell + 2^{t_{\text{top}}})^{\mathcal{O}(1)}.$$

Finally, computing $W_{\vec{s}}$ and recovering $L_{\vec{s}}$ can be done with a linear pass over $\mathcal{C}_{S,\vec{s}}$ in the same running time. Combining the recovered (disjoint) sets can then be done in time linear in the size of the returned result $L_1 \oplus L_2$.

Before we bound the total running time of the algorithm, we observe that we consider at most $\min\{|L_1|, |L_2|\}$ possible σ -vectors $\vec{s}$. In total, the algorithm runs in time

$$\begin{aligned} & n^{\mathcal{O}(1)} \cdot \max\{|L_1|, |L_2|, |L_1 \oplus L_2|\} + \sum_{\vec{s}} \left(|\mathcal{C}_{S,\vec{s}}| \cdot (\log \ell + t_{\text{top}} + n)^{\mathcal{O}(1)} + (\log \ell + 2^{t_{\text{top}}})^{\mathcal{O}(1)} \right) \\ &= (\log \ell + 2^{t_{\text{top}}} + n)^{\mathcal{O}(1)} \cdot \max\{|L_1|, |L_2|, |L_1 \oplus L_2|\} + (\log \ell + t_{\text{top}} + n)^{\mathcal{O}(1)} \cdot \sum_{\vec{s}} |\mathcal{C}_{S,\vec{s}}|. \end{aligned} \quad (9.1.17)$$

As a last step, we bound the sum $\sum_{\vec{s}} |\mathcal{C}_{S,\vec{s}}|$. To this extent, for a σ -vector $\vec{s} \in \vec{\sigma}(L_1) \cap \vec{\sigma}(L_2)$, we write $k(\vec{s}) := |\{i \in S \mid \vec{s}[i] = 1\}|$ for the number of 1-entries of the vector $\vec{s}$ on positions from S . With this notation we immediately obtain from Lemma 9.1.19 that

$$\begin{aligned} \sum_{\vec{s}} |\mathcal{C}_{S,\vec{s}}| &= \sum_{\vec{s}} (t_{\text{top}} + 1)^{n-|S|} \cdot \left\lceil \frac{s_{\text{top}} + 1}{m} \right\rceil^{k(\vec{s})} \cdot \left\lceil \frac{r_{\text{top}} + 1}{m} \right\rceil^{|S|-k(\vec{s})} \cdot (n \cdot t_{\text{top}} + 1) \\ &\leq (n \cdot t_{\text{top}} + 1) \cdot (t_{\text{top}} + 1)^{n-|S|} \cdot \sum_{k=0}^{|S|} \binom{|S|}{k} \cdot \left\lceil \frac{s_{\text{top}} + 1}{m} \right\rceil^k \cdot \left\lceil \frac{r_{\text{top}} + 1}{m} \right\rceil^{|S|-k}. \end{aligned} \quad (9.1.18)$$

Since L_1 and L_2 are quasi-orthogonal, the set $L_1 \oplus L_2$ is quasi-orthogonal as well. Therefore, by Lemma 9.1.9 and Corollary 9.1.15, the bound from Equation (9.1.18) also bounds the size of the three sets L_1 , L_2 , and $L_1 \oplus L_2$.

Combined with Equation (9.1.17), the algorithm operates in time

$$(\log \ell + n + 2^{t_{\text{top}}})^{\mathcal{O}(1)} \cdot (t_{\text{top}} + 1)^{n-|S|} \cdot \sum_{k=0}^{|S|} \binom{|S|}{k} \cdot \left\lceil \frac{s_{\text{top}} + 1}{m} \right\rceil^k \cdot \left\lceil \frac{r_{\text{top}} + 1}{m} \right\rceil^{|S|-k}$$

and Lemma 9.1.10 yields the tidier upper bound which completes the proof. $\blacktriangleleft$

The proof of Main Lemma 9.1.23 follows from the stated algorithm and Claims 9.1.23.1 and 9.1.23.2. $\square$

Next, we show the final algorithm for (σ, ρ) -DOMSET.

9.1.3 Finalizing the Algorithm

Now we have everything ready to obtain faster algorithms for (σ, ρ) -DOMSET when (σ, ρ) are m -structured, that is, we prove Theorem 9.1.24.

Theorem 9.1.24. *Fix two finite non-empty sets $\sigma, \rho \subseteq \mathbb{N}$ such that (σ, ρ) is m -structured for some $m \geq 2$. Then, there is an algorithm $\mathcal{A}$ that, given a graph G , a nice tree decomposition of G of width w , and an integer s , computes the number of (σ, ρ) -sets of G of size exactly s .*

If $m \geq 3$ or t_{top} is odd or $s_{\text{top}} \neq r_{\text{top}}$, then algorithm $\mathcal{A}$ runs in time

$$(t_{\text{top}} + 1)^w \cdot 2^{(t_{\text{top}})^{\mathcal{O}(1)}} \cdot |G|^{\mathcal{O}(1)}.$$

If $m = 2$, t_{top} is even, and $s_{\text{top}} = r_{\text{top}} = t_{\text{top}}$, then algorithm $\mathcal{A}$ runs in time

$$(t_{\text{top}} + 2)^w \cdot 2^{(t_{\text{top}})^{\mathcal{O}(1)}} \cdot |G|^{\mathcal{O}(1)}.$$

Proof. For ease of notation, let us define $\tau := t_{\text{top}} + 1$ if $m \geq 3$ or if t_{top} is odd or if $\min\{s_{\text{top}}, r_{\text{top}}\} < t_{\text{top}}$, and define $\tau := t_{\text{top}} + 2$ otherwise, i.e., if $m = 2$, t_{top} is even and $s_{\text{top}} = r_{\text{top}} = t_{\text{top}}$.

Let (T, β) denote the nice tree decomposition of the input graph G where we additionally assume introduce edge nodes and every edge is introduced exactly once. Note that, by Fact 2.2.1, we can assume that these requirements are satisfied. This additional step might take time $\mathcal{O}(m \cdot w)$ where m denotes the number of edges of G . By using introduce edge nodes, we can state a simpler algorithm that still has a good running time.

We fill a table $\text{ParSol}[\star, \star]$ with sets of realizable state-vectors and compute, for each table entry, a function defined on this set. For each node $t \in V(T)$, and each size bound $\kappa \in [0 \dots n]$, we compute the table entry $\text{ParSol}[t, \kappa] \subseteq \mathbb{A}^{\beta(t)}$ and a function $f_{t,\kappa}: \text{ParSol}[t, \kappa] \rightarrow [1 \dots 2^n]$ such that $\text{ParSol}[t, \kappa]$ contains all state-vectors that are compatible with $(G_t, \beta(t))$, and the function $f_{t,\kappa}$ maps each realizable state-vector $x \in \text{ParSol}[t, \kappa]$ to the number of witnessing solution sets S_x such that $|S_x \setminus \beta(t)| = \kappa$.

Observe that we use $(G_t, \beta(t))$ as a graph with portals where $\beta(t)$ denotes the set of portal vertices. This allows us to use standard notation for dynamic programming algorithms on tree decompositions.

By the definition of the sets $\text{ParSol}[t, \kappa]$, they satisfy the property from Lemma 9.1.2 and thus, are quasi-orthogonal by Definition 9.1.3. Hence, Main Lemma 9.1.11 yields

$$|\text{ParSol}[t, \kappa]| \leq \tau^{|\beta(t)|} \leq \tau^{w+1}. \quad (9.1.19)$$

We compute the sets $\text{ParSol}[t, \kappa]$ and functions $f_{t,\kappa}$ for all nodes $t \in V(T)$ in a bottom-up fashion. For this we traverse the given tree T of the tree decomposition (T, β) in post-order and hence, can assume that the values for all children of the current node are already computed and directly available.

Depending on the type of the current node t , the algorithm applies one of the following procedures.

Leaf Node: If the current node t is a leaf node, then we have that $\beta(t) = V_t = \emptyset$.

We define $\text{ParSol}[t, 0] := \{\varepsilon\}$ where ε denotes the empty state-vector. Furthermore, we define the function $f_{t,0}: \text{ParSol}[t, 0] \rightarrow [1 \dots 2^n]$ as $f_{t,0}(\varepsilon) = 1$.

Conversely, for all $\kappa \in [1 \dots n]$, we define $\text{ParSol}[t, \kappa] := \emptyset$ with $f_{t,\kappa}$ as the empty function.

Introduce Vertex Node: Next, consider the case that t is an introduce vertex node, and denote by t' the unique child of t . Suppose that $\beta(t) = \beta(t') \cup \{v\}$, that is, $v \in V(G)$ is the vertex introduced at t .

We have that

$$\text{ParSol}[t, \kappa] := \{x \sqcup (v \mapsto \sigma_0), x \sqcup (v \mapsto \rho_0) \in \mathbb{A}^{\beta(t)} \mid x \in \text{ParSol}[t', \kappa]\}$$

and

$$f_{t,\kappa}(x \sqcup (v \mapsto \sigma_0)) = f_{t,\kappa}(x \sqcup (v \mapsto \rho_0)) = f_{t',\kappa}(x).$$

Introduce Edge Node: Next, consider the case that t is an introduce-edge node with child t' and denote by uv the edge introduced. Note that we have $u, v \in \beta(t')$.

We say that a state-vector $x \in \mathbb{A}^{\beta(t)}$ is the uv -extension of a state-vector $y \in \mathbb{A}^{\beta(t')}$, denoted by $x = y^{+uv}$, if the following four conditions are satisfied:

1. $x[w] = y[w]$ for all $w \in \beta(t) \setminus \{u, v\}$, that is, the state-vectors might only change for the vertices u and v ,
2. $\vec{\sigma}(x)[u] = \vec{\sigma}(y)[u]$ and $\vec{\sigma}(x)[v] = \vec{\sigma}(y)[v]$, that is, the vertices u and v do not change their selection state,
3. $\vec{w}(x)[u] = \vec{w}(y)[u] + \llbracket y[v] \in \mathbb{S} \rrbracket$, that is, vector x takes care of the new, possibly selected neighbor v of u , and
4. $\vec{w}(x)[v] = \vec{w}(y)[v] + \llbracket y[u] \in \mathbb{S} \rrbracket$, that is, vector x takes care of the new, possibly selected neighbor u of v .

We say that a uv -extension is *valid* if, for all $w \in \beta(t)$, we have that $x[w] \in \{\rho_i \mid i \in [0 \dots r_{\text{top}}]\} \cup \{\sigma_i \mid i \in [0 \dots s_{\text{top}}]\}$.

We define

$$\text{ParSol}[t, \kappa] := \{y^{+uv} \in \mathbb{A}^{\beta(t)} \mid y^{+uv} \text{ is valid} \wedge y \in \text{ParSol}[t', \kappa]\}$$

and, for all $y^{+uv} \in \text{ParSol}[t, \kappa]$, we define

$$f_{t,\kappa}(y^{+uv}) = f_{t',\kappa}(y).$$

Forget Node: Next, consider the case that t is a forget node, and denote by t' the unique child of t . Suppose that $\beta(t') = \beta(t) \cup \{v\}$, that is, vertex $v \in V(G)$ is the vertex forgotten at t .

We define

$$\begin{aligned} \text{ParSol}[t, \kappa] := & \{x[\beta(t)] \mid x \in \text{ParSol}[t', \kappa - 1] \text{ such that } \exists d \in \sigma : x[v] = \sigma_d\} \\ & \cup \{x[\beta(t)] \mid x \in \text{ParSol}[t', \kappa] \text{ such that } \exists c \in \rho : x[v] = \rho_c\} \end{aligned}$$

and, for all $x \in \text{ParSol}[t, \kappa]$, we define

$$f_{t,\kappa}(x) = \sum_{d \in \sigma} f_{t',\kappa-1}(x \sqcup (v \mapsto \sigma_d)) + \sum_{c \in \rho} f_{t',\kappa}(x \sqcup (v \mapsto \rho_c)).$$

Join Node: Finally, suppose that t is a join node, and let t_1 and t_2 denote the two children of t . Observe that $\beta(t) = \beta(t_1) = \beta(t_2)$ and that the graphs G_{t_1} and G_{t_2} are edge-disjoint because of the introduce-edge nodes.

We iterate over all $s \in [0 \dots n]$ and all $\bar{\kappa} \in [0 \dots \kappa]$ and use Main Lemma 9.1.23 to compute the language $R_{\kappa, \bar{\kappa}} := \text{ParSol}[t_1, \bar{\kappa}] \oplus \text{ParSol}[t_2, \kappa - \bar{\kappa}]$ and the function $g_{\kappa, \bar{\kappa}}: R_{\kappa, \bar{\kappa}} \rightarrow [1 \dots 2^n]$ as the convolution of $f_{t_1, \bar{\kappa}}$ and $f_{t_2, \kappa - \bar{\kappa}}$.

Then, for all $\kappa \in [0 \dots n]$, we define

$$\text{ParSol}[t, \kappa] := \bigcup_{\bar{\kappa} \in [0 \dots \kappa]} R_{\kappa, \bar{\kappa}} = \bigcup_{\bar{\kappa} \in [0 \dots n]} \text{ParSol}[t_1, \bar{\kappa}] \oplus \text{ParSol}[t_2, \kappa - \bar{\kappa}]$$

and

$$f_{t, \kappa}(x) := \sum_{\substack{\bar{\kappa} \in [0 \dots \kappa] \\ \text{s.t. } x \in R_{\kappa, \kappa - \bar{\kappa}}}} g_{\kappa, \kappa - \bar{\kappa}}(x)$$

for all state-vectors $x \in \text{ParSol}[t, \kappa]$.

This concludes the computation of the languages $\text{ParSol}[t, \kappa]$.

To decide whether the input graph G has a (σ, ρ) -set of a given size s , we consider the root node $r \in V(T)$ for which we know that $\beta(r) = \emptyset$ and $V_r = V(G)$. If $s \in [0 \dots n]$, then graph G has $f_{r, s}(\varepsilon)$ different (σ, ρ) -set of size exactly s and, otherwise, trivially no solution.

It remains to bound the running time of the algorithm.

Claim 9.1.24.1. *The algorithm terminates in time $\tau^w \cdot (2^{t_{\text{top}}} + n)^{\mathcal{O}(1)}$.*

Proof of Claim. As a first step, we analyze the procedure for the individual types of nodes. Clearly each the leaf node can be processed in time $\mathcal{O}(n)$ as the sets $\text{ParSol}[t, \kappa]$ are either empty or contain a single element.

Introduce Vertex Node: For every state-vector $x \in \mathbb{A}^{\beta(t')}$, we create two new state-vectors. Thus, we can compute all languages $\text{ParSol}[t, \kappa]$ by iterating over each set $\text{ParSol}[t', \kappa]$ once. This takes time $\tau^w \cdot w^{\mathcal{O}(1)} \cdot n$ in total by the bound from Equation (9.1.19).

Introduce Edge Node: For every state-vector $x \in \mathbb{A}^{\beta(t')}$, there is at most one valid extension. Thus, we can compute all languages $\text{ParSol}[t, \kappa]$ by iterating over all sets $\text{ParSol}[t', \kappa]$ once. This takes time $\tau^w \cdot w^{\mathcal{O}(1)} \cdot n$ in total using the bound on $|\text{ParSol}[t', \kappa]|$ from Equation (9.1.19).

Forget Node: By the bound from Equation (9.1.19), we can compute each set $\text{ParSol}[t, \kappa]$ in time $\tau^w \cdot w^{\mathcal{O}(1)}$. By the same bound, computing the function $f_{t, \kappa}$ takes time $\tau^w \cdot t_{\text{top}} \cdot w^{\mathcal{O}(1)}$ as we have to iterate over all entries of σ and ρ .

In total, a forget node can be processed in time $\tau^w \cdot t_{\text{top}} \cdot w^{\mathcal{O}(1)} \cdot n$.

Join Node: The time to process a join node is determined by the time to compute the sets $R_{\kappa, \bar{\kappa}}$ and the functions $g_{\kappa, \bar{\kappa}}$. For these computations, we invoke Main Lemma 9.1.23 $\binom{n}{2}$ times. Therefore, the entire join node can be processed in time

$$n^2 \cdot (\tau^w \cdot (w + 2^{t_{\text{top}}} + n)^{\mathcal{O}(1)}) \leq \tau^w \cdot (2^{t_{\text{top}}} + n)^{\mathcal{O}(1)}.$$

Observe that the root node can be process in constant time assuming that all sets and functions have been computed.

As a tree decomposition of width w with introduce edge nodes contains $\mathcal{O}(n \cdot w + m)$ nodes and $w \leq n$, the entire algorithm operates in time

$$\tau^w \cdot (2^{t_{\text{top}}} + n)^{\mathcal{O}(1)}$$

which concludes the proof to bound the running time. $\triangleleft$

This concludes the proof of Theorem 9.1.24, which provides an algorithm for (σ, ρ) -DOMSET in the case of structured sets. $\square$

9.2 Parameterizing by the Number of Excluded Degrees

In this section we focus on (σ, ρ) -DOMSET where at least one of the sets σ, ρ is cofinite. The previous general algorithm from Theorem 9.1.24 determines, for every node t and state-vector $x \in \mathbb{A}^{\beta(t)}$, whether there is a partial solution $S \subseteq V_t$ that has type x . From our lower bound results in Chapter 10, we know that this strategy is essentially optimal when we want to count the number of solutions. But, for the decision version, the idea of *representative sets* can give significant improvements in some cases.

To formally state the running time, we use the definition of cost from Definition 8.1.7, which we restate here for convenience.

Definition 8.1.7 (Cost of a Set). *Consider a set $\tau \subseteq \mathbb{N}$ that is either finite or cofinite. Based on the set τ , we define the number $\text{cost } \tau$ as*

$$\text{cost } \tau := \begin{cases} \max \tau & \text{if } \tau \text{ is finite,} \\ |\mathbb{N} \setminus \tau| & \text{if } \tau \text{ is cofinite.} \end{cases}$$

Formally, the aim of this section is to prove Main Theorem III.2 where $\omega \leq 2.371552$ denotes the matrix multiplication exponent [135].

Main Theorem III.2. *Fix two finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$. Based on these sets we define $t_{\text{cost}} := \max\{\text{cost } \sigma, \text{cost } \rho\}$.*

Given a graph G with a tree decomposition of width w , and an integer s , we can decide whether G has a (σ, ρ) -set of size exactly s in time

$$2^w \cdot (t_{\text{cost}} + 1)^{w \cdot (\omega + 1)} \cdot |G|^{\mathcal{O}(1)}.$$

We stress again that this algorithm from Main Theorem III.2 works for all finite or cofinite sets σ and ρ . However, in the case where both sets σ and ρ are finite, it is slower than the existing algorithm from Main Theorem III.1.

A Motivating Example. To understand the idea behind the proof of Main Theorem III.2, let us consider the problem with $\sigma = \{0\}$, $\rho = \mathbb{N} \setminus \{10\}$, and suppose that $\beta(t) = \{v\}$, that is, we consider only a single vertex. Then, the partial solutions $S \subseteq V_t$ have $r_{\text{top}} + s_{\text{top}} + 2 = 13$ different types; either $v \in S$ and v has 0 neighbors in S , or $v \notin S$ and v has 0, 1, $\dots$, 10, or ≥ 11 neighbors in S .

However, we do not need to know exactly which of these types correspond to partial solutions. A smaller amount of information is already sufficient to decide if there is a partial solution that is compatible with some extension $S' \subseteq V(G) \setminus V_t$. For example, if we have partial solutions for, say, the types σ_0 , ρ_7 , and ρ_8 , then *every* extension S' that extends *some* partial solution $S \subseteq V_t$ extends one of these three partial solutions. Indeed, suppose that $v \notin S$ and extension S' gives r further neighbors to v , then S can be replaced by a partial solution corresponding to the ρ_7 state unless $r = 3$, in which case S can be replaced by the solution corresponding to ρ_8 .

We use the earlier results from Section 3.2 about representative sets to formalize this observation especially for the case when $|\beta(t)| > 1$. This allows us to efficiently solve the problem.

Idea of the Algorithm. Intuitively, for a node t we do not compute the entire set $L \subseteq \mathbb{A}^{\beta(t)}$ of *all* state-vectors that are compatible with V_t , but only a *representative set* $\mathcal{S} \subseteq L$ such that the following holds: If there is a partial solution for some $x \in L$ that can be extended to a full solution via some $y \in \mathbb{A}^{\beta(t)}$, then there is also a partial solution $x' \in \mathcal{S}$ that can be extended to a full solution via y .

To formalize the notion of representative sets, we use the definition of the compatibility graph from Definition 3.2.14.

Definition 3.2.14. Consider a list $\mathcal{F} = (F_1, \dots, F_k)$ of k sets $F_i \subseteq \mathbb{N}$ for all $i \in [1..k]$, which are considered to be sets of forbidden numbers and a list $\mathcal{P} = (P_1, \dots, P_\ell)$ of ℓ sets $P_j \subseteq \mathbb{N}$ for all $j \in [1..\ell]$, which are considered to be sets of positive numbers. The compatibility graph for forbidden sets $\mathcal{F}$ and positive sets $\mathcal{P}$ is the infinite graph $\mathcal{C}_{\mathcal{F}, \mathcal{P}}$ with

- vertex set $V(\mathcal{C}_{\mathcal{F}, \mathcal{P}}) := L^{k+\ell} \cup R^{k+\ell}$ where L and R are disjoint sets both identified with $\mathbb{N}$, and
- edge set $E(\mathcal{C}_{\mathcal{F}, \mathcal{P}}) := \{((a_1, \dots, a_{k+\ell}), (b_1, \dots, b_{k+\ell})) \mid \forall i \in [1..k] : a_i + b_i \notin F_i \text{ and } \forall j \in [1..\ell] : a_{k+j} + b_{k+j} \in P_j\}$.

The main conceptual idea behind our algorithm is as follows. If one of the sets σ, ρ is cofinite, then it is possible to obtain representative sets $\mathcal{S} \subseteq L$ that are much smaller than the number of partial solutions that are maintained by standard dynamic programming algorithms (for example, the algorithm by van Rooij [130] as stated in Theorem 8.1.2).

More precisely, the size of the representative sets depends on the *number* of missing integers from the cofinite sets but *does not* depend on the largest missing element. Thus, as outlined above, this technique is particularly efficient when few large integers are missing from ρ or σ . Formally, we use Lemma 3.2.15 to compute the representative sets of small size.

Lemma 3.2.15. Consider a list $\mathcal{F} = (F_1, \dots, F_k)$ of k sets $F_i \subseteq \mathbb{N}$ with $|F_i| \leq r$ for all $i \in [1..k]$, and a list $\mathcal{P} = (P_1, \dots, P_\ell)$ of ℓ sets $P_j \subseteq \mathbb{N}$ with $\max P_j \leq r$ for all $j \in [1.. \ell]$.

Further, write $\mathcal{S} \subseteq \mathbb{N}^k$ for a finite set. Then, one can compute an $(\mathcal{F}; \mathcal{P})$ -representative set $\mathcal{S}'$ of $\mathcal{S}$ such that $|\mathcal{S}'| \leq (r+1)^{k+\ell}$ in time $\mathcal{O}(|\mathcal{S}| \cdot (r+1)^{\max\{\ell, k \cdot (\omega-1)\}} \cdot (k+\ell))$.

However, the price we have to pay by using Lemma 3.2.15 is that the algebraic techniques require matrix operations and the matrix multiplication exponent ω appears in the exponent of the running time. This makes it unlikely to obtain matching lower bounds similar to Main Theorem III.4.

With these preliminaries, the formal idea is as follows. For ease of notation, we assume for now that σ is cofinite and ρ is finite. Consider a vertex set $\beta \subseteq V(G)$ consisting of $k+\ell$ vertices $v_1, \dots, v_{k+\ell}$. Moreover, fix a σ -vector $\vec{s} \in \{0, 1\}^{k+\ell}$ that has k entries corresponding to selected vertices (with the cofinite set σ), and ℓ entries corresponding to unselected vertices (with the finite set ρ).

We relabel the vertices in β such that the first k vertices correspond to a cofinite set, that is, we have $\vec{s}[i] = 1$ for all $i \in [1..k]$, and the remaining ℓ vertices correspond to finite sets, that is, we have $\vec{s}[k+i] = 0$ for all $i \in [1.. \ell]$.

We define $F_i := \mathbb{N} \setminus \sigma$ for all $i \in [1..k]$ and $P_i := \rho$ for all $i \in [1.. \ell]$, and set $\mathcal{F}_{\vec{s}} = (F_1, \dots, F_k)$ and $\mathcal{P}_{\vec{s}} = (P_1, \dots, P_\ell)$. Then, it suffices to keep an $(\mathcal{F}_{\vec{s}}; \mathcal{P}_{\vec{s}})$ -representative set of the weight-vectors of those partial solutions that have σ -vector $\vec{s}$.

To modify the representative sets at the nodes of the given tree decomposition, we use predefined operations which preserve the representative set property. More formally, we base our algorithm on the operations from Lemma 3.2.16.

Lemma 3.2.16. Let $F_1, \dots, F_k, P_1, \dots, P_\ell \subseteq \mathbb{N}$ denote $k+\ell$ finite sets and write $\mathcal{F} = (F_1, \dots, F_k)$ and $\mathcal{P} = (P_1, \dots, P_\ell)$. Further, write $\mathcal{S} \subseteq \mathbb{N}^{k+\ell}$ for a finite set and let $\mathcal{S}'$ denote an $(\mathcal{F}; \mathcal{P})$ -representative set of $\mathcal{S}$.

- (a) For every vector $d \in \mathbb{Z}^{k+\ell}$, the set $\mathcal{S}' + d := \{s + d \mid s \in \mathcal{S}\}$ is an $(\mathcal{F}; \mathcal{P})$ -representative set of $\mathcal{S} + d$.
- (b) For every $i \in [1..k]$, the set $\mathcal{S}' \boxplus i := \{s[1..i-1] \ s[i+1..k+\ell] \mid s \in \mathcal{S}', s[i] \notin F_i\}$ is an $(\mathcal{F}'; \mathcal{P})$ -representative set of $\mathcal{S} \boxplus i$ where $\mathcal{F}' := (F_1, \dots, F_{i-1}, F_{i+1}, \dots, F_k)$.
- (c) For every $j \in [1.. \ell]$, the set $\mathcal{S}' \boxplus (k+j) := \{s[1..k+j-1] \ s[k+j+1..k+\ell] \mid s \in \mathcal{S}', s[k+j] \in P_j\}$ is an $(\mathcal{F}; \mathcal{P}')$ -representative set of $\mathcal{S} \boxplus (k+j)$ where $\mathcal{P}' := (P_1, \dots, P_{j-1}, P_{j+1}, \dots, P_\ell)$.

For every finite set $Q \subseteq \mathbb{N}$, and

- (d) for every $i \in [1..k+1]$, the set $\mathcal{S}' \boxplus i := \{s[1..i-1] \ 0 \ s[i..k+\ell] \mid s \in \mathcal{S}'\}$ is an $(\mathcal{F}'; \mathcal{P})$ -representative set of $\mathcal{S} \boxplus i$ where $\mathcal{F}' := (F_1, \dots, F_{i-1}, Q, F_i, \dots, F_k)$, and
- (e) for every $j \in [1.. \ell+1]$, the set $\mathcal{S}' \boxplus (k+j) := \{s[1..k+j-1] \ 0 \ s[k+j..k+\ell] \mid s \in \mathcal{S}'\}$ is an $(\mathcal{F}; \mathcal{P}')$ -representative set of $\mathcal{S} \boxplus (k+j)$ where $\mathcal{P}' := (P_1, \dots, P_{j-1}, Q, P_j, \dots, P_\ell)$.

For every finite set $\mathcal{R} \subseteq \mathbb{N}^{k+\ell}$ and every $(\mathcal{F}; \mathcal{P})$ -representative sets $\mathcal{R}'$ of $\mathcal{R}$,

- (f) the set $S' \cup \mathcal{R}'$ is an $(\mathcal{F}; \mathcal{P})$ -representative set of $S \cup \mathcal{R}$, and
- (g) the set $S' + \mathcal{R}' := \{s + r \mid s \in S', r \in \mathcal{R}'\}$ is an $(\mathcal{F}; \mathcal{P})$ -representative set of $S + \mathcal{R}$.

With these results, we have everything ready to state the improved algorithm for (σ, ρ) -DOM-SET when at least one set is cofinite, that is, we prove Main Theorem III.2.

Main Theorem III.2. Fix two finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$. Based on these sets we define $t_{\text{cost}} := \max\{\text{cost } \sigma, \text{cost } \rho\}$.

Given a graph G with a tree decomposition of width w , and an integer s , we can decide whether G has a (σ, ρ) -set of size exactly s in time

$$2^w \cdot (t_{\text{cost}} + 1)^{w \cdot (\omega + 1)} \cdot |G|^{\mathcal{O}(1)}.$$

Proof. Denote by (T, β) a nice tree decomposition of G of width w where we assume the existence of introduce edge nodes and each edge is introduced exactly once. See Section 2.2 for further details on how to achieve this.

For the remainder of this proof, with $n := |V(G)|$ we define $\mathbb{A} := \mathbb{A}_n = \{\sigma_0, \dots, \sigma_n, \rho_0, \dots, \rho_n\}$ as the possible states of the vertices.

As the notion of the representative set depends on the current node $t \in V(T)$ and the σ -vector $\vec{s} \in \{0, 1\}^{\beta(t)}$ (but not on the size bound κ), we first introduce some notation.

To this extent, we fix a node $t \in V(T)$ and a σ -vector $\vec{s} \in \{0, 1\}^{\beta(t)}$. Also, consider a vertex $v \in \beta(t)$. We say that vertex v is an F -vertex if $\vec{s}[v] = 1$ and σ is cofinite, or if $\vec{s}[v] = 0$ and ρ is cofinite. In the former case, we define $F_v := \mathbb{N} \setminus \sigma$, and in the latter case we define $F_v := \mathbb{N} \setminus \rho$. If vertex v is not an F -vertex, then we say that v is a P -vertex. Note that v is a P -vertex if $\vec{s}[v] = 1$ and σ is finite, or if $\vec{s}[v] = 0$ and ρ is finite. In the former case, we define $P_v := \sigma$, and in the latter case we define $P_v := \rho$.

We order the vertices in $\beta(t)$ as $\beta(t) = \{v_1, \dots, v_k, v_{k+1}, \dots, v_{k+\ell}\}$ such that $v_1, \dots, v_k$ are F -vertices and $v_{k+1}, \dots, v_{k+\ell}$ are P -vertices.

We write $\mathcal{F}(t, \vec{s}) := (F_{v_1}, \dots, F_{v_k})$ and $\mathcal{P}(t, \vec{s}) := (P_{v_{k+1}}, \dots, P_{v_{k+\ell}})$ in the following. We denote by $\mathcal{C}(t, \vec{s}) := \mathcal{C}_{\mathcal{F}(t, \vec{s}), \mathcal{P}(t, \vec{s})}$ the compatibility graph from Definition 3.2.14 for the forbidden sets $\mathcal{F}(t, \vec{s})$ and the positive sets $\mathcal{P}(t, \vec{s})$. To simplify notation, we say that a set R is a $(t, \vec{s})$ -representative set of some set L if and only if $R \subseteq_{\mathcal{C}(t, \vec{s})\text{-rep}} L$.

For each node $t \in V(T)$, each size bound $\kappa \in [0 \dots |V(G)|]$, and each σ -vector $\vec{s} \in \{0, 1\}^{\beta(t)}$, we define $\hat{L}_{t, \kappa, \vec{s}} \subseteq \mathbb{A}^{\beta(t)}$ as the set of all state-vectors $x \in \mathbb{A}^{\beta(t)}$ with $\vec{\sigma}(x) = \vec{s}$ that are compatible with $(G_t, \beta(t))$ and the witnessing partial solution contains κ selected vertices from $V_t \setminus \beta(t)$. As all vectors in $\hat{L}_{t, \kappa, \vec{s}}$ share the same σ -vector, we focus on the weight-vectors only and define $L_{t, \kappa, \vec{s}} := \vec{w}(\hat{L}_{t, \kappa, \vec{s}}) = \{\vec{w}(x) \mid x \in \hat{L}_{t, \kappa, \vec{s}}\}$.

Then, the goal of the algorithm is to fill the table $\text{ParSol}[\star, \star, \star]$ such that

$$\text{ParSol}[t, \kappa, \vec{s}] \subseteq_{\mathcal{C}(t, \vec{s})\text{-rep}} L_{t, \kappa, \vec{s}}.$$

Algorithm. With these definitions we have everything ready to formally state the algorithm that fills the table $\text{ParSol}[\star, \star, \star]$. We traverse the tree T of the given tree decomposition in post-order and apply one of the following rules depending on the type of the current node t . Thus, for each step we can assume that the algorithm already computed all table entries $\text{ParSol}[t', \star, \star]$ for each child t' of the current node t .

Leaf Node: When t is a leaf node, then the corresponding bag is empty, that is, we know that $\beta(t) = V_t = \emptyset$. Therefore, we set $\text{ParSol}[t, 0, \emptyset] = \{\varepsilon\}$. For all other choices of κ and $\vec{s}$, we set $\text{ParSol}[t, \kappa, \vec{s}] = \emptyset$.

Introduce Vertex Node: Next, consider the case that t is an introduce vertex node and write t' for the unique child of t . Suppose that $\beta(t) = \beta(t') \cup \{v\}$, that is, $v \in V(G)$ is the vertex introduced at t . Note that v is an isolated vertex in G_t because of the introduce edge nodes.

For all $\kappa \in [0 \dots |V(G)|]$ and $\vec{s} \in \{0, 1\}^{\beta(t)}$ with $\vec{s}[v] = 1$, we define

$$R_{\kappa, \vec{s}} := \text{ParSol}[t', \kappa, \vec{s}[\beta(t')]] \boxplus v.$$

Then, we use Lemma 3.2.15 to compute a $(t, \vec{s})$ -representative set of $R_{\kappa, \vec{s}}$ and define $\text{ParSol}[t, \kappa, \vec{s}]$ as the resulting set. Note that depending on $\vec{s}$, the notion of the $(t, \vec{s})$ -representation changes as the vertex v is either selected or not; however, the definition of the new vectors is independent of this.

Introduce Edge Node: Next, consider the case that t is an introduce edge node with child t' and denote by uv the edge introduced. Note that we have $u, v \in \beta(t') = \beta(t)$.

For each $\vec{s}$, we define the vector $\chi_{uv, \vec{s}} \in \{0, 1\}^{\beta(t)}$ where $\chi_{uv, \vec{s}}[u] = \vec{s}[v]$ and $\chi_{uv, \vec{s}}[v] = \vec{s}[u]$. For all vertices $w \in \beta(t) \setminus \{u, v\}$, we set $\chi_{uv, \vec{s}}[w] = 0$, that is, the vector χ_{uv} accounts of the additional selected neighbors of u and v when edge uv is added.

For all $\kappa \in [0 \dots |V(G)|]$ and $\vec{s} \in \{0, 1\}^{\beta(t)}$, we define

$$R_{\kappa, \vec{s}} := \text{ParSol}[t', \kappa, \vec{s}] + \chi_{uv, \vec{s}}.$$

Then, we use Lemma 3.2.15 to compute a $(t, \vec{s})$ -representative set of $R_{\kappa, \vec{s}}$. We define $\text{ParSol}[t, \kappa, \vec{s}]$ as the resulting set of this computation.

Forget Node: Consider the case that the current node t is a forget node and write t' for the unique child of t . Suppose that $\beta(t) = \beta(t') \setminus \{v\}$, that is, $v \in V(G)$ is the vertex forgotten at t .

For all $\kappa \in [0 \dots |V(G)|]$ and $\vec{s} \in \{0, 1\}^{\beta(t)}$, we define

$$R_{\kappa, \vec{s}} := (\text{ParSol}[t', \kappa - 1, \vec{s} \sqcup (v \mapsto 1)] \boxplus v) \cup (\text{ParSol}[t', \kappa, \vec{s} \sqcup (v \mapsto 0)] \boxplus v).$$

As for the previous cases, we use Lemma 3.2.15 to compute a $(t, \vec{s})$ -representative set of $R_{\kappa, \vec{s}}$ and define $\text{ParSol}[t, \kappa, \vec{s}]$ as the resulting set.

Join Node: Finally, suppose that the current node t is a join node and write t_1, t_2 for the two children of t . Note that $\beta(t) = \beta(t_1) = \beta(t_2)$.

For all $\kappa \in [0 \dots |V(G)|]$ and $\vec{s} \in \{0, 1\}^{\beta(t)}$, the algorithm computes

$$R_{\kappa, \vec{s}} := \bigcup_{\bar{\kappa} \in [0 \dots \kappa]} \text{ParSol}[t_1, \bar{\kappa}, \vec{s}] + \text{ParSol}[t_2, \kappa - \bar{\kappa}, \vec{s}].$$

As before, we use Lemma 3.2.15 to compute a $(t, \vec{s})$ -representative set of $R_{\kappa, \vec{s}}$ and define $\text{ParSol}[t, \kappa, \vec{s}]$ as the resulting set.

It remains to determine the output of the entire algorithm. We assume that $r \in V(T)$ denotes the root node of the tree decomposition for which $\beta(t) = \emptyset$ and $G_r = G$. Then, the input graph G has a (σ, ρ) -set of size $\kappa \in [0 .. |V(G)|]$ if $\varepsilon \in \text{ParSol}[r, \kappa, \emptyset]$. Otherwise, there is no solution of size κ . This concludes the description of the algorithm.

Correctness. As a next step we prove the correctness of the algorithm. Afterward, we analyze the running time.

Claim 9.2.0.1. *The algorithm is correct, that is, for all nodes $t \in V(T)$, all size bounds $\kappa \in [0 .. |V(G)|]$, and all σ -vectors $\vec{s} \in \{0, 1\}^{\beta(t)}$, we have that $\text{ParSol}[t, \kappa, \vec{s}] \subseteq_{\mathcal{C}(t, \vec{s})\text{-rep}} L_{t, \kappa, \vec{s}}$.*

Proof of Claim. We prove the correctness by a structural induction on the given tree decomposition.

The leaf nodes provide the base cases and trivially satisfy the claim as the bags of the nodes are empty. We proceed to the other nodes and assume inductively that the claim holds for all children of the current node.

Introduce Vertex Node: As the vertex v is introduced at t , it is an isolated vertex. Hence, we know that $L_{t, \kappa, \vec{s}} = L_{t', \kappa, \vec{s}}[\beta(t')] \boxplus v$. Observe that the notion of $(t, \vec{s})$ -representation changes depending on whether $\vec{s}[v] = 1$ or not. However, the definition of the operation $\boxplus v$ is *not* affected by this.

By the induction hypothesis and Cases (d) and (e) of Lemma 3.2.16, we know that $R_{\kappa, \vec{s}}$ is a $(t, \vec{s})$ -representative set of $L_{t, \kappa, \vec{s}}$ and the claim follows by transitivity of $(t, \vec{s})$ -representation.

Introduce Edge Node: As the edge uv is introduced at this node, the vertices u and v are unconnected in the graph $G_{t'}$. Hence, adding the edge uv might change the number of selected neighbors for the vertices u and v depending on the other vertex. Thus, we have that $L_{t, \kappa, \vec{s}} = L_{t', \kappa, \vec{s}} + \chi_{uv, \vec{s}}$.

As before, by the induction hypothesis and Case (a) of Lemma 3.2.16, the set $R_{\kappa, \vec{s}}$ is a $(t, \vec{s})$ -representative set of $L_{t, \kappa, \vec{s}}$ and the claim follows by transitivity.

Forget Node: We can classify the solutions that contribute to $L_{t, \kappa, \vec{s}}$ depending on the selection status of vertex v . As the size bound κ counts only the selected vertices that are not contained in the current bag, we have

$$L_{t, \kappa, \vec{s}} = (L_{t', \kappa, \vec{s} \sqcup (v \mapsto 0)} \boxplus v) \cup (L_{t', \kappa-1, \vec{s} \sqcup (v \mapsto 1)} \boxplus v).$$

Indeed, this covers the solutions where v is not selected and the number of neighbors is contained in ρ , and the solutions where v is selected and the number of neighbors is contained in σ .

Once more, by the induction hypothesis and Cases (b), (c), and (d) of Lemma 3.2.16, the set $R_{\kappa, \vec{s}}$ is a $(t, \vec{s})$ -representative set of $L_{t, \kappa, \vec{s}}$. Then, the claim follows by transitivity of $(t, \vec{s})$ -representation.

Join Node: As the vertex sets $V_{t_1} \setminus \beta(t)$ and $V_{t_2} \setminus \beta(t)$ are disjoint, we can partition the κ selected vertices of each partial solution into the $\bar{\kappa}$ selected vertices from $V_{t_1} \setminus \beta(t)$ and the $\kappa - \bar{\kappa}$ selected vertices from $V_{t_2} \setminus \beta(t)$. Note that the selected vertices in $\beta(t)$ are not included in this count but need the same selection status in the partial solutions corresponding to t_1 and t_2 . Hence, we have that $L_{t,\kappa,\vec{s}} = \bigcup_{\bar{\kappa} \in [0 \dots \kappa]} L_{t_1,\bar{\kappa},\vec{s}} + L_{t_2,\kappa-\bar{\kappa},\vec{s}}$.

For the last time, by the induction hypothesis and Cases (f) and (g) of Lemma 3.2.16, the set $R_{\kappa,\vec{s}}$ is a $(t, \vec{s})$ -representative set of $L_{t,\kappa,\vec{s}}$. Then, the claim follows by the transitivity of $(t, \vec{s})$ -representation. $\blacktriangleleft$

As a last step of the proof, we analyze the running time.

Claim 9.2.0.2. *The algorithm operates in time $2^w \cdot (t_{\text{cost}} + 1)^{w \cdot (\omega+1)} \cdot |V(G)|^{\mathcal{O}(1)}$ and in time $2^w \cdot (t_{\text{cost}} + 1)^{w \cdot \omega} \cdot |V(G)|^{\mathcal{O}(1)}$ if we are given path decomposition of width w .*

Proof of Claim. We first observe that for all nodes $t \in V(T)$, all size bounds $\kappa \in [0 \dots |V(G)|]$, and σ -vectors $\vec{s} \in \{0, 1\}^{\beta(t)}$, we have that $|\text{ParSol}[t, \kappa, \vec{s}]| \leq (\max\{\text{cost } \sigma, \text{cost } \rho\} + 1)^{|\beta(t)|}$. Indeed, this follows directly from the computation of the representative sets as the last step in each case of the algorithm.

By the design of the algorithm, the computation at the join nodes determines the running time of the algorithm.

Now, fix a join node t , a size bound $\kappa \in [0 \dots |V(G)|]$ and a σ -vector $\vec{s}$. The computation of the set $R_{\kappa,\vec{s}}$ takes time $\mathcal{O}(|V(G)| \cdot (t_{\text{cost}} + 1)^{2(w+1)})$ which also bound their size. The procedure from Lemma 3.2.15 computes the representative sets in time

$$\mathcal{O}(|R_{\kappa,\vec{s}}| \cdot (t_{\text{cost}} + 1)^{(w+1) \cdot (\omega-1)} \cdot (w + t_{\text{cost}}))$$

and the size bound on $R_{\kappa,\vec{s}}$ yields

$$\begin{aligned} & \mathcal{O}(|V(G)| \cdot (t_{\text{cost}} + 1)^{2(w+1)} \cdot (t_{\text{cost}} + 1)^{(w+1) \cdot (\omega-1)} \cdot (w + t_{\text{cost}})) \\ & \leq (t_{\text{cost}} + 1)^{w \cdot (\omega+1)} \cdot |V(G)|^{\mathcal{O}(1)} \end{aligned}$$

as a bound on the running time.

So in total, the computation of $\text{ParSol}[t, \kappa, \vec{s}]$ for all nodes $t \in V(T)$, all size bounds $\kappa \in [0 \dots |V(G)|]$, and all σ -vectors $\vec{s} \in \{0, 1\}^{\beta(t)}$, takes time

$$(w \cdot |V(G)| + |E(G)|) \cdot |V(G)| \cdot 2^{w+1} \cdot ((t_{\text{cost}} + 1)^{w \cdot (\omega+1)} \cdot |V(G)|^{\mathcal{O}(1)})$$

which concludes the proof. $\blacktriangleleft$

This concludes the improved algorithm for (σ, ρ) -DOMSET for the case when at least one set is cofinite. $\square$

Lower Bounds for General Domination

10

In the following we prove the lower bound results for the decision and counting version of (σ, ρ) -DOMSET. As previously mentioned, the running times depend non-trivially on σ and ρ . Thus, we state them in terms of $c_{\sigma, \rho}$ as defined in Definition 8.1.4, which we restate here for convenience.

Definition 8.1.4. Consider two finite or cofinite sets σ and ρ of non-negative integers. We define $c_{\sigma, \rho}$ by setting

- $c_{\sigma, \rho} := s_{\text{top}} + r_{\text{top}} + 2$ if (σ, ρ) is not m -structured for any $m \geq 2$,
- $c_{\sigma, \rho} := \max\{s_{\text{top}}, r_{\text{top}}\} + 2$ if (σ, ρ) is 2-structured, but not m -structured for any $m \geq 3$, and $s_{\text{top}} = r_{\text{top}}$ is even, and
- $c_{\sigma, \rho} := \max\{s_{\text{top}}, r_{\text{top}}\} + 1$, otherwise.

With this definition, our goal is the formal proof of Main Theorem III.3 for the decision version of (σ, ρ) -DOMSET.

Main Theorem III.3. Fix two finite sets $\sigma, \rho \subseteq \mathbb{N}$ with $0 \notin \rho$. If there is an $\varepsilon > 0$ and an algorithm that decides whether there is a (σ, ρ) -set in a given graph G with a given path decomposition of width w in time $(c_{\sigma, \rho} - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-SETH fails.

We note that this lower bound requires that the sets σ and ρ are finite. For the counting version, we remove this requirement and consider all pairs of sets that are finite or cofinite and non-trivial. Formally, we prove Main Theorem III.4.

Main Theorem III.4. Fix two non-empty finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$ with $\rho \neq \{0\}$ and $(\sigma, \rho) \neq (\mathbb{N}, \mathbb{N})$.

If there is an $\varepsilon > 0$ and an algorithm that counts the number of (σ, ρ) -sets in a given graph G with a given path decomposition of width w in time $(c_{\sigma, \rho} - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-#SETH fails.

Organization. In Section 10.1 we first present the outline of the proofs of Main Theorems III.3 and III.4. Along with this outline, we modularize the proof and provide multiple intermediate steps. Afterward, we prove these intermediate steps in the mostly self-contained subsequent sections.

10.1 Outline and Preliminaries

The proofs of the lower bounds consist of a plethora of intermediate results that, taken together, make the proofs rather long and complex. In order to keep this complexity manageable, we break the lower bound results into multiple smaller pieces. This allows us to focus on one problem at a time and makes each of the intermediate results independently verifiable. Additionally, this way, the purpose of the different proof ingredients is clear before we present their (in some cases long and technical) proofs.

High-Level Approach. We prove the lower bound results by a modularized approach that partially follows previous work (for example, [35, 97]), on problems parameterized by treewidth and pathwidth. However, the high-level structure is then filled with many technical details that are designed specifically for the problem at hand.

From a higher perspective, the proofs of Main Theorems III.3 and III.4 consist of the following steps. We refer to Figure 10.1 for an overview of these steps and how they are combined.

1. We first construct auxiliary gadgets which we refer to as *providers* that allow us to “provide” a certain state for the *portal* vertices of these gadgets. Phrased differently, depending on the selection status of these portal vertices, the providers give a particular number of selected neighbors to these portal vertices where the number of selected neighbors can be chosen from some specified list. Moreover, all internal vertices of the providers receive a valid number of neighbors. We construct these gadgets in Section 10.2.
2. As a second step, we combine the providers to form *managers*. These managers act similarly as providers, but the portal vertices receive a valid number of neighbors as well. In contrast to the providers, the managers do this by two components, that is, for each portal vertex there is a left block and a right block, and only together these block give a valid state to the vertex.

As a main result about the managers, we prove that, depending on the properties of σ and ρ , different versions of the managers exist. Depending on the “quality” of the managers, this then leads to lower bounds of different “quality”. We construct the managers in Section 10.3.

3. Once we have said managers at hand, the next step is a lower bound for a generalization of (σ, ρ) -DOMSET where we additionally allow relational constraints on the selection status of vertices. Such a constraint consists of a scope D of vertices together with a $|D|$ -ary relation that specifies which subsets of D are allowed to be selected in a solution. We refer to this variant of the problem as (σ, ρ) -DOMSET^{Rel} for the decision version and as (σ, ρ) -#DOMSET^{Rel} for the counting version.

We use the managers from the previous step to provide a reduction from q -CSP- B to (σ, ρ) -DOMSET^{Rel} (and respectively for the counting version) and prove a lower bound for the latter problem when σ and ρ are finite or simple cofinite sets. We formalize these results in Section 10.4.

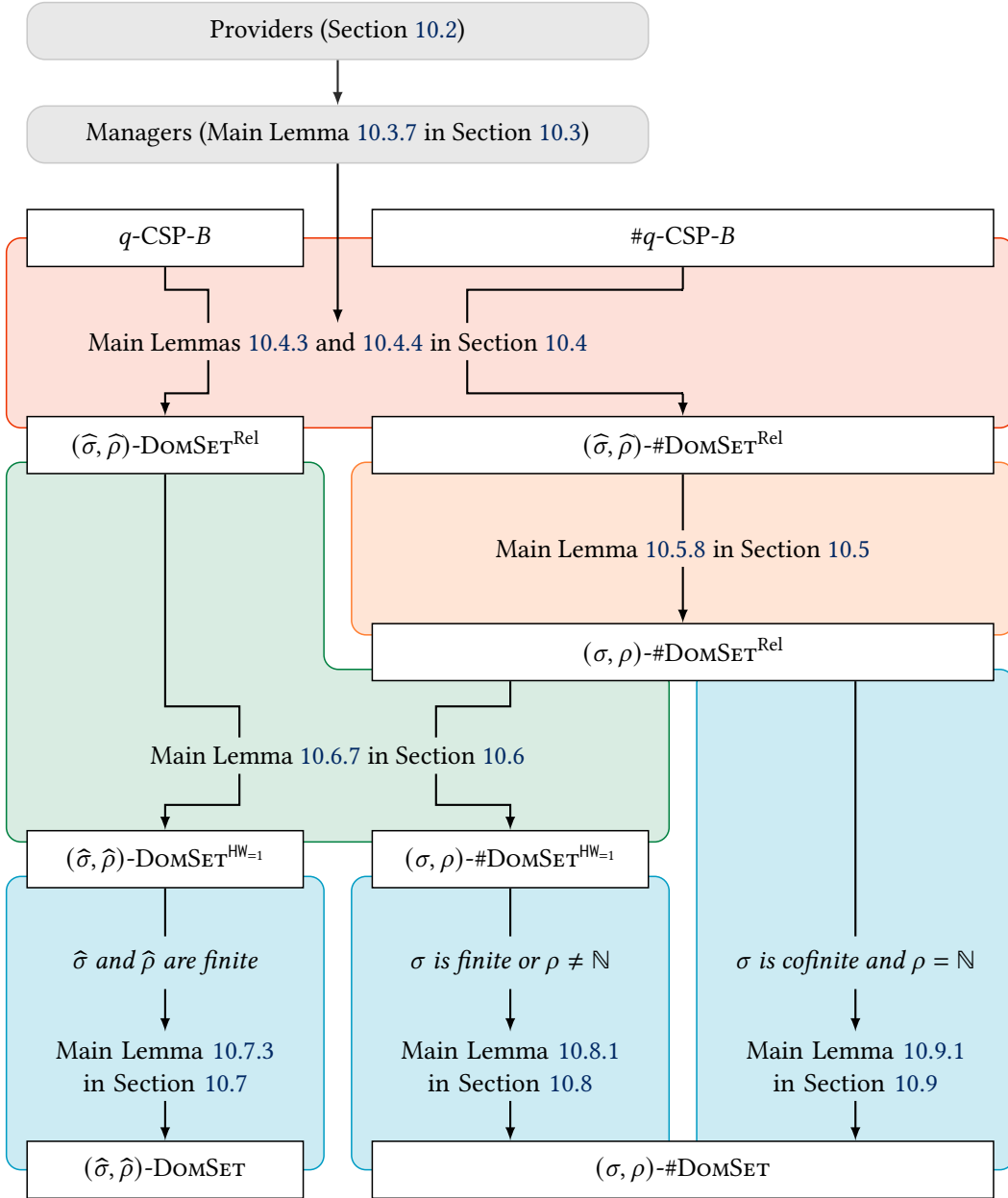


Figure 10.1: Outline of the proof structure for the lower bounds of (σ, ρ) -DOMSET from Main Theorems III.3 and III.4 proven in Chapter 10. Each colored box refers to an intermediate result which we prove in the respective section.

The left sequence considers the decision version, and the middle and right sequence consider the counting version (see Figure 10.23 on page 389 for a detailed figure for the case of the counting version).

The sets $\hat{\sigma}$ and $\hat{\rho}$ are finite or simple cofinite sets. The sets σ and ρ are finite or cofinite sets (not necessarily simple cofinite).

Additionally, for the counting version, we extend these results to arbitrary cofinite sets in Section 10.5.

4. For the remaining steps, we show how to remove the relations while preserving their properties. Then, the lower bounds for the intermediate problems extend to the original problems (σ, ρ) -DOMSET and (σ, ρ) -#DOMSET, respectively.

As a first result, we replace the arbitrary relations of the obtained (σ, ρ) -DOMSET^{Rel} instance by gadgets that only consist of simple vertices and $\text{HW}_{=1}$ -relations. We formalize these results in Section 10.6.

5. Afterward, we consider different cases to realize the remaining relations of type $\text{HW}_{=1}$. We start with the decision version as the easiest case and formalize the reduction from (σ, ρ) -DOMSET^{HW₌₁} to (σ, ρ) -DOMSET in Section 10.7.

For the counting version, we consider two cases. In the first case we assume that $\rho \neq \mathbb{N}$ or that σ is finite and provide the reduction to realize $\text{HW}_{=1}$ in Section 10.8. For the more complicated case when $\rho = \mathbb{N}$ and σ is cofinite, we provide the realization in Section 10.9.

When we combine the previous results, we obtain the lower bound for (σ, ρ) -DOMSET and (σ, ρ) -#DOMSET.

Organization. In the following, we first define some necessary notation in Section 10.1.1. Then, we formalize the intuitive results from the previous steps, give some intuition for their proofs, and combine the results to prove Main Theorems III.3 and III.4 in Section 10.1.9.

Afterward in Sections 10.2 to 10.9, we provide the formal proofs of each of these steps in the order in which they are presented above.

10.1.1 Notation

Our proofs heavily depend on the notion of relations, which we use to impose additional constraints on the selection status of vertices. We formalize this by making use of relations which we add to the graph.

Definition 10.1.1 (Graph with Relations). *We define a graph with relations as a tuple $G = (V, E, \mathcal{C})$, where V is a set of vertices, E is a set of edges on V , and $\mathcal{C}$ is a set of relational constraints, that is, each $C \in \mathcal{C}$ is in itself a tuple $(\text{scp}(C), R(C))$*

- where the scope $\text{scp}(C)$ of C is a tuple of $|\text{scp}(C)|$ vertices from V , and
- where $R(C) \subseteq 2^{\text{scp}(C)}$ is a $|\text{scp}(C)|$ -ary relation specifying possible selections within $\text{scp}(C)$.

We also say that $R(C)$ observes $\text{scp}(C)$.

The size of G is $|G| := |V| + \sum_{C \in \mathcal{C}} |R(C)|$. Slightly abusing notation, we usually do not distinguish between G and its underlying graph (V, E) . We use G to refer to both objects depending on the context.

Some relations that we frequently use are equality and Hamming weight relations. Note that these relations are symmetric, that is, they only depend on the *number* of selected vertices. Hence, we can omit the ordering of the vertices in the scope and represent their selection status by 0-1-vectors.

Definition 2.1.6. For a k -dimensional vector $x \in \{0, 1\}^k$, we define $\text{hw}(x)$ as the number of 1s in x , that is, we define $\text{hw}(x)$ as the Hamming weight of x with $\text{hw}(x) := \sum_{i=1}^k x[i]$.

For two positive integers $i, k \in \mathbb{N}$ and a set $S \subseteq \mathbb{N}$, we define the relations

$$\text{HW}_{\in S}^{(k)} := \{x \in \{0, 1\}^k \mid \text{hw}(x) \in S\}, \quad \text{HW}_{=i}^{(k)} := \text{HW}_{\in \{i\}}^{(k)} \quad \text{and} \quad \text{EQ}_k := \text{HW}_{\in \{0, k\}}^{(k)}.$$

We say that EQ_k is the equality relation on k inputs.

We use $\text{HW}_{\in S}$ to denote the relation $\text{HW}_{\in S}^{(k)}$ when the arity k of the relation is clear from the context. Additionally, we also write $\text{HW}_{\in S}$ to denote the set of the relations $\text{HW}_{\in S}^{(k)}$ for all $k \in \mathbb{N}$.

We transfer this abuse of notation to $\text{HW}_{=i}$ and EQ .

Given Definition 10.1.1, we extend the notion of (σ, ρ) -sets to graphs with relations.

Definition 10.1.2 ((σ, ρ) -Sets of a Graph with Relations). Given a graph with relations $G = (V, E, \mathcal{C})$, a (σ, ρ) -set S of G is a (σ, ρ) -set of the underlying graph (V, E) that additionally satisfies $S \cap \text{scp}(C) \in R(C)$ for every $C \in \mathcal{C}$.

We explicitly emphasize that the relations do not affect the (σ, ρ) -sets of the underlying graph (V, E) in the sense that they do not provide additional selected neighbors to the vertices in their scope. That is, the relations only “observe” the selection status of these vertices. Moreover, this allows us to add multiple relations to a single vertex.

Measures of Graphs with Relations. To formally state treewidth-preserving (pathwidth-preserving, respectively) reductions (see Definition 2.2.2), we define the treewidth (pathwidth, respectively) of a graph with relations. The crucial part is that in this definition we treat the scope of a relation as a clique.

Definition 10.1.3 (Width Measures for Graphs with Relations). Let $G = (V, E, \mathcal{C})$ be a graph with relations. Denote by G' the graph that we obtain from (V, E) when, for all $C \in \mathcal{C}$, we additionally introduce a complete set of edges on the scope $\text{scp}(C)$.

The treewidth of a graph with relations G , is the treewidth of the graph G' . Analogously, we define tree decompositions, path decompositions, and pathwidth of G as the corresponding concepts in the graph G' .

To make use of the concept of arity-preserving reductions (see Definition 2.2.3), we define the arity of a graph with relations. We ensure that this definition agrees with the intuition that the arity of a graph without relations is 0.

Definition 10.1.4 (Arity of a Graph with Relations). Given a graph with relations $G = (V, E, \mathcal{C})$, the arity of a graph with relations is $\text{ar}(G) := \max(\{0\} \cup \{|\text{scp}(C)| \mid C \in \mathcal{C}\})$, that is, the maximum arity of some relation of $\mathcal{C}$ or 0 if no relation is present.

If, for some constant d , a graph with relations has arity at most d , then its size is upper bounded by $|V| + |\mathcal{C}| \cdot 2^d$. Hence, for the sake of polynomial-time reductions, this size is essentially $|V| + |\mathcal{C}|$. By defining pathwidth and arity of a graph with relations, we can now use computational problems that take as input a graph with relations in the context of pathwidth-preserving reductions, as well as in the context of arity-preserving reductions, or even pa-reductions (Definition 2.2.4).

Providers and Realizers. Additionally, we generalize the definition of L -providers and L -realizers from Definition 8.1.10 to graphs with relations.

Definition 10.1.5 (Providers and Realizers with Relations). *Let $G = (V, E, \mathcal{C})$ denote a graph with relations and let $U \subseteq V$. Then, (G, U) is a graph with relations and portals.*

A set $S \subseteq V$ is a partial solution of (G, U) if

- *the set S is a partial solution of (V, E, U) , that is, S has the properties from Definition 8.1.8, and*
- *the set S satisfies that $S \cap \text{scp}(C) \in R(C)$ for every $C \in \mathcal{C}$.*

We generalize the notions of compatible vectors, L -providers, and L -realizers from Definitions 8.1.9 and 8.1.10 accordingly.

10.1.2 Providers and Managers

As a first set of results, we construct various providers in Section 10.2. These providers serve multiple purposes. One of the most useful providers are the $\{\sigma_s, \rho_r\}$ -providers from Lemma 10.2.1 for any $s \in \sigma$ and $r \in \rho$. These providers allow us to introduce isolated vertices that are only connected to the remaining graph via some relation, that is, these vertices do not have any neighbors outside their provider which is fine as the providers provide a valid number of neighbors.

We use the remaining providers in Section 10.2—which is the significantly larger number—to construct new type of gadgets. These so-called *managers* allow us to add selected neighbors to possibly selected vertices. In fact, a manager additionally ensures that the vertices to which it is connected get a valid number of neighbors such that the σ -constraints and ρ -constraints are always satisfied. Note that this is not the case for providers, as they only ensure that all vertices *inside* the gadget receive a valid number of selected neighbors. We refer to Figure 10.2 for an illustration of such a manager.

Definition 10.3.1 (Managers). *For a set $A \subseteq \mathbb{A}_{\text{full}}$ of states, an A -manager is an infinite family $((G_\ell, U_\ell))_{\ell \geq 1}$ of pairs (G_ℓ, U_ℓ) such that*

- *G_ℓ is a graph with relations and*
- *$U_\ell = \{u_1, \dots, u_\ell\} \subseteq V(G_\ell)$ is a set of ℓ distinguished vertices.*

Moreover, there is a non-negative integer b that depends only on $s_{\text{top}}, r_{\text{top}}$ such that the following holds for every $\ell \geq 1$:

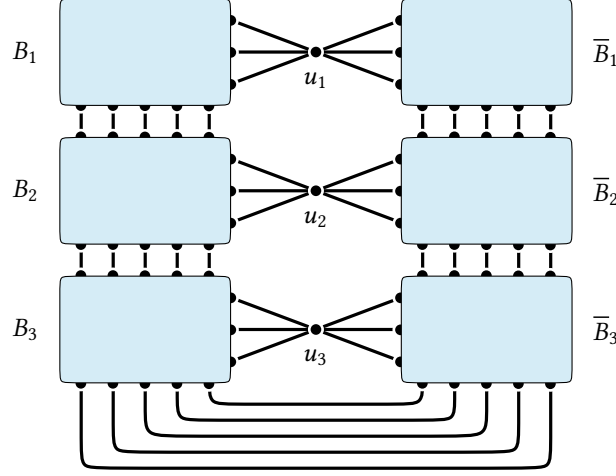


Figure 10.2: An illustration of an A -manager from Definition 10.3.1. Here shown for three distinguished vertices, $\ell = 3$, and with $A \subseteq \{\sigma_i, \rho_i \mid i \in [0 \dots 3]\}$.

- The vertices from $V(G_\ell) \setminus U_\ell$ can be partitioned into 2ℓ vertex-disjoint sets $B_1, \dots, B_\ell$ and $\bar{B}_1, \dots, \bar{B}_\ell$ called blocks, such that
 - $|B_i| \leq b$ and $|\bar{B}_i| \leq b$ for all $i \in [1 \dots \ell]$,
 - $N(u_i) \subseteq B_i \cup \bar{B}_i$ for all $i \in [1 \dots \ell]$,
 - there are edges only between B_i and B_{i+1} , between $\bar{B}_i$ and $\bar{B}_{i+1}$, for each $i \in [1 \dots \ell - 1]$, and between B_ℓ and $\bar{B}_\ell$, and
 - for each relation R of G_ℓ , there is some $i \in [1 \dots \ell]$ such that either $\text{scp}(R) \subseteq N[B_i]$ or $\text{scp}(R) \subseteq N[\bar{B}_i]$.
- Each $x \in A^\ell \subseteq \mathbb{A}^\ell$ is managed in the sense that there is a unique (σ, ρ) -set S_x of G_ℓ such that for all $i \in [1 \dots \ell]$:
 - If $x[i] = \sigma_s$, then $u_i \in S_x$. Moreover, u_i has exactly s neighbors in $B_i \cap S_x$ and exactly $s_{\text{top}} - s$ neighbors in $\bar{B}_i \cap S_x$.
 - If $x[i] = \rho_r$, then $u_i \notin S_x$. Moreover, u_i has exactly r neighbors in $B_i \cap S_x$ and exactly $r_{\text{top}} - r$ neighbors in $\bar{B}_i \cap S_x$.

We refer to G_ℓ as the A -manager of rank ℓ (with distinguished vertices U_ℓ).

It turns out that these managers cannot always be constructed, and so it might be the case that depending on (σ, ρ) there are only A -managers for a restricted set of states A from $\mathbb{A}$.

For each case mentioned in Definition 8.1.4, we define a suitable subset $A \subseteq \mathbb{A}$ with $|A| = c_{\sigma, \rho}$ and then construct the respective A -manager. To handle the case of (simple) cofinite sets, we also ensure that the supported set of states A is *closed under the inverse*. This means that whenever A supports a state σ_i or ρ_i , then it also supports $\text{inv}(\sigma_i) = \sigma_{s_{\text{top}} - i}$ or $\text{inv}(\rho_i) = \rho_{r_{\text{top}} - i}$, respectively.

Definition 10.1.6 (Inverse of a State). Consider two states $\sigma_s \in \mathbb{S}$ and $\rho_r \in \mathbb{R}$. We define

- the inverse of σ_s with respect to σ as the state $\text{inv}^\sigma(\sigma_s) = \sigma_{s_{\text{top}} - s}$, and
- the inverse of ρ_r with respect to ρ as the state $\text{inv}^\rho(\rho_r) = \rho_{r_{\text{top}} - r}$.

We set $\text{inv}^\sigma(\rho_r) = \rho_r$ and $\text{inv}^\rho(\sigma_s) = \sigma_s$.

For all states $\alpha_a \in \mathbb{A}$, we define the inverse of α_a with respect to σ, ρ as the state $\text{inv}(\alpha_a) = \text{inv}^\sigma(\text{inv}^\rho(\alpha_a))$.

Intuitively, the property that A is closed under the inverse, allows us to flip the manager such that the left and the right block change their roles.

With these preliminaries, we can state the main result regarding managers. Formally, we prove Main Lemma 10.3.7 in Section 10.3.

Main Lemma 10.3.7 (Existence of Managers). Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite or cofinite with $\rho \neq \{0\}$.

1. There is an A -manager with $|A| = r_{\text{top}} + 1$.
2. If $s_{\text{top}} \geq r_{\text{top}} \geq 1$, then there is an A -manager with $|A| = s_{\text{top}} + 1$.
3. If (σ, ρ) is 2-structured, but not m -structured for any $m \geq 3$, and s_{top} and $r_{\text{top}} \geq 1$ are even, then there is an A -manager with $|A| = (s_{\text{top}} + r_{\text{top}})/2 + 2$.
4. If (σ, ρ) is not m -structured for any $m \geq 2$, then there is an A -manager with $|A| = s_{\text{top}} + r_{\text{top}} + 2$.

Moreover, each A -manager is such that A is closed under the inverse with respect to σ, ρ .

We emphasize once more that the bounds for the managers precisely coincide with the values $c_{\sigma, \rho}$ as stated in Definition 8.1.4 which determine the “quality” of our lower bounds. Hence, these managers are the key ingredients to obtain *matching* lower bounds.

As a next step, we show how to use these managers to provide the first intermediate lower bound.

10.1.3 Lower Bound for the Intermediate Problem

We first consider the decision problem of (σ, ρ) -DOMSET with relations, and assume for now that the sets σ and ρ are single-element sets. Afterward, we briefly comment on the extension to arbitrary finite and simple cofinite sets, and the changes for the counting version.

Before we give the construction, we formally define the problem (σ, ρ) -DOMSET^{Rel} for which we want to prove the intermediate lower bound.

Definition 10.1.7 ((σ, ρ) -DOMSET^{Rel}). For each set $\mathcal{R}$ of relations, The problem (σ, ρ) -DOMSET^{Rel} takes as input a graph with relations $G = (V, E, \mathcal{C})$ where the relational constraints in $\mathcal{C}$ are from $\mathcal{R}$ only, and the task is to decide whether there is a (σ, ρ) -set of G . We use $|G|$ as the size of the input.

The counting problem (σ, ρ) -#DOMSET^{Rel} is defined analogously.

If we do not restrict the set $\mathcal{R}$ to specific relations, we write *Rel* for the set of all computable relations.

To obtain strong lower bounds of the form $(|\mathbb{A}| - \varepsilon)^{\text{pw}} \cdot n^{\mathcal{O}(1)}$, we reduce from q -CSP- B as introduced in Section 4.1.2.

Definition 4.1.6 (q -CSP- B [94]). *Fixed two positive numbers $q \geq 2$ and $B \geq 2$.*

A q -CSP- B instance consists of a set X of n variables which take values in $[1..B]$ and a set $\mathcal{C}$ of q -constraints on X . A q -constraint C is defined by an ordered tuple $\text{scp}(C)$ of q variables of X , and a set $R(C) \subseteq [1..B]^q$ of satisfying assignments for C .

The task of q -CSP- B is to decide if such a given q -CSP- B instance is satisfiable, that is, decide if there is an assignment $\pi: X \rightarrow [1..B]$ which satisfies all constraints $C \in \mathcal{C}$ where a constraint $C \in \mathcal{C}$ is satisfied if π restricted to $\text{scp}(C)$ appears in $R(C)$.

The parity of a constraint is the parity of the 1-norm of the accepted assignments restricted to the scope. The parity can be even, odd or undefined if not all accepted assignment restricted to the scope have the same 1-norm.

Intuitively, we construct a graph with relations where we use the states of the vertices, that is, the selection status and the number of selected neighbors, to encode the assignment of the variables, and we use the relations to encode the constraints.

To achieve the highest number of possible states, we use managers in the construction. Then, the states of the distinguished vertices of these managers encode the assignment. More precisely, for each distinguished vertex we use the selected vertices to the left to encode the assignment, whereas the selected vertices to the right ensure that the information vertex has a feasible number of selected neighbors overall.

Ideally, we would like to allow all possible states from $\mathbb{A}$ to encode the highest number of variable assignments. However, as this is generally not possible, we use the managers from the previous step. That is, with an A -manager, we can achieve $|A|$ states. When we reduce from the CSP where $B = |A|$ we can simulate all assignments by the distinguished vertices.

Construction. To construct the graph, we assume that we are given a q -CSP- B instance ϕ along with a path decomposition of the primal graph of ϕ . Assume that the path decomposition has m bags and there is a one-to-one correspondence between bags and constraints. We refer to Figure 10.3 for an illustration of the following construction.

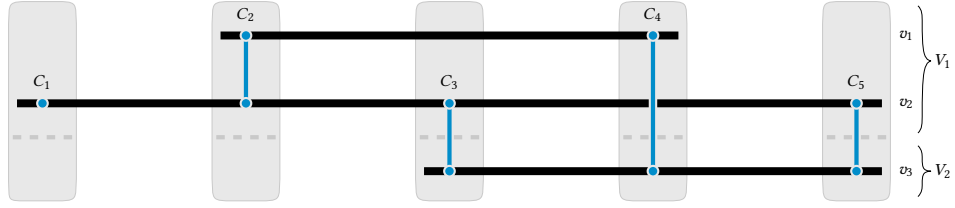
Therefore, we also have m constraints.

Based on the path decomposition of ϕ , we construct a grid-like graph with m columns, each corresponding to one node of the path decomposition of the primal graph, and for each bag β , we create $|\beta|$ rows each corresponding to one variable in β . We denote these vertices as t_i^j for a node j and a variable v in bag $\beta(j)$ of node j . Moreover, we add all vertices of one column, that is, the vertices $t_\star^j$ to a common A -manager.

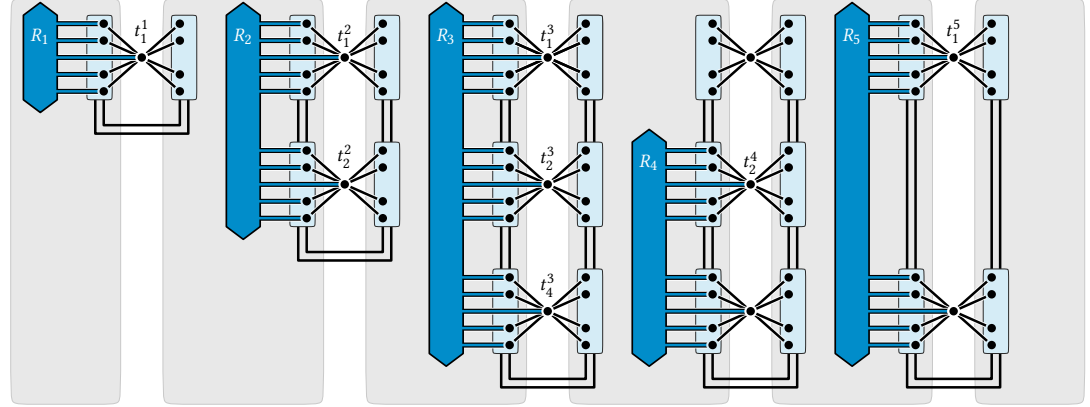
Additionally, for each bag, that is, for each constraint C_j of ϕ , we construct a relation R_j which checks whether the assignment encoded by the vertices $t_\star^j$ satisfies the clause C_j . For this check, the relation observes the *left block* of the manager and infers the state of the distinguished vertex from the selected vertices.

So far, the graph consists of m managers that are *not* connected. Therefore, it remains to connect the managers such that the assignments are encoded consistently, that is, for each variable, the corresponding vertices encode the same assignment.

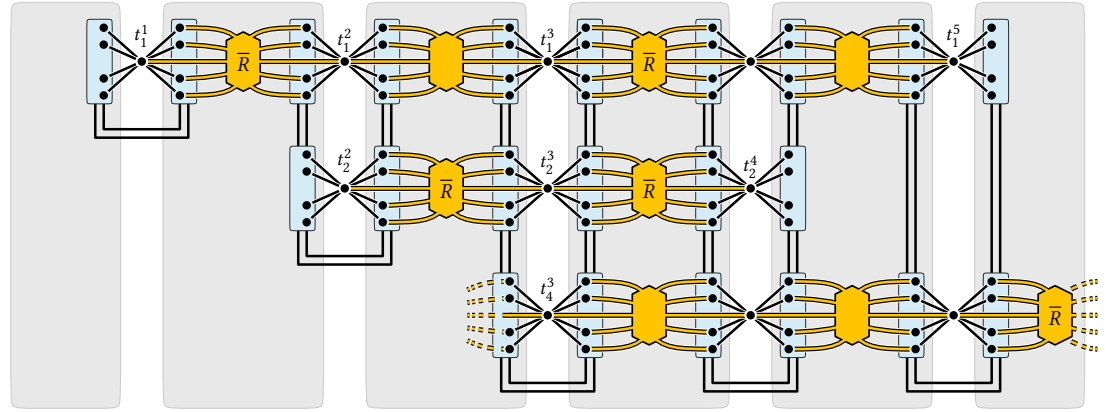
For the first idea, we think of σ and ρ as single-element sets. For example, let $\sigma = \{4\}$ and $\rho = \{3\}$. Whenever the information vertex t_v^j is selected and has $\ell \in [0..s_{\text{top}}]$ selected



(a) The underlying formula together with a path decomposition. The horizontal lines represent the three variables and the vertical lines the five constraints.



(b) The graph of the (σ, ρ) -DOMSET^{Rel}-instance. Here we show the managers and the relations R_j that check if the assignment for the variables in bag $\beta(j)$ satisfy the j th constraint.



(c) The graph of the (σ, ρ) -DOMSET^{Rel}-instance. Here we show the managers and the relations $\bar{R}$ that ensures that the state of the left block inverts the state of the right block.

Figure 10.3: The idea of the construction for the lower bound of (σ, ρ) -DOMSET^{Rel} given a q -CSP- B formula with three variables, depicted as rows, and five constraints, depicted as columns.

neighbors to the left, it must have $s_{\text{top}} - \ell = 4 - \ell$ selected neighbors on the right because of the constraints imposed by σ , that is, in our case a selected vertex needs a total of 4 selected neighbors.

For each variable we create a relation that ensures that the next information vertex t_v^{j+1} receives ℓ selected neighbors from the left whenever t_v^j has $4 - \ell$ selected neighbors to its right. We add such a relation $\bar{R}$ for each variable v and for each bag $\beta(j)$ where v also appears in bag $\beta(j+1)$.

By this definition, all information vertices corresponding to the same variable, receive the same number of neighbors from the left. Hence, they can encode the same assignment. The same arguments work for the case when t_v^j is not selected, but then t_v^j must have $r_{\text{top}} - \ell = 3 - \ell$ neighbors to the right if it has ℓ neighbors to the left. As the behavior of the relation depends on the information vertices, the relations are additionally connected to the information vertices and not only their neighbors in the blocks.

When the sets σ and ρ contain more than one element, the situation is more complicated. Assume as illustrative example that $\sigma = [0..4]$. If vertex t_v^j is selected and has $\ell \in [0..s_{\text{top}}]$ selected neighbors in the left block, it might receive any number of selected neighbors from the right block, that is, at most $s_{\text{top}} - \ell = 4 - \ell$ neighbors. Say, the vertex receives $\ell' \leq 4 - \ell$ neighbors from the right. Then, by our constraint $\bar{R}$, vertex t_v^{j+1} must receive $4 - \ell'$ neighbors from the left. By the previous observation, vertex t_v^{j+1} receives $4 - \ell' \geq \ell$ neighbors from the left; the assignment of the variable increased.

Instead of adjusting the construction to avoid this change, we reduce from a different version of q -CSP- B which takes care of this already. Formally, we base our intermediate lower bound construction on Conjecture 4.1.11.

Conjecture 4.1.11 (MONO-pw-CSPH). *For all $B \geq 2$ and $\varepsilon > 0$, there exists a q such that no algorithm can, for a q -CSP- B instance ϕ that is given with*

- *a partition $V_1 \dot{\cup} V_2$ of the variables,*
- *a nice path decomposition of the primal graph of ϕ of width w*
- *where every bag contains at most $\mathcal{O}(B \cdot \log|V_1|)$ variables from V_2 ,*
- *an injective mapping c from the constraints of ϕ to the bags such that the bag contains the variables of the respective constraint,*
- *the promise that every satisfying multi-assignment that is consistent on V_2 is also consistent on V_1 ,*

decide if there is a monotone satisfying multi-assignment that is consistent on V_2 in time

$$(B - \varepsilon)^w \cdot |\phi|^{\mathcal{O}(1)}.$$

To ensure that the variables in V_2 receive a consistent assignment, we simply connect the vertex corresponding to the last occurrence with the vertex from the first occurrence. As the assignment might only increase monotonically by our previous observation, and this now also

holds for the transition from the last to the first occurrence, the assignment cannot change at all and must be consistent. Hence, we can immediately handle the case of arbitrary finite sets.

We also extend this intermediate result to *simple cofinite* sets. A set is simple cofinite if it contains, for some $k \in \mathbb{N}$, precisely the elements greater than or equal to k . When we use these simple cofinite sets, we can use essentially the same approach except that we have to invert the states. Intuitively, the simple cofinite sets behave opposite to their complement, that is, they behave opposite to sets of the form $[0 \dots k]$. Hence, the assignment might *decrease* monotonically and by inverting the states, they might increase again.

Bound on the Pathwidth. Observe that the information vertices of one column cut the graph in two halves (except for a constant number of edges from the managers and the small number of variables from V_2). As the number of vertices in one column corresponds to the number of variables in the respective bag of ϕ , the pathwidth of the graph can be bounded by the pathwidth of ϕ plus some constant which takes care of the relations and other vertices. For this we crucially need that there are not too many vertices from V_2 in each bag, as this might significantly increase pathwidth otherwise.

Moreover, when we cut the graph at the information vertices of one column, we get a direct correspondence to our algorithmic results. Indeed, the states which the information vertices might obtain in a solution (which are specified by their selection status and the number of selected vertices to the left) precisely correspond to the state-vectors in the language compatible with the graph.

From an A -manager we then obtain a lower bound of the form $(|A| - \varepsilon)^{\text{pw}} \cdot n^{\mathcal{O}(1)}$. Thus, we treat the manager as an input, and prove the lower bound based on it. Our bounds are defined such that they hold even if we restrict the problem to instances in which the arity of relations is bounded by some constant.

Main Lemma 10.4.3 (Lower Bound for (σ, ρ) -DOMSET^{Rel}). *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite or simple cofinite. Suppose there is an $A \subseteq \mathbb{A}$ that is closed under the inverse with respect to σ, ρ such that there is an A -manager.*

For all $\varepsilon > 0$, there is a constant d such that (σ, ρ) -DOMSET^{Rel} on instances of size n and arity at most d cannot be solved in time $(|A| - \varepsilon)^{p + \mathcal{O}(1)} \cdot n^{\mathcal{O}(1)}$, even if the input is given with a path decomposition of width p , unless MONO-pw-CSPH fails.

By a careful design of our graph, and additional relations that we add to the managers and providers, we can ensure that the same reduction also works for the counting version.

Main Lemma 10.4.4. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite or simple cofinite. Suppose there is an $A \subseteq \mathbb{A}$ that is closed under the inverse with respect to σ, ρ such that there is an A -manager.*

For all $\varepsilon > 0$, there is a constant d such that (σ, ρ) -#DOMSET^{Rel} on instances of size n and arity at most d cannot be solved in time $(|A| - \varepsilon)^{k + \mathcal{O}(1)} \cdot n^{\mathcal{O}(1)}$, even if the input is given with a path decomposition of width k , unless MONO-pw-#CSPH fails.

We note that both intermediate lower bounds require that σ and ρ are finite or simple cofinite sets. For the counting version, we generalize this result to arbitrary cofinite sets in a separate step that follows next.

We provide the formal proof of Main Lemmas 10.4.3 and 10.4.4 in Section 10.4.

Before we move to the next step, we briefly summarize the previous results. Assume that σ and ρ are finite or simple cofinite sets with $\rho \neq \{0\}$. With Definition 8.1.4, we observe that Main Lemma 10.3.7 together with Main Lemma 10.4.3 ensure that there is no $(c_{\sigma,\rho} - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$ time algorithm for (σ, ρ) -DOMSET^{Rel} on graphs G that are given with a path decomposition of width w , unless MONO-pw-CSPH (or SETH) fails.

10.1.4 Counting Version: Extension to Cofinite Sets

We handle general cofinite sets by a reduction from simple cofinite sets. In particular, suppose that σ is cofinite; then, we reduce from the case where $\sigma' = \{s_{\text{top}}, s_{\text{top}} + 1, \dots\}$. Intuitively, the set σ' imposes a constraint of the form $\geq s_{\text{top}}$ selected neighbors. The set σ only imposes a restriction that is much weaker. To restrict the vertices with set σ to behave like vertices with set σ' , we provide a reduction that uses interpolation techniques, and therefore, works only for the problem of *counting* (σ, ρ) -sets.

For now, we assume that ρ is a finite or simple cofinite set. The analogous approach works for the case when ρ is cofinite but not simple cofinite.

Given a graph with relations G , the reduction creates a new graph G' by attaching to each vertex v a certain gadget with the following properties:

- If v is unselected, then the gadget has a unique extension, which does not give any additional selected neighbors to v .
- If v is selected, then every extension of the gadget provides at most s_{top} new selected neighbors to v .
- If v is selected, then the gadget has d_i extensions that provide i new selected neighbors to v , where $i \in [0 \dots s_{\text{top}}]$.

Given the numbers $d_0, \dots, d_{s_{\text{top}}}$, it is not very difficult to use the relations to construct a gadget with exactly these properties and number of extensions.

Let us consider $\sigma = \{1, 3, 4, 6, 7, \dots\}$ as an illustrative example, and suppose for simplicity that we attach such a gadget only to a single vertex v . Note that we have $s_{\text{top}} = 6$. Consider now partial solutions of the original graph G where every vertex satisfies the requirements, except potentially v . Suppose that v is selected and has 0 selected neighbors in such a partial solution; it needs 1, 3, 4, or at least 6 further selected neighbors to satisfy the σ -constraint. Therefore, the gadget has exactly $d_1 + d_3 + d_4 + d_6$ extensions where the degree of v becomes a member of σ . Similarly, if v already has one selected neighbor in the partial solution of G , then it needs 0, 2, 3, or *at least* 5 further selected neighbors, and thus, the gadget has exactly $d_0 + d_2 + d_3 + d_5 + d_6$ extensions where v satisfies the degree condition, and so on.

We would like to build a gadget where the integers d_i are chosen in such a way that the gadget has exactly one (or some other constant) extension if v already has at least six selected neighbors, and 0 extensions if v has less than six selected neighbors. If we can build such a

gadget, then we can effectively force that vertex v has at least six selected neighbors. Based on the discussion above, the values $d_\star$ have to be chosen such that they satisfy

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

We observe that the matrix has a form that guarantees that it is non-singular: every element on or below the codiagonal is 1, while every element one row above the codiagonal is 0. Therefore, the system of equations has a solution, and we can construct the appropriate gadget. The catch is, however, that some of the values $d_\star$ could be negative, which makes it impossible to have a gadget with exactly that many extensions.

We can solve this issue in the following way. If d_i is negative, then we design the gadget to have $2^x \cdot |d_i|$ extensions, where x is some parameter of the construction. If we attach such gadgets to every vertex, then we observe that the number of solutions is a polynomial $P(y)$ where we set $y = 2^x$. Varying the values for x , we can use interpolation techniques to recover the coefficients of this polynomial P , and thus, the entire polynomial $P(y)$. Then, we evaluate P at $y = -1$, which then simulates exactly $-1 \cdot |d_i| = d_i$ extensions.

Formally, we prove the reduction from Main Lemma 10.5.8.

Main Lemma 10.5.8. *Consider two non-empty sets σ and ρ that are finite or cofinite with $\rho \neq \{0\}$.*

For all sets $\hat{\sigma}$ that are finite with $\sigma = \hat{\sigma}$ or simple cofinite with $\text{top } \sigma = \text{top } \hat{\sigma}$, and for all sets $\hat{\rho}$ that are finite with $\rho = \hat{\rho}$ or simple cofinite with $\text{top } \rho = \text{top } \hat{\rho}$, the following holds:

There is a pa-reduction from $(\hat{\sigma}, \hat{\rho})\text{-}\#\text{DOMSET}^{\text{Rel}}$ to $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{Rel}}$.

We prove Main Lemma 10.5.8 in Section 10.5. Instead of giving a direct reduction, we use intermediate steps with weighted relations and weighted vertices. This allows us to provide smaller reductions that serve a single purpose.

10.1.5 Reducing to One Type of Relations

In the previous steps we explained how to obtain conditional lower bounds for the intermediate problem $(\sigma, \rho)\text{-}\text{DOMSET}^{\text{Rel}}$ and its counting version. In the remaining steps, our goal is to show that these lower bounds transfer to the original problem versions (without the relations), that is, we remove the relations from the graph again.

Suppose there is some set of vertices U whose desired selection status is subject to some relation R . Intuitively, we aim to attach some graph J to the vertices in U such that the (σ, ρ) -constraints of the vertices in J require a selection of vertices in U that satisfies R . If this works out, we say that J is a *realization of R* . Crucially, we wish to be able to add such a realization without invalidating the σ - and ρ -constraints of the vertices in U with respect to

the original graph. Hence, we require that a realization does not add any *selected* neighbors to the vertices in U . See Definition 10.6.2 for the formal definition.

Following a previous approach by Curticapean and Marx [35], we first show how to model arbitrary relations using only “Hamming weight one”-relations and “Equality”-relations, that is, $\text{HW}_{=1}$ - and EQ -relations. In a second step, we prove that we can realize EQ by using $\text{HW}_{=1}$ -relations only.

In summary, we show a parsimonious reduction from $(\sigma, \rho)\text{-DOMSET}^{\text{Rel}}$ to $(\sigma, \rho)\text{-DOMSET}^{\text{HW}_{=1}}$, which denotes the restriction of $(\sigma, \rho)\text{-DOMSET}^{\text{Rel}}$ to instances that use only $\text{HW}_{=1}$ -relations. It is important that the reduction is parsimonious since we use it both in the decision setting and in the counting setting.

Main Lemma 10.6.7. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ with $\rho \neq \{0\}$. For all constants d , there is a parsimonious pathwidth-preserving reduction from $(\sigma, \rho)\text{-DOMSET}^{\text{Rel}}$ on instances of arity at most d to $(\sigma, \rho)\text{-DOMSET}^{\text{HW}_{=1}}$ on instances of arity at most $2^d + 1$.*

We prove Main Lemma 10.6.7 in Section 10.6. Afterward, we consider different cases to realize the remaining $\text{HW}_{=1}$ -relations.

10.1.6 Decision Version: Realizing Relations

For the decision problem, we can realize $\text{HW}_{=1}$ -relations by graphs whose size is bounded by a function in the arity d , and in the values of r_{top} and s_{top} . However, we obtain such realizations only for finite σ and ρ . As we have seen, this is justified by the fact that, for cofinite sets, algorithmic improvements are possible based on representative sets (see Main Theorem III.2).

With the realizations at hand, we provide a pathwidth-preserving reduction from the decision problem with $\text{HW}_{=1}$ -relations to the one without as long as there is some constant upper bound on the arity of relations. Note that our reduction from Main Lemma 10.7.3 holds only if $0 \notin \rho$, that is, if $(\sigma, \rho)\text{-DOMSET}$ is non-trivial.

Main Lemma 10.7.3. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite with $0 \notin \rho$. For all constants d , there is a pathwidth-preserving reduction from $(\sigma, \rho)\text{-DOMSET}^{\text{HW}_{=1}}$ on instances of arity at most d to $(\sigma, \rho)\text{-DOMSET}$.*

We present the proof of Main Lemma 10.7.3 in Section 10.7. Next, we consider the corresponding result for the counting version.

10.1.7 Counting Version: Realizing Relations

For the counting version, we observe that there are pairs (σ, ρ) for which the decision problem can be solved in polynomial time, but the counting version is non-trivial. In particular, all cases with $0 \in \rho$ are trivial for the decision version, but are non-trivial for the counting version if $\rho \neq \{0\}$. Moreover, we not only want to consider finite sets, but we aim to establish the lower bounds also for cofinite sets σ or ρ .

For this larger class of sets σ and ρ that we consider now, it is unlikely that we can realize the relations in all cases directly by attaching some graph gadget to the vertices. Indeed, these approaches would most likely also give lower bounds for the decision problem as well.

This makes the realization of $\text{HW}_{=1}$ much more challenging. In fact, the proof of Main Lemma 10.8.1 splits into a number of cases and sequences of reductions where we use additional techniques to work around the difficulties. We split into the following three cases depending on σ and ρ :

- (A) $\rho \neq \mathbb{N}$,
- (B) $\rho = \mathbb{N}$ and σ is finite, and
- (C) $\rho = \mathbb{N}$ and σ is cofinite.

The treatment of the first two cases is similar, whereas the third case is substantially different. Hence, we explicitly exclude this case for now and consider it separately afterward. Formally, we prove Main Lemma 10.8.1 in Section 10.8.

Main Lemma 10.8.1. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite or cofinite such that (σ, ρ) is non-trivial. Moreover, assume that σ is finite or $\rho \neq \mathbb{N}$.*

For all constants d , there is a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{HW}_{=1}}$ on instances with arity at most d to $(\sigma, \rho)\text{-}\#\text{DOMSET}$.

This mirrors the
Holant problem
for GENFAC.

To reduce $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{HW}_{=1}}$ to $(\sigma, \rho)\text{-}\#\text{DOMSET}$, it is convenient to introduce a natural generalization of the problem for which it is not necessary that every vertex has the same constraint (σ, ρ) . Instead, we have an additional set $\mathcal{P}$ of pairs and the input contains a mapping specifying for every vertex if it is constrained by (σ, ρ) or by some other pair in $\mathcal{P}$. For a fixed set $\mathcal{P}$ of pairs, we write $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$ for this extension. As the first step of the proof, we show that if

- $(\emptyset, \{1\}) \in \mathcal{P}$,
- $(\emptyset, \{0, 1\}) \in \mathcal{P}$,
- $(\emptyset, \mathbb{N}_{\geq 1}) \in \mathcal{P}$, or
- $(\{0\}, \mathbb{N}) \in \mathcal{P}$,

then we can reduce $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{HW}_{=1}}$ to $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$ in a pathwidth-preserving way because $\text{HW}_{=1}$ -relations can be simulated using one of the four pairs listed above.

With this observation, the main part of the proof is to show that, for every non-trivial pair (σ, ρ) where $\rho \neq \mathbb{N}$, that is, Case (A), or where $\rho = \mathbb{N}$ and σ is finite, that is, Case (B), we can simulate one of the four pairs with the $(\sigma, \rho)\text{-}\#\text{DOMSET}$ problem.

Toward this goal, we simulate some number of other pairs. A key intermediate goal is to “force a vertex to be unselected”, that is, to simulate an $(\emptyset, \rho)$ -vertex. Similarly, another important goal is to “force a vertex to be selected” by simulating a $(\sigma, \emptyset)$ -vertex. Now, consider the following construction:

- introduce $r_{\text{top}} - 1$ cliques, each consisting of $s_{\min} + 1$ vertices of type $(\sigma, \emptyset)$;
- introduce one vertex v of type $(\emptyset, \rho)$, and make it adjacent to one vertex in each clique.

In this gadget, the vertex v can be considered as an always unselected vertex that already has $r_{\text{top}} - 1$ extra selected neighbors. Phrased differently, we can consider v as a $(\emptyset, \rho - r_{\text{top}} + 1)$ -vertex, that is, the set ρ is “shifted down” by $r_{\text{top}} - 1$.

Crucially, this construction only works if $r_{\text{top}} \neq 0$, which is satisfied if $\rho \neq \{0\}$ and $\rho \neq \mathbb{N}$. As we consider non-trivial pairs (σ, ρ) only, we can assume $\rho \neq \{0\}$. Then, when we focus on Case (A), we additionally have $\rho \neq \mathbb{N}$. In this case, depending on whether ρ is cofinite, finite with $r_{\text{top}} - 1 \in \rho$ or finite with $r_{\text{top}} - 1 \notin \rho$, the vertex v effectively behaves as an $(\emptyset, \mathbb{N}_{\geq 1})$ -, $(\emptyset, \{0, 1\})$ -, or $(\emptyset, \{1\})$ -vertex, respectively.

When $\rho = \mathbb{N}$ and σ is finite, the reasoning is similar to Case (A) but arguably even slightly simpler. Indeed, for the treatment of Case (B) it suffices to force selected vertices, that is, it suffices to model $(\sigma, \emptyset)$ -vertices, in order to realize $(\{0\}, \mathbb{N})$ -vertices.

Let us now highlight some technical ideas behind the simulation of $(\emptyset, \rho)$ - and $(\sigma, \emptyset)$ -vertices. The following notation is convenient for the description. When $\mathcal{G}$ is a gadget with a single portal vertex, we write $\text{ext}_{\mathcal{G}}(\rho_i)$ for the number of partial solutions S of $\mathcal{G}$ where vertex v is unselected and receives i selected neighbors in S , that is, the number of extensions for state ρ_i . The definition of $\text{ext}_{\mathcal{G}}(\sigma_i)$ is analogous with $v \in S$.

Step 1. Let us assume first that σ and ρ are finite. We construct a gadget $\mathcal{G}$ with a single portal that has a single neighbor in the gadget $\mathcal{G}$. Moreover, we design the gadget such that $\text{ext}_{\mathcal{G}}(\rho_0), \text{ext}_{\mathcal{G}}(\rho_1) \geq 1$, and where $\text{ext}_{\mathcal{G}}(\rho_0)$ and $\text{ext}_{\mathcal{G}}(\sigma_0)$ are both at least one, but different. In other words, the choice of (not) selecting the portal v influences the number of extensions in the gadget where the neighbors of the portal are not selected.

Step 2. We use this gadget, to simulate a vertex $(\emptyset, \{0\})$, that is, a vertex that must be unselected and should not get any selected neighbors. We can use such a vertex to force any number of other vertices to be unselected.

Consider a graph G with a $(\emptyset, \{0\})$ -vertex v . Let a_d denote the number of partial solutions in G where we treat v as a (σ, ρ) -vertex that is unselected and has exactly d selected neighbors. Similarly, we denote by b_d the analogous number where v is selected. Hence, the (unknown) value of a_0 corresponds to the number of solutions of G .

Next, we create a new graph G' where we introduce x copies of $\mathcal{G}$ and identifying their portal vertices with the vertex v . As vertex v needs a valid number of neighbors, the number of solutions for G' is

$$\underbrace{\sum_{d=0}^{r_{\text{top}}} a_d \cdot \sum_{i+d \in \rho} \binom{x}{i} \cdot \text{ext}_{\mathcal{G}}(\rho_1)^i \cdot \text{ext}_{\mathcal{G}}(\rho_0)^{x-i}}_{A=} + \underbrace{\sum_{d=0}^{s_{\text{top}}} b_d \cdot \sum_{i+d \in \rho} \binom{x}{i} \cdot \text{ext}_{\mathcal{G}}(\sigma_1)^i \cdot \text{ext}_{\mathcal{G}}(\sigma_0)^{x-i}}_{B=}$$

The important observation here is that $\text{ext}_{\mathcal{G}}(\rho_0) \neq \text{ext}_{\mathcal{G}}(\sigma_0)$ implies that the two terms A and B above have very different orders of magnitude. Assume that $\text{ext}_{\mathcal{G}}(\rho_0) < \text{ext}_{\mathcal{G}}(\sigma_0)$ as the other case is similar. Then, $2^{|V(G)|} \cdot \text{ext}_{\mathcal{G}}(\rho_0)^x$ is a rough upper bound for A , while B is divisible by $\text{ext}_{\mathcal{G}}(\sigma_0)^{x-r_{\text{top}}}$, which is much larger for sufficiently large values of x . Therefore, if we compute $(A + B) \bmod \text{ext}_{\mathcal{G}}(\sigma_0)^{x-r_{\text{top}}}$, then B vanishes and we can recover A .

For simplicity, we assume here that $\text{ext}_{\mathcal{G}}(\rho_1) \neq 0$. We treat the other case separately.

When we divide A by $\text{ext}_{\mathcal{G}}(\rho_0)^x \neq 0$, we obtain a polynomial in x of degree r_{top} . Moreover, we can recover the coefficients with polynomial interpolation. Observe that only $d = 0$ contributes a term $x^{r_{\text{top}}}$, and thus, from the coefficient of $x^{r_{\text{top}}}$ we can recover the value of a_0 . This counts the number of partial solutions where v is not selected and has no selected neighbors, that is, it simulates an $(\emptyset, \{0\})$ -vertex, as we wanted.

Step 3. Once we can express a single $(\emptyset, \{0\})$ -vertex, we make it adjacent to all vertices that should be unselected. This forces these vertices to behave as $(\emptyset, \rho)$ -vertices as the $(\emptyset, \{0\})$ -vertex cannot be selected. A further interpolation argument of similar flavor allows us to simulate an arbitrary number of $(\sigma, \emptyset)$ -vertices.

Cofinite Sets. When at least one of σ and ρ is cofinite, then there are multiple difficulties with the previous approach. For example, suppose that ρ is cofinite and consider the term A above. As ρ is not finite, we cannot write A as the sum of a *constant* number of binomial terms. Instead, we can write A as all possible extensions to $\mathcal{G}$ minus those extensions that give the wrong number of selected neighbors to v . That is, when δ denotes the degree of v in G , we have

$$A := \sum_{d=0}^{\delta} a_d \cdot \left((\text{ext}_{\mathcal{G}}(\rho_0) + \text{ext}_{\mathcal{G}}(\rho_1))^x - \sum_{i+d \neq \rho} \binom{x}{i} \cdot \text{ext}_{\mathcal{G}}(\rho_1)^i \cdot \text{ext}_{\mathcal{G}}(\rho_0)^{x-i} \right).$$

In this case, if we divide A by $\text{ext}_{\mathcal{G}}(\rho_0)^x$, then this expression is a combination of an exponential term and some polynomials.

Phrased differently, in the cofinite cases, we have to deal with functions of the form $f(x) = c^x - p(x)$ where $p(x)$ is a polynomial. A key technique to handle such expressions is to compute

$$f(x+1) - c \cdot f(x) = (c^{x+1} - p(x+1)) - c \cdot (c^x - p(x)) = p(x+1) - c \cdot p(x),$$

which is a polynomial of the same degree as $p(x)$, and hence, much easier to work with.

We present the formal proof of Main Lemma 10.8.1 in Section 10.8. As a last intermediate step, we realize relations when $\rho = \mathbb{N}$ and σ is cofinite.

10.1.8 Special Case: σ is Cofinite and $\rho = \mathbb{N}$

For the counting version it remains to realize $\text{HW}_{=1}$ when σ is cofinite and $\rho = \mathbb{N}$. In this case, we need a different approach to realize relations. Before we state the formal reduction in Main Lemma 10.9.1 we give some intuition why this particular case requires a different treatment.

Consider some vertex v that is subject to a relation R . In order to model the relation, we need to distinguish between solutions in which v is selected, and those in which v is not selected. Phrased differently, depending on the selection status of v , the number of partial solutions extending the respective state must be different.

Now suppose that $\sigma = \mathbb{N}_{\geq 1}$ as this actually turns out to be the key case. Then, vertex v is a $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertex and can be unselected with an arbitrary number of selected neighbors,

or it can be selected and requires at least one more selected neighbor. By the assumption that $\rho = \mathbb{N}$, the unselected neighbors of v behave the same independently of the selection status of v . Also, in general, if v has only unselected neighbors, then their number does not affect the selection status of v . Therefore, to distinguish whether v is selected or not, vertex v needs at least one *selected* neighbor.

However, as soon as the $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertex v has at least one selected neighbor, then it has a feasible number of selected neighbors independently of its selection status and independently of how many additional selected neighbors it might obtain. So, at this point, vertex v is entirely unaffected by the selection status of its remaining neighbors and acts essentially as a $(\mathbb{N}, \mathbb{N})$ -vertex.

This is why, while we are able to observe the selection status of v , and in this sense model the relation R by attaching some gadgets and interpolating, we do not manage to simultaneously recover the situation where v behaves like a $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertex with respect to the original graph.

To handle this situation, we introduce another, different concept of “ (σ, ρ) -sets with relations”. Now, whenever a vertex v is subject to some relation, we treat v as a $(\mathbb{N}, \mathbb{N})$ -vertex, that is, vertex v does not impose any constraints on the solutions other than that it needs to satisfy the relation. So, we have graphs with two types of vertices: (1) (σ, ρ) -vertices that are not subject to any relation, and (2) $(\mathbb{N}, \mathbb{N})$ -vertices that are each subject to at least one relation. It turns out that the required lower bounds also hold for this modified problem with relations.

This vaguely reminds of quantum effects where measuring a particle can change its behavior or obfuscate some other measurable property. A common example is Heisenberg’s Uncertainty Principle.

Main Lemma 10.9.1. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are non-trivial where σ is cofinite and $\rho = \mathbb{N}$. For all constants d , there is a pathwidth-preserving reduction from (σ, ρ) -#DOMSET^{Rel} on instances with arity at most d to (σ, ρ) -#DOMSET.*

We formally prove Main Lemma 10.9.1 in multiple steps in Section 10.9.

As a first result, we establish the new concept of (σ, ρ) -sets with relations. We indicate this new interpretation of relations by $\ddagger$. The idea is to construct a (weighted) gadget such that we can copy a vertex v in the following sense. Whenever vertex v is unselected, the copy v' of v is also unselected, and when v is selected, then the gadget forces the copy v' to be selected as well and provides no neighbors to both vertices. To construct this gadget, we heavily rely on weighted relations such that partial solutions can cancel-out each other.

Then, in the following steps, we have to repeat almost all intermediate steps from the general case of the counting version. That is, we first remove the weights; this follows ideas from Section 10.5. Second, we replace arbitrary relations by $\text{HW}_{=1}$ -relations under the new interpretation; this follows the approach from Section 10.6. Finally, similar to the general results for the counting version, we realize $\text{HW}_{=1}$.

This concludes the last step of the intermediate results.

10.1.9 Combining the Results

With the previous results we have almost everything ready to prove the conditional lower bounds for (σ, ρ) -DOMSET and (σ, ρ) -#DOMSET. As the “quality” of the lower bounds is determined by the values $c_{\sigma, \rho}$, we restate their definition here for convenience.

Definition 8.1.4. *Consider two finite or cofinite sets σ and ρ of non-negative integers. We define $c_{\sigma, \rho}$ by setting*

- $c_{\sigma,\rho} := s_{\text{top}} + r_{\text{top}} + 2$ if (σ, ρ) is not m -structured for any $m \geq 2$,
- $c_{\sigma,\rho} := \max\{s_{\text{top}}, r_{\text{top}}\} + 2$ if (σ, ρ) is 2-structured, but not m -structured for any $m \geq 3$, and $s_{\text{top}} = r_{\text{top}}$ is even, and
- $c_{\sigma,\rho} := \max\{s_{\text{top}}, r_{\text{top}}\} + 1$, otherwise.

From the existence of managers in Main Lemma 10.3.7 and Definition 8.1.4, we immediately obtain Corollary 10.1.8.

Corollary 10.1.8. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite or cofinite with $\rho \neq \{0\}$.*

There is a set $A \subseteq \mathbb{A}$ with $|A| = c_{\sigma,\rho}$ that is closed under the inverse with respect to σ, ρ such that there is an A -manager.

Proof. By Definition 8.1.4, we have to consider three different cases.

- Suppose (σ, ρ) is not m -structured for any $m \geq 2$.

By Case 4 of Main Lemma 10.3.7, there is an A -manager with $|A| = s_{\text{top}} + r_{\text{top}} + 2$. As $c_{\sigma,\rho} = |A|$ in this case, the claim follows.

- Suppose (σ, ρ) is 2-structured, but not m -structured for any $m \geq 3$, and $s_{\text{top}} = r_{\text{top}}$ is even.

As ρ is 2-structured, we know that ρ is finite but $\rho \neq \{0\}$, by assumption. Hence, we directly get $r_{\text{top}} \geq 1$. Then, by Case 3 in Main Lemma 10.3.7, there is an A -manager with $|A| = (s_{\text{top}} + r_{\text{top}})/2 + 2 = \max\{s_{\text{top}}, r_{\text{top}}\} + 2$. As $c_{\sigma,\rho} = \max\{s_{\text{top}}, r_{\text{top}}\} + 2$ in this case, the claim follows.

- Finally, suppose that (σ, ρ) is m -structured for an $m \geq 3$, or 2-structured with $s_{\text{top}} \neq r_{\text{top}}$, or 2-structured with $s_{\text{top}} = r_{\text{top}}$ being odd.

If $r_{\text{top}} \geq s_{\text{top}}$, then we can use the A -manager from Case 1 in Main Lemma 10.3.7 with $|A| = r_{\text{top}} + 1$ as we have $c_{\sigma,\rho} = \max\{s_{\text{top}}, r_{\text{top}}\} + 1$.

Hence, assume that $s_{\text{top}} \geq r_{\text{top}}$. We claim that $r_{\text{top}} \geq 1$. Whenever $r_{\text{top}} = 0$, we either have $\rho = \{0\}$ or $\rho = \mathbb{N}$. The first case is not possible by assumption. The second case implies that ρ is 1-structured, but not m -structured for any $m \geq 2$. This immediately contradicts our assumption. Thus, we can use the A -manager from Case 2 in Main Lemma 10.3.7 with $|A| = s_{\text{top}} + 1$. As we have $c_{\sigma,\rho} = \max\{s_{\text{top}}, r_{\text{top}}\} + 1$, this proves the claim.

As all A -manager from Main Lemma 10.3.7 are such that A is closed under the inverse with respect to σ and ρ , this concludes the proof. $\square$

As a last ingredient, we recall the relation of SETH and MONO-pw-CSPH from Main Theorem I.3.

Main Theorem I.3. *If pw-SETH holds, then MONO-pw-CSPH holds.*

Moreover, if pw-#SETH holds, then MONO-pw-#CSPH holds.

Both results hold even if all constraints have a fixed parity.

Now we have everything ready to state the formal proofs.

Decision Problem. We start with the decision version as the easier problem. We combine the previous intermediate results from Corollary 10.1.8 and Main Lemmas 10.4.3, 10.6.7 and 10.7.3 in the straightforward way to prove Main Theorem III.3. We refer to Figure 10.1 for an illustration of the proof outline.

Main Theorem III.3. Fix two finite sets $\sigma, \rho \subseteq \mathbb{N}$ with $0 \notin \rho$. If there is an $\varepsilon > 0$ and an algorithm that decides whether there is a (σ, ρ) -set in a given graph G with a given path decomposition of width w in time $(c_{\sigma, \rho} - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-SETH fails.

Proof. By Corollary 10.1.8 we know that there is an A -manager with $|A| = c_{\sigma, \rho}$ as $0 \notin \rho$. Hence, by Main Lemma 10.4.3, there is a constant d such that there is no algorithm that solves (σ, ρ) -DOMSET^{Rel} in time $(c_{\sigma, \rho} - \varepsilon)^{w_G + \mathcal{O}(1)} \cdot |G|^{\mathcal{O}(1)}$ on a graph with relations G with arity at most d that is given with a tree decomposition of width w_G .

Toward a contradiction, let G denote precisely such an instance of (σ, ρ) -DOMSET^{Rel} where the arity is at most the constant d , and where we are given a path decomposition of width w_G . By the pathwidth-preserving reductions from Main Lemmas 10.6.7 and 10.7.3, we can transform this instance G into an instance H of (σ, ρ) -DOMSET such that, the width of the new path decomposition satisfies $\text{pw}(H) \leq w_G + \mathcal{O}(1)$ and the size of H depends polynomially on the size of G .

Now assume that (σ, ρ) -DOMSET can be solved in time $(c_{\sigma, \rho} - \varepsilon)^{w_{H'}} \cdot |H'|^{\mathcal{O}(1)}$ for a graph H' that is given with a path decomposition of width $w_{H'}$. We apply this algorithm on input H and recover the solution for G in time

$$|G|^{\mathcal{O}(1)} + (c_{\sigma, \rho} - \varepsilon)^{\text{pw}(H)} \cdot |H|^{\mathcal{O}(1)} \leq (c_{\sigma, \rho} - \varepsilon)^{w_G + \mathcal{O}(1)} \cdot |G|^{\mathcal{O}(1)}.$$

This immediately contradicts MONO-pw-CSPH by Main Lemma 10.4.3, and hence, the result follows by Main Theorem I.3. $\square$

Counting Problem. For the counting version we follow the same ideas as for the decision version. We combine Corollary 10.1.8 and Main Lemmas 10.4.4 and 10.5.8 with Main Lemmas 10.6.7 and 10.8.1 or Main Lemma 10.9.1. We refer to Figure 10.1 for an illustration of the proof outline and to Figure 10.23 on page 389 for a detailed overview of all reductions involved in the proofs for the counting version.

Main Theorem III.4. Fix two non-empty finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$ with $\rho \neq \{0\}$ and $(\sigma, \rho) \neq (\mathbb{N}, \mathbb{N})$.

If there is an $\varepsilon > 0$ and an algorithm that counts the number of (σ, ρ) -sets in a given graph G with a given path decomposition of width w in time $(c_{\sigma, \rho} - \varepsilon)^w \cdot |G|^{\mathcal{O}(1)}$, then pw-#SETH fails.

Proof. Based on σ and ρ , we define $\hat{\sigma}$ and $\hat{\rho}$ to be finite or simple cofinite sets such that

- if σ is finite, then we set $\hat{\sigma} = \sigma$,
- if σ is cofinite, then we define $\hat{\sigma}$ as a simple cofinite set with $\text{top } \hat{\sigma} = \text{top } \sigma$,
- if ρ is finite, then we set $\hat{\rho} = \rho$, and

- if ρ is cofinite, then we define $\hat{\rho}$ as a simple cofinite set with $\text{top } \hat{\rho} = \text{top } \rho$.

We observe that we have $c_{\hat{\sigma}, \hat{\rho}} = c_{\sigma, \rho}$.

By Corollary 10.1.8, we know that there is an A -manager with $|A| = c_{\sigma, \rho}$ as $\hat{\sigma}$ and $\hat{\rho}$ are non-trivial. Hence, by Main Lemma 10.4.4, there is a constant d such that there is no algorithm that solves $(\hat{\sigma}, \hat{\rho})\text{-}\#\text{DOMSET}^{\text{Rel}}$ in time $(c_{\sigma, \rho} - \varepsilon)^{w_G + \mathcal{O}(1)} \cdot |G|^{\mathcal{O}(1)}$ on a graph with relations G and arity at most d that is given with a tree decomposition of width w_G .

Toward a contradiction, let G denote precisely such an instance of $(\hat{\sigma}, \hat{\rho})\text{-}\#\text{DOMSET}^{\text{Rel}}$ where the arity is at most the constant d , and where we are given a path decomposition of width w_G . By the pathwidth- and arity-preserving reduction from Main Lemma 10.5.8, we transform instance G into polynomially (in the size of G) many instances of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{Rel}}$. Moreover, if σ is finite or $\rho \neq \mathbb{N}$, then we apply the reductions from Main Lemmas 10.6.7 and 10.8.1 or, if σ is cofinite and $\rho = \mathbb{N}$, then we apply the reduction from Main Lemma 10.9.1 to each of these instances of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{Rel}}$. We denote the resulting instances of $(\sigma, \rho)\text{-}\#\text{DOMSET}$ by H_i .

As all reductions are polynomial-time reductions in the size of the input graph, the number of instances H_i and their size is bounded by a polynomial in $|G|$. Moreover, the reductions provide a path decomposition for each H_i of width at most $\text{pw}(H_i) \leq w_G + \mathcal{O}(1)$.

Now assume that $(\sigma, \rho)\text{-}\#\text{DOMSET}$ can be solved in time $(c_{\sigma, \rho} - \varepsilon)^{w_H} \cdot |H|^{\mathcal{O}(1)}$ on a graph H that is given with a path decomposition of width w_H .

For each i , we apply this faster algorithm for $(\sigma, \rho)\text{-}\#\text{DOMSET}$ on input H_i and recover the solution for G in time

$$|G|^{\mathcal{O}(1)} + \sum_i (c_{\sigma, \rho} - \varepsilon)^{\text{pw}(H_i)} \cdot |H_i|^{\mathcal{O}(1)} \leq (c_{\sigma, \rho} - \varepsilon)^{w_G + \mathcal{O}(1)} \cdot |G|^{\mathcal{O}(1)}.$$

This immediately contradicts $\text{MONO-pw-}\#\text{CSPH}$ by Main Lemma 10.4.4, and hence, the result follows by Main Theorem 1.3. $\square$

10.2 Providers

In the following, we construct the managers from Main Lemma 10.3.7.

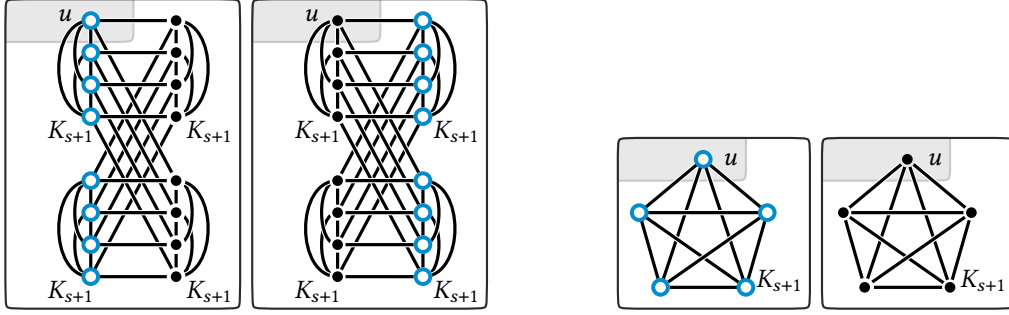
Definition 8.1.10 (Realized Language and L -Providers). *For a graph with portals (G, U) , we define realized language as*

$$L(G, U) := \{x \in \mathbb{A}_{\text{full}}^U \mid x \text{ is compatible with } (G, U)\}.$$

For a language $L \subseteq \mathbb{A}_{\text{full}}^U$, we say that (G, U) is an L -realizer if $L = L(G, U)$, and we say that (G, U) is an L -provider if $L \subseteq L(G, U)$.

Later, in Section 10.3, we then use said providers in the construction for Main Lemma 10.3.7.

We start with simple L -providers in Section 10.2.1, which in particular includes providers for languages that contain only ρ -states. Next, in Section 10.2.2 we construct providers for languages that contain only σ -states; later, we use these providers to construct the managers



(a) Valid partial solutions for the $\{\sigma_s, \rho_r\}$ -provider constructed in Lemma 10.2.1 for the case when $r \geq 1$. Here shown for $r = 2$ and $s = 3$.

(b) Valid partial solutions for the $\{\sigma_s, \rho_r\}$ -provider constructed in Lemma 10.2.2 for the case when $r = 0$. Here shown for $s = 4$.

Figure 10.4: The construction of the $\{\sigma_s, \rho_r\}$ -providers from Lemmas 10.2.1 and 10.2.2.

from Cases 1 and 2 in Main Lemma 10.3.7, that is, A -managers where we have that either $A \subseteq \mathbb{R}$ or $A \subseteq \mathbb{S}$.

Finally, in Section 10.2.3, we construct providers that contain σ -states together with ρ -states. These providers then allow us to construct A -managers for A that contain states from both $\mathbb{S}$ and $\mathbb{R}$. These managers correspond to Cases 3 and 4 in Main Lemma 10.3.7.

10.2.1 Simple General-Purpose Providers

We start with basic providers that have a single σ -state and a single ρ -state. This type of provider turns out to be useful in many constructions and is used repeatedly throughout the rest of this work. For this reason, we also give additional properties on the partial solution sets that are needed later on.

We refer to Figure 10.4a for a visualization of the following two constructions and the mentioned (partial) solutions.

Lemma 10.2.1. *Fix two non-empty sets $\sigma, \rho \subseteq \mathbb{N}$. For all $s \in \sigma$ and $r \in \rho$ with $r \geq 1$, there is a $\{\sigma_s, \rho_r\}$ -provider $(G, \{u\})$ that has disjoint (σ, ρ) -sets X and Y with $X \cup Y = V(G)$, where $u \in X$ with $|N(u) \cap X| = s$ and $|N(u) \cap Y| = r$.*

Proof. Let G consist of $2r$ cliques $X_1, \dots, X_r, Y_1, \dots, Y_r$, each of order $s + 1$. For $i \in [1 \dots r]$, let $x_0^{(i)}, \dots, x_s^{(i)}$ be the vertices of X_i , and let $y_0^{(i)}, \dots, y_s^{(i)}$ be the vertices of Y_i . These cliques are connected in a way that they form bicliques indexwise, that is, for each $i \in [1 \dots r]$ and each $j \in [0 \dots s]$, the vertex $x_j^{(i)}$ is adjacent not only to the vertices in its clique X_i , but also to the vertices $y_j^{(1)}, \dots, y_j^{(r)}$. Let $u = x_1^{(1)}$. Note that both $X = \bigcup_{i=1}^r X_i$ and $Y = \bigcup_{i=1}^r Y_i$ are (σ, ρ) -sets of G , where $u \in X$ with $|N(u) \cap X| = s$, and $u \notin Y$ with $|N(u) \cap Y| = r$. $\square$

We observe that Lemma 10.2.1 excludes the case $r = 0$. This is why we also give the following construction.

Lemma 10.2.2. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$. For all $s \in \sigma$ and $r \in \rho$, there is a $\{\sigma_s, \rho_r\}$ -provider.*

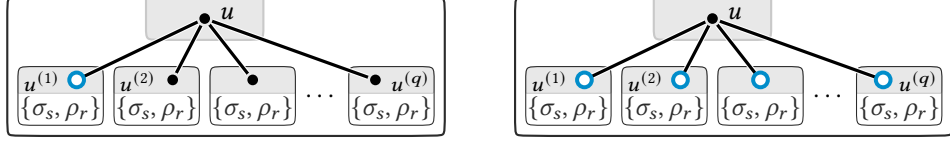


Figure 10.5: The construction of the provider from Lemma 10.2.3 with two exemplary partial solutions that correspond to the states ρ_1 and $\rho_{r_{\text{top}}}$ (here shown for $q = r_{\text{top}} = 4$). For any $\{\sigma_s, \rho_r\}$ -provider $(H^{(i)}, \{u^{(i)}\})$ used in the construction, we depict only its portal vertex and label it with $u^{(i)}$.

Proof. We define a graph G with a single portal u . If $r \neq 0$, then we define $(G, \{u\})$ to be the graph constructed in Lemma 10.2.1.

So, suppose $r = 0$. We choose G to be a clique on $s + 1$ vertices, and we declare any of its vertices to be u . Observe that both selecting all vertices and selecting no vertices constitute valid partial solutions, and hence, the states σ_s and $\rho_0 = \rho_r$ are compatible with $(G, \{u\})$. Therefore, $(G, \{u\})$ is a $\{\sigma_s, \rho_r\}$ -provider. $\square$

Next, we construct a provider for all possible elements from $\mathbb{R}$. We refer to Figure 10.5 for a visualization of the following construction.

Lemma 10.2.3. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$. For all $q \geq 1$, there is a $\{\rho_0, \rho_1, \dots, \rho_q\}$ -provider. Especially, if ρ is finite or cofinite, then there is a $\{\rho_0, \rho_1, \dots, \rho_{r_{\text{top}}}\}$ -provider.*

Proof. Fix two numbers $s \in \sigma$ and $r \in \rho$. We define a graph G with a single portal u . To that end, we take q independent copies $(H^{(i)}, \{u^{(i)}\})$ of the $\{\sigma_s, \rho_r\}$ -provider from Lemma 10.2.2, and connect each vertex $u^{(i)}$ to the vertex u .

By Lemma 10.2.2, each provider $(H^{(i)}, \{u^{(i)}\})$ has at least a partial solution that selects $u^{(i)}$ and a partial solution that does not select $u^{(i)}$. Hence, for each $j \in [0 \dots q]$, the constructed graph G has at least one partial solution that selects exactly j neighbors of u ; this completes the proof. $\square$

10.2.2 A Provider for σ -States

For our next construction, we need regular (bipartite) graphs, which are fortunately easy to construct.

Proposition 10.2.4. *For any non-negative integer $d \leq n$, the bipartite graph $M_n^{(d)}$, where*

$$\begin{aligned} V(M_n^{(d)}) &:= \{v_0, \dots, v_{n-1}\} \cup \{w_0, \dots, w_{n-1}\}, \\ E(M_n^{(d)}) &:= \{\{v_i, w_j\} \mid ((i - j) \bmod n) \in [0 \dots d]\} \end{aligned}$$

is d -regular and contains an edge $\{v_i, w_i\}$ for each $i \in [0 \dots n)$.

Proof. Observe that, for each $b \in [0 \dots d)$ and each $i \in [0 \dots n)$, there is exactly one solution to the equation $i - x \equiv_n b$ and similarly, there is one solution for the equation $x - i \equiv_n b$.

This concludes the proof as, for each $i \in [0 \dots n)$, there are d choices for b . $\square$

Recall from Chapter 2 that, for a vector x and a state $a \in \mathbb{A}$, we write $\#_a(x)$ to denote the number of occurrences of a in x .

Lemma 10.2.5. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$. For all $r \in \rho$ with $r \geq 1$ such that there is some $s \in \sigma$ with $s \geq r$, there is an L_r -provider for the language*

$$L_r := \{x \in \{\sigma_0, \sigma_1\}^{4r} \mid \#_{\sigma_1}(x) \in \{0, 2r\}\}.$$

Moreover, the closed neighborhoods of the portals are pairwise disjoint.

Proof. We denote by $U = \{u_1, \dots, u_{4r}\}$ the set of portal vertices. We define an L -provider (G, U) as follows with r and s as in the statement.

For every $A \subseteq [1 \dots 4r]$ such that $|A| = 2r$, the provider G has the following vertices in addition to its portals.

- We add $(2s + 2) \cdot r$ vertices which are partitioned into blocks $V_A^1, \dots, V_A^{2s+2}$, where each block V_A^q contains r vertices $v_{A,1}^q, \dots, v_{A,r}^q$.
- Moreover, we add $2s \cdot r + 2(r - 1)$ vertices which are partitioned into blocks $W_A^1, \dots, W_A^{2s+2}$, where the first two blocks W_A^q for $q \in \{1, 2\}$, contain $r - 1$ vertices $w_{A,1}^q, \dots, w_{A,r-1}^q$. All other blocks W_A^q , for $q \in [3 \dots 2s + 2]$, contain r vertices $w_{A,1}^q, \dots, w_{A,r}^q$.
- We define $V_A := \bigcup_{q \in [1 \dots 2s+2]} V_A^q$ and $W_A := \bigcup_{q \in [1 \dots 2s+2]} W_A^q$.

To define the edges, fix a set A and suppose that $A = \{k_1, \dots, k_{2r}\}$. Then, we add the following edges to the provider G .

- For all $q \in [1 \dots 2s + 2]$, we add a complete bipartite graph between V_A^q and W_A^q .
- We add a graph on V_A such that every $v \in V_A^1 \cup V_A^2$ has degree $s - 1$, and every $v \in V_A \setminus (V_A^1 \cup V_A^2)$ has degree s .

This is possible by Proposition 10.2.4. Indeed, $|V_A|$ is even and $|V_A|/2 \geq s$. So, we can build an s -regular graph on V_A . Additionally, by numbering vertices appropriately, we can assume that there is a matching between V_A^1 and V_A^2 . We remove these matching edges to decrease the degree of every vertex from $V_A^1 \cup V_A^2$ by one to achieve the desired result.

- We add an s -regular graph on W_A which is possible by Proposition 10.2.4 since $|W_A|$ is even and $|W_A|/2 \geq s$.
- For all $p \in [1 \dots r]$, we add the edge $u_{k_p} v_{A,p}^1$.
- For all $p \in [r + 1 \dots 2r]$, we add the edge $u_{k_p} v_{A,p-r}^2$.

This completes the description of the graph G . From the last two items, it follows that no two portals share an edge and that their neighborhoods are disjoint.

We start with some simple observations. We fix a set $A \subseteq [1 \dots 4r]$ such that $|A| = 2r$ and start with some simple observations.

1. The graph $G[W_A]$ is s -regular. This follows immediately from third item of the construction above.
2. We have $|N_G(v) \cap V_A| = s - 1$ for all $v \in V_A^1 \cup V_A^2$, and $|N_G(v) \cap V_A| = s$ for all $v \in V_A \setminus (V_A^1 \cup V_A^2)$. This follows immediately from the second item of the construction above as we remove a matching on the vertices in $V_A^1 \cup V_A^2$.
3. We have $|N_G(v) \cap W_A| = r - 1$ for all $v \in V_A^1 \cup V_A^2$, and $|N_G(v) \cap W_A| = r$ for all $v \in V_A \setminus (V_A^1 \cup V_A^2)$. This follows from the definition of W_A^1 and W_A^2 to contain $r - 1$ vertices only.
4. Moreover, $|N_G(w) \cap V_A| = r$ for all $w \in W_A$ as the blocks V_A^q have size r each.
5. Finally, $|N_G(v) \cap U| = 1$ for every $v \in V_A^1 \cup V_A^2$.

Now, fix some state-vector $x \in \{\sigma_0, \sigma_1\}^{4r}$ such that $\#_{\sigma_1}(x) \in \{0, 2r\}$, and write $x = \sigma_{i_1} \sigma_{i_2} \dots \sigma_{i_{4r}}$. We need to show that $x \in L(G, U)$. Let $A_x := \{p \in [1 \dots 4r] \mid i_p = 1\}$.

First, suppose that $A_x = \emptyset$. Then, we define a solution set

$$S_x := U \cup \bigcup_{A \subseteq [1 \dots 4r] : |A|=2r} W_A.$$

Using the basic observations outlined above, it is easy to verify that this solution set indeed witnesses that $x \in L(G, U)$.

Otherwise, we have $|A_x| = 2r$. Now, we define a solution set

$$S_x := U \cup V_{A_x} \cup \bigcup_{A \subseteq [1 \dots 4r] : |A|=2r, A \neq A_x} W_A.$$

Again, using the basic observations outlined above, it is straightforward to verify that this solution set indeed witnesses that $x \in L(G, U)$. $\square$

10.2.3 Providers for Combinations of σ - and ρ -States

Complementary to the previous section, we establish L -providers in this section, where L contains states from both $\mathbb{S}$ and $\mathbb{R}$. These providers are relevant if (σ, ρ) is m -structured for $m \leq 2$, but not m -structured for any $m \geq 3$.

We start with two auxiliary providers which we use in later constructions.

We refer to Figure 10.6 for an illustration of the provider from Lemma 10.2.6.

Lemma 10.2.6. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$.*

For all $s \in \sigma$ and $r \in \rho$ with $s, r \geq 1$, there is an $L_{s,r}$ -provider where we set

$$L_{s,r} := \{\sigma_{s-1} \sigma_{s-1} \rho_{r-1}, \quad \rho_r \rho_r \rho_r, \quad \rho_{r-1} \rho_r \sigma_s\}.$$

Proof. We wish to construct a graph G with three portals $U = \{u_1, u_2, u_3\}$, such that G has (at least) three partial solutions: (1) one partial solution with exactly two portals, namely u_1 and u_2 , selected, (2) one partial solution with no selected portal, and (3) one partial solution

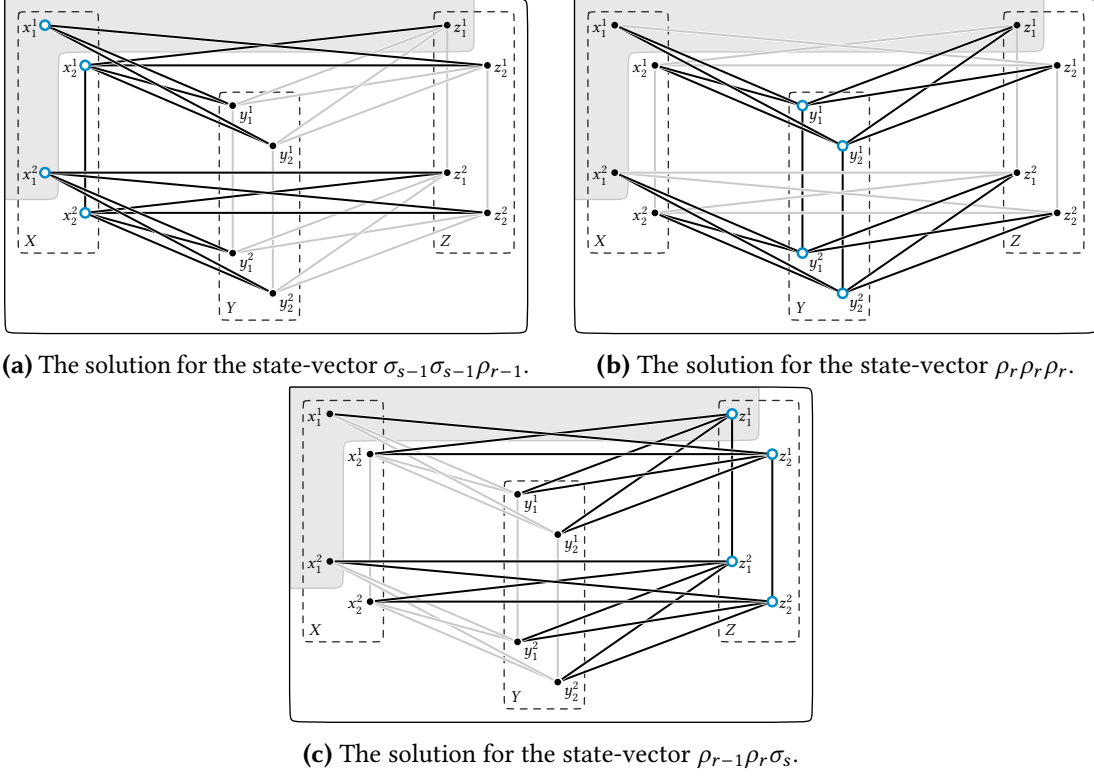


Figure 10.6: The construction of the provider $L_{s,r}$ from Lemma 10.2.6 together with the solutions for the three provided states where $u_1 = x_1^1$, $u_2 = x_1^2$, and $u_3 = z_1^1$. Here shown for $r = 2$ and $s = 1$.

with exactly one portal, namely u_3 , selected. For each case we have additional constraints on the number of selected neighbors.

Intuitively, we construct G to consist of three parts X, Y, Z that, when (fully) selected (and when nothing else is selected), each correspond to one of the desired partial solutions; that is, we have $u_1, u_2 \in X$ and $u_3 \in Z$.

Hence, for each of the parts X, Y, Z , we need to be able to either fully select the part or to not select any vertex of the part. To ensure the former, each vertex in X, Y, Z is part of a clique on $(s + 1)$ vertices, and to ensure the latter, each vertex in X, Y, Z has r neighbors in each of the corresponding other parts. We achieve this property by introducing a total of r cliques (each of order $s + 1$) per part. For each pair of different parts, we add edges to form a complete r -regular bipartite graph between vertices that have the same index in different cliques.

Formally, we define G as follows. For the vertices, we set

$$\begin{aligned} X &:= \{x_i^j \mid i \in [1..r], j \in [1..s+1]\}, \\ Y &:= \{y_i^j \mid i \in [1..r], j \in [1..s+1]\}, \quad \text{and} \\ Z &:= \{z_i^j \mid i \in [1..r], j \in [1..s+1]\}. \end{aligned}$$

For the vertices u_1, u_2, u_3 , slightly adapted requirements hold.

Further, we set

$$u_1 := x_1^1, \quad u_2 := x_1^2, \quad \text{and} \quad u_3 := z_1^1.$$

Now, for each $i \in [1 \dots r]$, we connect the vertices in $\{x_i^j \mid j \in [1 \dots s+1]\}$ to form a clique on $s+1$ vertices. We apply the same to the vertices in $\{y_i^j \mid j \in [1 \dots s+1]\}$ and the vertices in $\{z_i^j \mid j \in [1 \dots s+1]\}$,

Then, between each pair of sets from X, Y, Z , we connect the vertices with the same index $j \in [1 \dots s+1]$ to form a complete bipartite graph, which has r vertices in each part. Finally, we remove the edge between $u_1 = x_1^1$ and $u_2 = x_1^2$, and the edge between $u_1 = x_1^1$ and $u_3 = z_1^1$.

It remains to show that $(G, \{x_1^1, x_1^2, z_1^1\})$ is an $L_{s,r}$ -provider. We give partial solutions corresponding to the three elements of $L_{s,r}$.

$\sigma_{s-1}\sigma_{s-1}\rho_{r-1}$: Select the vertices from X . Thus, x_1^1 and x_1^2 are selected, but z_1^1 is not. To satisfy the σ -constraints, each vertex in X , except for x_1^1 and x_1^2 , obtains s selected neighbors from (their respective clique in) X . Because of the missing edge, both x_1^1 and x_1^2 have only $s-1$ selected neighbors each.

To satisfy the ρ -constraints, each vertex from Y and Z , except for z_1^1 , obtains r selected neighbors from X . Because of the missing edge to x_1^1 , vertex z_1^1 has only $r-1$ selected neighbors.

$\rho_r\rho_r\rho_r$: Select the vertices from Y . As before, we can check that this selection witnesses $\rho_r\rho_r\rho_r$.

$\rho_{r-1}\rho_r\sigma_s$: Select the vertices from Z . As before, we can check that this selection witnesses $\rho_{r-1}\rho_r\sigma_s$. $\square$

For ease of notation, we abbreviate, for all $\tau_t \in \mathbb{A}$ and all $\delta \in \mathbb{N}$, the state $\tau_t \dots \tau_t \in \mathbb{A}^\delta$ of a graph with δ portals by the notation $(\tau_t)^\delta$.

Lemma 10.2.7. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$. Consider $s, s' \in \sigma$, $r \in \rho$ with $s, r \geq 1$ and $s' > s_{\min}$, where $s_{\min} := \min \sigma$. For any $k \in \mathbb{N}$ and $\delta := k \cdot (s' - s_{\min})$, there is a $\{(\rho_r)^\delta, (\rho_{r-1})^\delta, (\sigma_s)^\delta\}$ -provider (with δ portals).*

Proof. We define a graph G with $\delta = k \cdot (s' - s_{\min})$ portals as follows.

- There are three disjoint sets of vertices $A = \{a_i \mid i \in [1 \dots \delta]\}$, $B = \{b_i \mid i \in [1 \dots \delta]\}$, and $C = \{c_i \mid i \in [1 \dots \delta]\}$, where the vertices of C form the δ portals of G .
- For each $i \in [1 \dots \delta]$,
- the three vertices (a_i, b_i, c_i) are in this order the portals of an attached $\{\sigma_{s-1}\sigma_{s-1}\rho_{r-1}, \rho_r\rho_r\rho_r, \rho_{r-1}\rho_r\sigma_s\}$ -provider J_i , which exists by Lemma 10.2.6 and the fact that $s, r \geq 1$.
- There is a set D_A of size k , where each vertex of D_A is the portal of an attached $\{\sigma_{s_{\min}}, \rho_r\}$ -provider, which exists by Lemma 10.2.2. We introduce edges between A and D_A in such a way that each vertex in A has precisely 1 neighbor in D_A , and each vertex in D_A has precisely $s' - s_{\min}$ neighbors in A . This is possible as $\delta = k \cdot (s' - s_{\min})$.

- Similarly, there is a set D_B of size k , where each vertex of D_B is the portal of an attached $\{\sigma_{s_{\min}}, \rho_r\}$ -provider, each vertex in B has precisely 1 neighbor in D_B , and each vertex in D_B has precisely $s' - s_{\min}$ neighbors in B .

We show that (G, C) is a $\{(\rho_r)^\delta, (\rho_{r-1})^\delta, (\sigma_s)^\delta\}$ -provider. We give solutions corresponding to the three states.

$(\rho_r)^\delta$: None of the vertices in A, B, C, D_A, D_B are selected. For each $i \in [1.. \delta]$, we select the vertices in J_i according to the solution that witnesses the state $\rho_r \rho_r \rho_r$. In the $\{\sigma_{s_{\min}}, \rho_r\}$ -providers attached to D_A and D_B , we select the vertices according to the ρ_r -state.

For each i , the vertices a_i and b_i , as well as the portal c_i , are unselected and each obtain r selected neighbors from J_i . Each vertex in D_A and D_B is also unselected and obtains r selected neighbors from the attached $\{\sigma_{s_{\min}}, \rho_r\}$ -provider.

$(\rho_{r-1})^\delta$: We select the vertices in $A \cup B \cup D_A \cup D_B$. For each $i \in [1.. \delta]$, we select the vertices in J_i according to the solution that witnesses the state $\sigma_{s-1} \sigma_{s-1} \rho_{r-1}$. In the $\{\sigma_{s_{\min}}, \rho_r\}$ -providers attached to D_A and D_B , select vertices according to the $\sigma_{s_{\min}}$ -state.

For each i , the portal c_i is unselected and obtains $r - 1$ selected neighbors from J_i . The vertices a_i and b_i are selected and each obtain $s - 1$ selected neighbors from J_i , and a_i also obtains 1 selected neighbor from D_A , and b_i obtains 1 selected neighbor from D_B . Thus, both obtain a total of $s \in \sigma$ selected neighbors.

The vertices in D_A are selected and each obtain $s' - s_{\min}$ selected neighbors from A , as well as $s_{\min}$ selected neighbors from the attached $\{\sigma_{s_{\min}}, \rho_r\}$ -provider, for a total of $s' \in \sigma$. Similarly, the vertices in D_B are selected and obtain $s' \in \sigma$ selected neighbors.

$(\sigma_s)^\delta$: We select the vertices $C \cup D_A$. For each $i \in [1.. \delta]$, we select the vertices in J_i according to the solution that witnesses the state $\rho_{r-1} \rho_r \sigma_s$. In the $\{\sigma_{s_{\min}}, \rho_r\}$ -providers attached to D_A , select vertices according to the $\sigma_{s_{\min}}$ -state. In the $\{\sigma_{s_{\min}}, \rho_r\}$ -providers attached to D_B , select vertices according to the ρ_r -state.

For each i , the portal c_i is selected and obtains s selected neighbors from J_i . The vertex a_i is unselected, obtains $r - 1$ selected neighbors from J_i , and obtains 1 selected neighbor from D_A . The vertex b_i is unselected and obtains r selected neighbors from J_i .

The vertices in D_A are selected and each obtain $s_{\min}$ selected neighbors from the attached $\{\sigma_{s_{\min}}, \rho_r\}$ -provider. The vertices in D_B are unselected and each obtain r selected neighbors from the attached $\{\sigma_{s_{\min}}, \rho_r\}$ -provider. $\square$

Before we continue with the next gadget, we first prove a technical result which ensures the existence of bipartite graphs with certain degree sequences.

Proposition 10.2.8. *Let $c_1, \dots, c_\ell, d_1, \dots, d_k \in \mathbb{N}$ such that*

$$\sum_{i \in [1.. \ell]} c_i = \sum_{j \in [1.. k]} d_j.$$

Also, suppose a is another natural number such that $a \geq \max\{c_1, \dots, c_\ell, d_1, \dots, d_k\}$. Then, for every $s \geq a$, there is a bipartite graph $G = (V \dot{\cup} W, E)$ such that

1. $V = \{v_1, \dots, v_\ell, x_1, \dots, x_s\}$,
2. $\deg_G(v_i) = c_i$ for all $i \in [1.. \ell]$, and $\deg_G(x_i) = a$ for all $i \in [1.. s]$,
3. $W = \{w_1, \dots, w_k, y_1, \dots, y_s\}$, and
4. $\deg_G(w_i) = d_i$ for all $i \in [1.. k]$, and $\deg_G(y_i) = a$ for all $i \in [1.. s]$.

Proof. We give an iterative construction for G . Initially, we set $E(G) \equiv \emptyset$. We define $\gamma(v_i) \equiv c_i$ for all $i \in [1.. \ell]$, and $\gamma(x_j) \equiv a$ for $j \in [1.. s]$. Similarly, we define $\gamma(w_i) \equiv d_i$ for all $i \in [1.. k]$, and $\gamma(y_j) \equiv a$ for $j \in [1.. s]$. Throughout the construction, the value $\gamma(u)$ denotes the number of neighbors still to be added to $u \in V \cup W$.

We maintain the property that

- (i) $\gamma(u) \geq 0$ for all $u \in V \cup W$,
- (ii) $\sum_{v \in V} \gamma(v) = \sum_{w \in W} \gamma(w)$, and
- (iii) $|\{v \in V \mid \gamma(v) \in \{\gamma_{\max}, \gamma_{\max} - 1\}\}| \geq a$ where $\gamma_{\max} \equiv \max_{v \in V} \gamma(v)$.

Observe that this property is initially satisfied by the choice of s and a .

If $\gamma(w) = 0$ for all $w \in W$, then $\gamma(v) = 0$ for all $v \in V$ by Conditions (i) and (ii), and the construction is complete.

So, fix some vertex $w \in W$ such that $c(w) > 0$ and set $d \equiv \gamma(w)$. Let $v'_1, \dots, v'_{\ell+s}$ be a list of all the vertices from V ordered according to the current capacities, that is, we have $\gamma(v'_i) \geq \gamma(v'_{i+1})$ for all $i \in [1.. \ell + s - 1]$.

First, we claim that $\gamma(v'_i) \geq 1$ for all $i \in [1.. d]$. If $c_{\max} \geq 2$, then this follows directly from Condition (iii) because $d \leq a$. Otherwise, $\gamma_{\max} = 1$. But, then $|\{v \in V \mid \gamma(v) = \gamma_{\max}\}| \geq d$ by Condition (ii).

We add an edge between w and v'_i for all $i \in [1.. d]$. Also, we update $\gamma(w) \equiv 0$ and decrease $\gamma(v'_i)$ by one for all $i \in [1.. d]$.

Clearly, Conditions (i) and (ii) are still satisfied. For Condition (iii), we observe that every $v \in V$ such that $\gamma(v) \in \{\gamma_{\max}, \gamma_{\max} - 1\}$ before the update, still satisfies this condition after the update because of the ordering of the vertices.

Repeating this process until all capacities are equal to zero, we obtain the desired graph G . $\square$

For the following construction, we exploit that (σ, ρ) is 1-structured or 2-structured. For the latter case, the property of being 2-structured raises parity issues which do not allow us to have the same provider as for the 1-structured case. Instead, we design a provider where the state of one portal depends on the state of the other portal, and in fact, both states must be equal for this provider.

Lemma 10.2.9. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ with $\rho \neq \{0\}$. Suppose there is a maximum value m such that (σ, ρ) is m -structured.*

1. *If $m = 1$, then there is a $\{\rho_0, \rho_1, \sigma_0\}$ -provider.*
2. *If $m = 2$, then there is a $\{\rho_0, \rho_2, \sigma_0\}$ -provider. There is also a $\{\rho_0\rho_0, \rho_1\rho_1, \sigma_0\sigma_0\}$ -provider.*

Proof. We first claim that there are finite subsets $\sigma' \subseteq \sigma$ and $\rho' \subseteq \rho$ such that there exists a maximum value m' such that (σ', ρ') is m' -structured, and moreover $m' = m$. Indeed, if σ is finite, we simply set $\sigma' := \sigma$. Otherwise, we can greedily add to σ' the smallest element from σ that is not yet contained in σ' until σ' is m structured but not m' -structured for any $m' > m$. In particular, if σ is cofinite, then there is some $s \in \mathbb{N}$ such that $s, s+1 \in \sigma$. In particular, it follows that $m = 1$. In this case we set $\sigma' := \{s, s+1\}$ which implies that σ' is 1-structured, but not m' -structured for any $m \geq 2$. The set ρ' is defined analogously.

Since every L -provider with respect to (σ', ρ') is also an L -provider with respect to (σ, ρ) , it suffices to prove the lemma for the pair (σ', ρ') . For ease of notation, let us suppose that $(\sigma, \rho) = (\sigma', \rho')$, that is, in the remainder of the proof, we suppose that both σ and ρ are finite.

Let $\sigma = \{s_1, \dots, s_{|\sigma|}\}$ and $\rho = \{r_1, \dots, r_{|\rho|}\}$, where we assume that elements are ordered with respect to size. The fact that m is the maximum value such that (σ, ρ) is m -structured is equivalent to

$$\gcd(\{r - r_1 \mid r \in \rho\} \cup \{s - s_1 \mid s \in \sigma\}) = m. \quad (10.2.1)$$

We claim that there are non-negative integers $x_i, \tilde{x}_i$ for $i \in [2 \dots |\rho|]$ and non-negative integers $y_j, \tilde{y}_j$ for $j \in [2 \dots |\sigma|]$ such that

$$m + \sum_{i=2}^{|\rho|} (r_i - r_1) \cdot (x_i - \tilde{x}_i) + \sum_{j=2}^{|\sigma|} (s_j - s_1) \cdot (y_j - \tilde{y}_j) = 0 \quad (10.2.2)$$

Indeed, Equation (10.2.2) corresponds to a Diophantine equation with variables $(x_i - \tilde{x}_i)$ and $(y_j - \tilde{y}_j)$ (for $i \in [2 \dots |\rho|]$ and $j \in [2 \dots |\sigma|]$) and hence, has solutions by Equation (10.2.1). Moreover, we can choose non-negative values for the indeterminates accordingly.

Using these integer solutions, we define a graph G with one portal as follows. The vertices of G are partitioned into two sets V and $\tilde{V}$ together with some attached providers.

The set V consists of the following vertices and edges.

- The portal vertex u is part of V and is intended to have m neighbors in $\tilde{V}$.
- For each $i \in [2 \dots |\rho|]$, we create a set A_i of x_i vertices. Each such vertex is intended to have $r_i - r_1$ neighbors in $\tilde{V}$.
- For some yet to be determined positive integer k , we create a set A^* of k vertices. Each such vertex is intended to have $r_{|\rho|} - r_1$ neighbors in $\tilde{V}$.
- For each $j \in [2 \dots |\sigma|]$, we create a set B_j of $b_j := (s_j - s_1) \cdot y_j$ vertices. Each such vertex is intended to have 1 neighbor in $\tilde{V}$.
- We set $A := A^* \cup \bigcup_{i \in [2 \dots |\rho|]} A_i$ and $B := \bigcup_{j \in [2 \dots |\sigma|]} B_j$.
- We make each vertex in A the portal of an attached $\{\sigma_{s_1}, \rho_{r_1}\}$ -provider which exists by Lemma 10.2.2.
- As $\rho \neq \{0\}$ by assumption, we can choose some $r \in \rho$ with $r \geq 1$; for example, we can set $r = r_2$. For each $j \in [2 \dots |\sigma|]$, we make the vertices of B_j the b_j portals of an attached $\{(\rho_r)^{b_j}, (\rho_{r-1})^{b_j}, (\sigma_{s_j})^{b_j}\}$ -provider J_j which exists by Lemma 10.2.7 using the definition of b_j and the fact that $s_j, r \geq 1$.

When $|\sigma| = 1$, we could have $\sigma = \{0\}$ and the provider J_j would not exist. However, in this case no such provider is needed at all.

Analogously, the side $\tilde{V}$ consists of the following vertices and edges.

- For each $i \in [2 \dots |\rho|]$, we create a set $\tilde{A}_i$ of $\tilde{x}_i$ vertices. Each such vertex is intended to have $r_i - r_1$ neighbors in V .
- We create a set $\tilde{A}^*$ of k vertices. Each such vertex is intended to have $r_{|\rho|} - r_1$ neighbors in V .
- For each $j \in [2 \dots |\sigma|]$, we create a set $\tilde{B}_j$ of $\tilde{b}_j := (s_j - s_1) \cdot \tilde{y}_j$ vertices. Each such vertex is intended to have 1 neighbor in V .
- We set $\tilde{A} := \tilde{A}^* \cup \bigcup_{i \in [2 \dots |\rho|]} \tilde{A}_i$ and $\tilde{B} := \bigcup_{j \in [2 \dots |\sigma|]} \tilde{B}_j$.
- We make each vertex in $\tilde{A}$ the portal of an attached $\{\sigma_{s_1}, \rho_{r_1}\}$ -provider.
- For each $j \in [2 \dots |\sigma|]$, we make the vertices in $\tilde{B}_j$ the $\tilde{b}_j$ portals of an attached $\{(\rho_r)^{\tilde{b}_j}, (\rho_{r-1})^{\tilde{b}_j}, (\sigma_{s_j})^{\tilde{b}_j}\}$ -provider $\tilde{J}_j$. Here we can use the same value for r as in the definition of V .

We first verify that the intended number of edges going from V to $\tilde{V}$ is the same as the intended number of edges going from $\tilde{V}$ to V . Based on the construction it must hold that

$$\begin{aligned} m + \left(\sum_{i=2}^{|\rho|} (r_i - r_1) \cdot x_i \right) + ((r_{|\rho|} - r_1) \cdot k) + \left(\sum_{j=2}^{|\sigma|} (s_j - s_1) \cdot y_j \right) \\ = \left(\sum_{i=2}^{|\rho|} (r_i - r_1) \cdot \tilde{x}_i \right) + ((r_{|\rho|} - r_1) \cdot k) + \left(\sum_{j=2}^{|\sigma|} (s_j - s_1) \cdot \tilde{y}_j \right) \end{aligned}$$

which follows from Equation (10.2.2).

Note that $r_{|\rho|} - r_1$ is the maximum of the intended degrees. Therefore, we can use Proposition 10.2.8 with $a := r_{|\rho|} - r_1$ and some sufficiently large $k \geq a$, to introduce the edges between V and $\tilde{V}$ such that the intended degrees constraints in the bipartite graph are met.

We show that $(G, \{u\})$ is a $\{\rho_0, \rho_m, \sigma_0\}$ -provider. We give solutions corresponding to the three states.

ρ_0 : Both V and $\tilde{V}$ are unselected. In each of the $\{\sigma_{s_1}, \rho_{r_1}\}$ -providers attached to A and $\tilde{A}$, we select vertices according to the ρ_{r_1} -state. For each $j \in [2 \dots |\sigma|]$, we select the vertices in the providers J_j and $\tilde{J}_j$ according to the $(\rho_r)^{b_j}$ - and $(\rho_r)^{\tilde{b}_j}$ -state, respectively.

The portal u is unselected and has no selected neighbors, as required. Moreover, the vertices in A and $\tilde{A}$ are unselected and obtain $r_1 \in \rho$ selected neighbors from the attached $\{\sigma_{s_1}, \rho_{r_1}\}$ -providers. The vertices in B and $\tilde{B}$ are unselected and obtain $r \in \rho$ selected neighbors from the attached providers J_j and $\tilde{J}_j$, respectively.

ρ_m : The vertices in V are unselected, but the vertices in $\tilde{V}$ are selected. In each of the $\{\sigma_{s_1}, \rho_{r_1}\}$ -providers attached to A , select vertices according to the ρ_{r_1} -state. In each of the $\{\sigma_{s_1}, \rho_{r_1}\}$ -providers attached to $\tilde{A}$, select vertices according to the σ_{s_1} -state. For each

$j \in [2 \dots |\sigma|]$, we select the vertices in J_j according to the $(\rho_{r-1})^{b_j}$ -state, and select the vertices in $\tilde{J}_j$ according to the $(\sigma_{s_j})^{\tilde{b}_j}$ -state.

The portal u is unselected and has m selected neighbors in $\tilde{V}$, as required. For $i \in [2 \dots |\rho|]$, each vertex in A_i is unselected, obtains $r_1 \in \rho$ selected neighbors from the attached $\{\sigma_{s_1}, \rho_{r_1}\}$ -provider, and additionally obtains $r_i - r_1$ selected neighbors in $\tilde{V}$, which results in a total of $r_i \in \rho$ selected neighbors. The vertices in $\tilde{A}$ are selected and obtain $s_1 \in \sigma$ selected neighbors from the attached $\{\sigma_{s_1}, \rho_{r_1}\}$ -providers. They have no further selected neighbors.

For $j \in [2 \dots |\sigma|]$, each vertex in B_j is unselected, obtains $r - 1$ selected neighbors from J_j , and additionally obtains 1 selected neighbor from $\tilde{V}$, which results in a total of $r \in \rho$ selected neighbors. The vertices in $\tilde{B}$ are selected and obtain $s_j \in \sigma$ selected neighbors from the attached provider $\tilde{J}_j$.

σ_0 : The selection is symmetric to the previous state, that is, the vertices in V are selected, but the vertices in $\tilde{V}$ are unselected. In each of the $\{\sigma_{s_1}, \rho_{r_1}\}$ -providers attached to A , we select the vertices according to the σ_{s_1} -state. In each of the $\{\sigma_{s_1}, \rho_{r_1}\}$ -providers attached to $\tilde{A}$, we select the vertices according to the ρ_{r_1} -state. For each $j \in [2 \dots |\sigma|]$, we select the vertices in J_j according to the $(\sigma_{s_j})^{b_j}$ -state, and we select the vertices in $\tilde{J}_j$ according to the $(\rho_{r-1})^{\tilde{b}_j}$ -state.

The portal u is selected and has no selected neighbors, as required. The remaining analysis is symmetric to the previous case.

So, for $m \in \{1, 2\}$, this proves the existence of a $\{\rho_0, \rho_m, \sigma_0\}$ -provider. Note that, for $m = 2$, if we replace the single portal u of degree 2 with two portals, each of degree 1, then the construction yields a $\{\rho_0 \rho_0, \rho_1 \rho_1, \sigma_0 \sigma_0\}$ -provider. $\square$

In the remaining part, we extend these providers to a setting where the portals can also get neighbors when they are selected. That is, we also have the state σ_1 for the portal. We first handle the case when (σ, ρ) is 1-structured in Section 10.2.3.1, and then, independently, consider the case when the sets are 2-structured in Section 10.2.3.2.

10.2.3.1 1-Structured Sets

In this section, we consider the case where $m = 1$ is the largest m for which (σ, ρ) is m -structured. We first construct an auxiliary provider which we combine afterward with the provider from Lemma 10.2.9.

Lemma 10.2.10. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ such that there is a $\{\rho_0, \rho_1, \sigma_0\}$ -provider.*

If $\sigma \neq \{0\}$ and $\rho \neq \{0\}$, then there is a $\{\rho_0 \rho_0, \rho_0 \sigma_0, \sigma_0 \sigma_0, \sigma_1 \sigma_1\}$ -provider. Moreover, the closed neighborhoods of both portals are disjoint.

Proof. Let $(J, \{\hat{u}\})$ be a $\{\rho_0, \rho_1, \sigma_0\}$ -provider. We define a graph G with two portals u_1 and u_2 as follows. For this, we fix two integers $s \in \sigma$ and $r \in \rho$ with $s, r \geq 1$.

Let K be a clique with $s + 1$ vertices $v_1, \dots, v_{s+1}$ where we remove the edge between v_1

Note that K contains at least two vertices since $\sigma \neq \{0\}$.

and v_2 . Then, G consists of the modified clique K where we add the two edges u_1v_1 and u_2v_2 . Moreover, for every $i \in [1 \dots s+1]$, we add r copies of J to G , where we use v_i as the portal vertex for all r copies.

It remains to show that $(G, \{u_1, u_2\})$ is a $\{\rho_0\rho_0, \rho_0\sigma_0, \sigma_0\sigma_0, \sigma_1\sigma_1\}$ -provider. We give solutions corresponding to the four states.

$\rho_0\rho_0$: No vertex from K is selected, and so, the vertices u_1 and u_2 have no selected neighbors. To satisfy the ρ -constraints of the vertices in K , each vertex v_i obtains r selected neighbors from its attached copies of J , that is, for each of the r attached $\{\rho_0, \rho_1, \sigma_0\}$ -providers, we select the vertices corresponding to the ρ_1 -state.

$\rho_0\sigma_0$: No vertex from K is selected. Since u_2 is selected, v_2 already has one selected neighbor. From the copies of J attached to v_2 , vertex v_2 receives another $r-1$ selected neighbors, for a total of r selected neighbors, that is, $r-1$ of the attached $\{\rho_0, \rho_1, \sigma_0\}$ -providers are in the ρ_1 -state, and one provider is in the ρ_0 -state. The remaining vertices of K receive r selected neighbors via the attached copies of J .

$\sigma_0\sigma_0$: No vertex from K is selected. Now, both v_1 and v_2 already have one selected neighbor each, and receive another $r-1$ selected neighbors through their attached copies of J . The remaining vertices of K again get r selected neighbors via the attached copies of J .

$\sigma_1\sigma_1$: All vertices from K are selected. The copies of J do not provide further neighbors to these vertices as all of them are in the σ_0 -state. Both vertices v_1 and v_2 have $s-1$ selected neighbors in K since the edge between v_1 and v_2 is missing, and they are adjacent to the selected vertices u_1 and u_2 , respectively. Hence, they each have s selected neighbors in total. By construction, the remaining vertices of K have s selected neighbors in K , and thus, in total. $\square$

As a last result for the 1-structured case, we combine Lemma 10.2.10 with Lemma 10.2.9, to obtain the following result.

Lemma 10.2.11. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ with $\sigma \neq \{0\}$ and $\rho \neq \{0\}$. Suppose that $m = 1$ is the maximum value such that (σ, ρ) is m -structured.*

There is a $\{\rho_0\rho_0, \rho_1\rho_1, \rho_0\sigma_0, \rho_1\sigma_0, \sigma_0\sigma_0, \sigma_1\sigma_1\}$ -provider. Moreover, the closed neighborhoods of the portals in the provider are disjoint.

Proof. Let $(G_1, \{u_1\})$ and $(G_2, \{u_2\})$ be two copies of a $\{\rho_0, \rho_1, \sigma_0\}$ -provider as constructed in Lemma 10.2.9 which is possible as $\rho \neq \{0\}$. Let $(G_3, \{u_1, u_2\})$ be a $\{\rho_0\rho_0, \rho_0\sigma_0, \sigma_0\sigma_0, \sigma_1\sigma_1\}$ -provider which exists by Lemma 10.2.10. It is straightforward to check that $(G_1 \cup G_2 \cup G_3, \{u_1, u_2\})$ is a $\{\rho_0\rho_0, \rho_1\rho_1, \rho_0\sigma_0, \rho_1\sigma_0, \sigma_0\sigma_0, \sigma_1\sigma_1\}$ -provider. $\square$

10.2.3.2 2-Structured Sets

We now treat the case where (σ, ρ) is 2-structured, but not m -structured for any m larger than 2. We again start with an auxiliary result.

Lemma 10.2.12. Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ such that there is a $\{\rho_0, \rho_2, \sigma_0\}$ -provider. If $\sigma \neq \{0\}$ and $\rho \neq \{0\}$, then there is a $\{\rho_0\rho_0, \rho_1\rho_1, \sigma_0\sigma_0\}$ -provider, and there is a $\{\rho_0, \rho_2, \sigma_0, \sigma_2\}$ -provider.

Proof. Let $(G', \{u'\})$ be a $\{\rho_0, \rho_2, \sigma_0\}$ -provider and assume we can construct a $\{\rho_0, \sigma_0, \sigma_2\}$ -provider $(G, \{u\})$. By setting $u = u'$, $(G \cup G', \{u\})$ is a $\{\rho_0, \rho_2, \sigma_0, \sigma_2\}$ -provider. In the remainder of this proof, we show how to obtain a $\{\rho_0, \sigma_0, \sigma_2\}$ -provider $(G, \{u\})$.

By assumption, there is two number $s \in \sigma$ and $r \in \rho$ with $s, r \geq 1$.

Let W be a complete bipartite graph with vertex partition $\{w_1, \dots, w_s\}$ and $\{w'_1, \dots, w'_s\}$. We remove the edge between w_1 and w'_1 . The portal u is adjacent to w_1 and w'_1 . For each $i \in [1 \dots s]$, we define w_i and w'_i as the two portals of r copies of a $\{\rho_0\rho_0, \rho_1\rho_1, \sigma_0\sigma_0\}$ -provider.

Observe that $s \geq 1$, so these vertices always exist.

Then, G is the graph consisting of the bipartite graph W , the $s \cdot r$ attached $\{\rho_0\rho_0, \rho_1\rho_1, \sigma_0\sigma_0\}$ -providers, and the portal u .

We show that $(G, \{u\})$ is a $\{\rho_0, \sigma_0, \sigma_2\}$ -provider by giving solutions corresponding to the three states.

ρ_0 : No vertex from W is selected, and so, u has no selected neighbor. To satisfy the ρ -constraints of the vertices in W , all vertices w_i and w'_i receive r selected neighbors from the attached $\{\rho_0\rho_0, \rho_1\rho_1, \sigma_0\sigma_0\}$ -providers, that is, all such providers are in state $\rho_1\rho_1$.

σ_0 : No vertex from W is selected, and so, u has no selected neighbor. However, now u is selected, and so, w_1 and w'_1 already have one selected neighbor. Therefore, w_1 and w'_1 receive only $r - 1$ selected neighbors from the attached $\{\rho_0\rho_0, \rho_1\rho_1, \sigma_0\sigma_0\}$ -providers, that is, $r - 1$ of these providers in state $\rho_1\rho_1$, and the last one is in state $\rho_0\rho_0$. For $i \in [2 \dots s]$, the vertices w_i and w'_i receive $r \in \rho$ selected neighbors from the attached $\{\rho_0\rho_0, \rho_1\rho_1, \sigma_0\sigma_0\}$ -providers, as before.

σ_2 : All vertices from W are selected. Since W is a complete bipartite graph (minus one edge), the vertices $w_2, \dots, w_s$ and $w'_2, \dots, w'_s$ are adjacent to $s \in \sigma$ selected neighbors. The vertices w_1 and w'_1 have $s - 1$ selected neighbors in W and, additionally, the selected portal u as a neighbor. All $\{\rho_0\rho_0, \rho_1\rho_1, \sigma_0\sigma_0\}$ -providers are in state $\sigma_0\sigma_0$, and so, do not give any further selected neighbors to vertices in W . $\square$

As a last step, we combine Lemma 10.2.12 with Lemma 10.2.9 to obtain the final provider for the 2-structured case.

Lemma 10.2.13. Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ with $\sigma \neq \{0\}$ and $\rho \neq \{0\}$. Suppose that $m = 2$ is the maximum value such that (σ, ρ) is m -structured.

Suppose further that the elements of ρ and σ are even. Then, for all $q \geq 1$, there is an L_q -provider where $L_q := \{\rho_i \mid i \in [0 \dots q]\} \cup \{\sigma_i \mid i \in [0 \dots q]\}$.

Proof. Combining Lemmas 10.2.9 and 10.2.12, there is a $\{\rho_0, \rho_2, \sigma_0, \sigma_2\}$ -provider $(G, \{u\})$. We create $q/2$ copies $(G_i, \{u_i\})$ for $i \in [1 \dots q]$ of $(G, \{u\})$ and set $u_1 = \dots = u_q$, that is, we identify all portals with the same vertex. It is straightforward to verify that $(G_1 \cup \dots \cup G_q, \{u_1\})$ is an L_q -provider. $\square$

10.3 Managers

The goal of this section is to establish four A -managers with different $A \subseteq \mathbb{A}$.

Before we formally state the result, we first recall the definition of a manager here. See Figure 10.7 for an illustration of this definition.

Definition 10.3.1 (Managers). *For a set $A \subseteq \mathbb{A}_{\text{full}}$ of states, an A -manager is an infinite family $((G_\ell, U_\ell))_{\ell \geq 1}$ of pairs (G_ℓ, U_ℓ) such that*

- G_ℓ is a graph with relations and
- $U_\ell = \{u_1, \dots, u_\ell\} \subseteq V(G_\ell)$ is a set of ℓ distinguished vertices.

Moreover, there is a non-negative integer b that depends only on $s_{\text{top}}, r_{\text{top}}$ such that the following holds for every $\ell \geq 1$:

- The vertices from $V(G_\ell) \setminus U_\ell$ can be partitioned into 2ℓ vertex-disjoint sets $B_1, \dots, B_\ell$ and $\bar{B}_1, \dots, \bar{B}_\ell$ called blocks, such that
 - $|B_i| \leq b$ and $|\bar{B}_i| \leq b$ for all $i \in [1.. \ell]$,
 - $N(u_i) \subseteq B_i \cup \bar{B}_i$ for all $i \in [1.. \ell]$,
 - there are edges only between B_i and B_{i+1} , between $\bar{B}_i$ and $\bar{B}_{i+1}$, for each $i \in [1.. \ell - 1]$, and between B_ℓ and $\bar{B}_\ell$, and
 - for each relation R of G_ℓ , there is some $i \in [1.. \ell]$ such that either $\text{scp}(R) \subseteq N[B_i]$ or $\text{scp}(R) \subseteq N[\bar{B}_i]$.
- Each $x \in A^\ell \subseteq \mathbb{A}^\ell$ is managed in the sense that there is a unique (σ, ρ) -set S_x of G_ℓ such that for all $i \in [1.. \ell]$:
 - If $x[i] = \sigma_s$, then $u_i \in S_x$. Moreover, u_i has exactly s neighbors in $B_i \cap S_x$ and exactly $s_{\text{top}} - s$ neighbors in $\bar{B}_i \cap S_x$.
 - If $x[i] = \rho_r$, then $u_i \notin S_x$. Moreover, u_i has exactly r neighbors in $B_i \cap S_x$ and exactly $r_{\text{top}} - r$ neighbors in $\bar{B}_i \cap S_x$.

We refer to G_ℓ as the A -manager of rank ℓ (with distinguished vertices U_ℓ).

Note that all vertices in G_ℓ already have a feasible number of neighbors. That is, the graph does not have any potentially “unsatisfied” portals. In this sense, it is different from the graphs with portals that we use in other gadget constructions such as providers, for example.

Note that Definition 10.3.1 does not a priori rule out solutions that do *not* correspond to values $x \in A^\ell$. This especially implies that there might be solutions S'_x for $x \in A^\ell$ where vertex u_i with $x[i] = \sigma_s$ does not get exactly $s_{\text{top}} - s$ neighbors in $\bar{B}_i \cap S'_x$ but some different, yet feasible, number although u_i gets s neighbors in $B_i \cap S'_x$ (and similarly for the case when $x[i] = \rho_r$).

Now we can formally state our main result about managers, that is, we formally prove Main Lemma 10.3.7 in the following.

Main Lemma 10.3.7 (Existence of Managers). *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite or cofinite with $\rho \neq \{0\}$.*

1. *There is an A-manager with $|A| = r_{\text{top}} + 1$.*
2. *If $s_{\text{top}} \geq r_{\text{top}} \geq 1$, then there is an A-manager with $|A| = s_{\text{top}} + 1$.*
3. *If (σ, ρ) is 2-structured, but not m-structured for any $m \geq 3$, and s_{top} and $r_{\text{top}} \geq 1$ are even, then there is an A-manager with $|A| = (s_{\text{top}} + r_{\text{top}})/2 + 2$.*
4. *If (σ, ρ) is not m-structured for any $m \geq 2$, then there is an A-manager with $|A| = s_{\text{top}} + r_{\text{top}} + 2$.*

Moreover, each A-manager is such that A is closed under the inverse with respect to σ, ρ .

We split the proof of Main Lemma 10.3.7 into four cases, each case corresponding to one manager. For the construction of these managers, we mostly rely on the providers from Section 10.2.

In the following, we first construct the managers from the first and third case in Lemmas 10.3.2 and 10.3.3 as they follow rather directly from the constructed providers. Then, we show how to obtain the remaining two managers in Lemmas 10.3.5 and 10.3.6 assuming a general construction for managers based on specific providers. We defer to proof of this construction to Section 10.3.1. As a last step, we combine the intermediate results to prove Main Lemma 10.3.7.

Two Easy Managers. We start with the managers for the first and second case of Main Lemma 10.3.7. For the first manager, we use the provider from Lemma 10.2.3.

Lemma 10.2.3. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$. For all $q \geq 1$, there is a $\{\rho_0, \rho_1, \dots, \rho_q\}$ -provider. Especially, if ρ is finite or cofinite, then there is a $\{\rho_0, \rho_1, \dots, \rho_{r_{\text{top}}}\}$ -provider.*

When we use one provider for each block of the manager, we directly obtain Lemma 10.3.2.

Lemma 10.3.2. *For non-empty sets σ and ρ , there is an $\mathbb{R}$ -manager.*

Proof. Let ℓ be a fixed rank of the manager. For each $i \in [1.. \ell]$, the graph G contains a distinguished vertex u_i that is the portal of two independent copies of a $\{\rho_0, \rho_1, \dots, \rho_{r_{\text{top}}}\}$ -provider from Lemma 10.2.3. The first and second copy of the providers neighboring u_i form the sets B_i and $\bar{B}_i$, respectively.

To each copy of the provider, we add a relational constraint whose scope contains all vertices of the respective copy. This constraint ensures that, for each state ρ_r the corresponding copy could provide to vertex u_i , there is a unique way to obtain this state. Formally, for all $i \in [1.. \ell]$, we create two relations R_i and $\bar{R}_i$ with $\text{scp}(R_i) \coloneqq B_i$ and $\text{scp}(\bar{R}_i) \coloneqq \bar{B}_i$. Moreover, for all $r \in [0.. r_{\text{top}}]$, we pick a solution $S_{i,r}$ for B_i and a solution $\bar{S}_{i,r}$ for $\bar{B}_i$ where the vertex u_i obtains state ρ_r , respectively. Then, the relation R_i only accepts the sets $S_{i,r}$ as solutions and the relation $\bar{R}_i$ only accepts the sets $\bar{S}_{i,r}$ as solutions, that is, $R(R_i) \coloneqq \{S_{i,r} \mid r \in [0.. r_{\text{top}}]\}$ and $R(\bar{R}_i) \coloneqq \{\bar{S}_{i,r} \mid r \in [0.. r_{\text{top}}]\}$.

This defines an $\mathbb{R}$ -manager of rank ℓ since, for each $r \in [0.. r_{\text{top}}]$, both ρ_r and $\rho_{r_{\text{top}}-r}$ are compatible with the $\{\rho_0, \rho_1, \dots, \rho_{r_{\text{top}}}\}$ -provider. $\square$

As a next step, we construct the first manager with σ and ρ -states. For this construction we use the provider from Lemma 10.2.13.

Lemma 10.2.13. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ with $\sigma \neq \{0\}$ and $\rho \neq \{0\}$. Suppose that $m = 2$ is the maximum value such that (σ, ρ) is m -structured.*

Suppose further that the elements of ρ and σ are even. Then, for all $q \geq 1$, there is an L_q -provider where $L_q := \{\rho_i \mid i \in [0 \dots q]\} \cup \{\sigma_i \mid i \in [0 \dots q]\}$.

We use the same approach as for the previous manager to prove Lemma 10.3.3.

Lemma 10.3.3. *Let σ and ρ be two non-empty sets with $\rho \neq \{0\}$. Suppose that $m = 2$ is the maximum value such that (σ, ρ) is m -structured. Suppose further that all elements of ρ and σ are even.*

Then, there is an L -manager where

$$L := \{\rho_i \in \mathbb{R} \mid i \in 2\mathbb{N}\} \cup \{\sigma_i \in \mathbb{S} \mid i \in 2\mathbb{N}\}$$

and L is closed under the inverse with respect to σ, ρ .

Proof. The construction is analogous to that in the proof of Lemma 10.3.2, but we use the L_q -provider from Lemma 10.2.13 for $q := t_{\text{top}}$ instead of the $\{\rho_0, \rho_1, \dots, \rho_{r_{\text{top}}}\}$ -provider.

Note that r_{top} and s_{top} are even, and thus, $r_{\text{top}} - k$ and $s_{\text{top}} - k$ are even if k is even. $\square$

Two Non-Trivial Managers. Before we provide the last two managers, we first introduce some notation. Let x be a state-vector over $\mathbb{A}$ for this. Recall that, for $a \in \mathbb{A}$, we denote by $\#_a(x)$ the number of occurrences of a in x . Similarly, for a set $A \subseteq \mathbb{A}$, we denote by $\#_A(x)$ the number of occurrences of elements from A in x , that is, we write $\#_A(x) := \sum_{a \in A} \#_a(x)$.

With this notation, we define the following language which we use to construct the managers.

Definition 10.3.4. *Let $d \geq 1$ and $\beta \in [0 \dots 2d]$ denote two integers. We define a set*

$$L_\beta^{(2d)} := \{x \in \{\rho_0, \rho_1, \sigma_0, \sigma_1\}^{2d} \mid \#_{\sigma_1}(x) \in \{0, d\}, \#_{\mathbb{R}}(x) \leq \beta\}.$$

Using this definition, we can state a general construction for a manager.

Lemma 10.3.8. *Suppose that, for some $d \geq 1$ and $\beta \in \{0, 1\}$, there is an $L_\beta^{(2d)}$ -provider where the closed neighborhoods of the portals are disjoint. Then, there is an $\mathbb{S}$ -manager, and if $\beta = 1$, then there is also an $\mathbb{A}$ -manager.*

Due to its complexity, we defer the proof of Lemma 10.3.8 to Section 10.3.1.

We use Lemma 10.3.8 to prove Lemmas 10.3.5 and 10.3.6, that is, to construct the remaining two managers from Main Lemma 10.3.7. For the first manager, we require that the distinguished vertices are always selected. In this case, the necessary provider exists by Lemma 10.2.5.

Lemma 10.2.5. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$. For all $r \in \rho$ with $r \geq 1$ such that there is some $s \in \sigma$ with $s \geq r$, there is an L_r -provider for the language*

$$L_r := \{x \in \{\sigma_0, \sigma_1\}^{4r} \mid \#_{\sigma_1}(x) \in \{0, 2r\}\}.$$

Moreover, the closed neighborhoods of the portals are pairwise disjoint.

When we combine Lemmas 10.2.5 and 10.3.8 we directly obtain the following manager.

Lemma 10.3.5. *Let σ and ρ be two non-empty sets such that there is some $s \in \sigma$ and $r \in \rho$ with $s \geq r \geq 1$. Then, there is an $\mathbb{S}$ -manager.*

Proof. By Lemma 10.2.5, there is an $L_0^{(4r)}$ -provider. Moreover, we know that the closed neighborhoods of the portals in this provider are disjoint. Thus, the manager follows by Lemma 10.3.8. $\square$

For the last manager, we use the general provider from Lemma 10.2.11.

Lemma 10.2.11. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ with $\sigma \neq \{0\}$ and $\rho \neq \{0\}$. Suppose that $m = 1$ is the maximum value such that (σ, ρ) is m -structured.*

There is a $\{\rho_0\rho_0, \rho_1\rho_1, \rho_0\sigma_0, \rho_1\sigma_0, \sigma_0\sigma_0, \sigma_1\sigma_1\}$ -provider. Moreover, the closed neighborhoods of the portals in the provider are disjoint.

With this provider, the following manager concludes the series of constructions by providing an $\mathbb{A}$ -manager.

Lemma 10.3.6. *Let σ and ρ be two non-empty sets with $\rho \neq \{0\}$. Suppose that $m = 1$ is the maximum value such that (σ, ρ) is m -structured. Then, there is an $\mathbb{A}$ -manager.*

Proof. We first handle the case when $\sigma \neq \{0\}$. By Lemma 10.2.11, there is a Z -provider with $Z = \{\rho_0\sigma_0, \rho_1\sigma_0, \sigma_0\sigma_0, \sigma_1\sigma_1\}$. Moreover, the closed neighborhoods of both portals are disjoint. From this, we construct an $L_1^{(4)}$ -provider as follows. We introduce four vertices $u_1, \dots, u_4$ that serve as portals of the $L_1^{(4)}$ -provider. For each pair (u_i, u_j) of distinct portals, we attach an independent copy of a Z -provider where u_i is the first portal, and u_j is the second portal vertex. It is straightforward to verify that the resulting gadget is an $L_1^{(4)}$ -provider where the closed neighborhoods of the portals are disjoint. The lemma then follows from Lemma 10.3.8.

The other case is when $\sigma = \{0\}$, but we still have $\rho \neq \{0\}$. By Lemma 10.2.9, there is a $\{\rho_0, \rho_1, \sigma_0\}$ -provider J . For all i , the blocks B_i and $\bar{B}_i$ consist of t_{top} copies of J each using u_i as a portal. For each copy we introduce a relational constraint whose scope covers all vertices of that respective copy. This constraint ensures that for each state there is only a unique solution. Since $\sigma = \{0\}$ this finishes the construction of the $\mathbb{A}$ -manager of rank ℓ and proves its correctness. $\square$

By the definition of providers, we do not care if there are also solutions for other states.

Now we have everything ready, to prove Main Lemma 10.3.7.

Main Lemma 10.3.7 (Existence of Managers). *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite or cofinite with $\rho \neq \{0\}$.*

1. *There is an A -manager with $|A| = r_{\text{top}} + 1$.*
2. *If $s_{\text{top}} \geq r_{\text{top}} \geq 1$, then there is an A -manager with $|A| = s_{\text{top}} + 1$.*
3. *If (σ, ρ) is 2-structured, but not m -structured for any $m \geq 3$, and s_{top} and $r_{\text{top}} \geq 1$ are even, then there is an A -manager with $|A| = (s_{\text{top}} + r_{\text{top}})/2 + 2$.*

4. If (σ, ρ) is not m -structured for any $m \geq 2$, then there is an A -manager with $|A| = s_{\text{top}} + r_{\text{top}} + 2$.

Moreover, each A -manager is such that A is closed under the inverse with respect to σ, ρ .

Proof. For the first case, we use the $\mathbb{R}$ -manager from Lemma 10.3.2.

For the second case, we use the $\mathbb{S}$ -manager from Lemma 10.3.5 with $s := s_{\text{top}}$ and $r := r_{\text{top}}$.

For the third case, we use the L -manager from Lemma 10.3.3. As s_{top} and r_{top} are even, and (σ, ρ) are 2-structured, the sets σ and ρ contain only even numbers. Thus, we obtain that $|L| = s_{\text{top}}/2 + 1 + r_{\text{top}}/2 + 1$.

For the fourth case, we use the $\mathbb{A}$ -manager from Lemma 10.3.6, which concludes the proof. $\square$

In the remainder we prove Lemma 10.3.8.

10.3.1 A Generic Manager

Now, we turn to the proof of Lemma 10.3.8 and provide a general construction for a manager based on a given provider.

Lemma 10.3.8. *Suppose that, for some $d \geq 1$ and $\beta \in \{0, 1\}$, there is an $L_\beta^{(2d)}$ -provider where the closed neighborhoods of the portals are disjoint. Then, there is an $\mathbb{S}$ -manager, and if $\beta = 1$, then there is also an $\mathbb{A}$ -manager.*

Proof. To simplify notation, we set $L := L_\beta^{(2d)}$ in the following.

Construction of the Graph. Let ℓ be a fixed rank of the manager. We first define the graph G_ℓ with its set of distinguished vertices $U_\ell = \{u_1, \dots, u_\ell\}$. Afterward, we establish that it is an $\mathbb{S}$ -manager or $\mathbb{A}$ -manager for $\beta = 0$ or $\beta = 1$, respectively. We refer to Figure 10.7 for an illustration of the construction.

For ease of notation, we omit ℓ from G_ℓ and U_ℓ in the following, and just write G and U .

Apart from the vertices in U , for some value ℓ^* to be chosen later (depending on β), the graph G also contains $\ell^* - \ell$ auxiliary vertices $u_{\ell+1}, \dots, u_{\ell^*}$, which we use to circumvent some parity issues that would otherwise prevent us from obtaining proper solutions. In the construction of G , we treat these auxiliary vertices in the same way as the vertices in U . However, for the encoding that we want to model (see Definition 10.3.1), we are interested only in the vertices $u_1, \dots, u_\ell$ and their selected neighbors.

For all $i \in [1 \dots \ell^*]$, there are some subgraphs X_i and $\bar{X}_i$, which we define subsequently. These subgraphs provide the neighbors for u_i in G , and with some minor modifications, the subgraphs X_i and $\bar{X}_i$ form the blocks B_i and $\bar{B}_i$ of the manager G .

We define the remaining graph as follows.

- For all $i \in [0 \dots \ell^*]$ and all $j \in [1 \dots d]$, we create vertices $v_{i,j}$ and $\bar{v}_{i,j}$. For $j \in [1 \dots d]$, we identify $v_{\ell^*,j} = \bar{v}_{\ell^*,d+1-j}$. Intuitively, the case $i = \ell^*$ is special as it is the connection between the $v_{i,j}$'s and the $\bar{v}_{i,j}$'s.

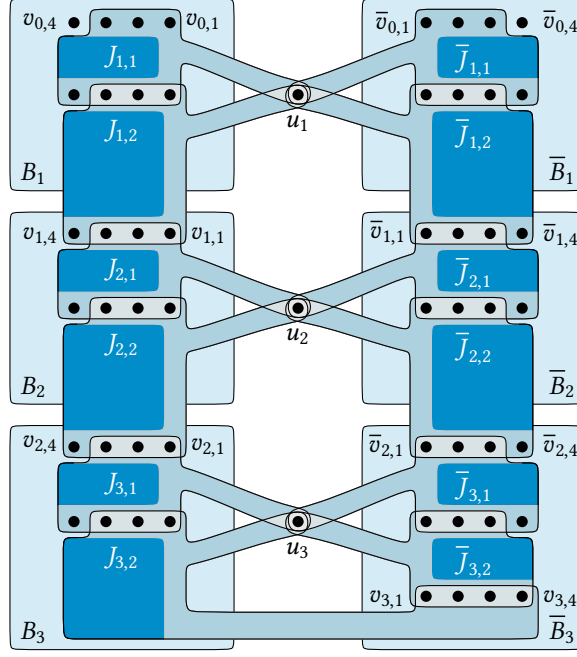


Figure 10.7: Construction of the manager from Lemma 10.3.8. The larger boxes represent the blocks, while the smaller boxes represent the L -providers $J_{i,z}$. The L -providers $F_{i,z}^j, \bar{F}_{i,z}^j$ are not shown to keep the figure simple.

- For each $i \in [0 \dots \ell^*]$, the vertices $v_{i,1}, \dots, v_{i,d}$ serve as portals of s_{top} copies of the given L -provider. Let us name these providers $F_i^1, \dots, F_i^{s_{\text{top}}}$.

Note that the L -provider has $2d$ portals, and therefore, each vertex in $v_{i,1}, \dots, v_{i,d}$ serves as two portals for each of these L -providers. Identifying two portals of an L -provider is possible without forming loops or multiple edges by the assumption that the portals of the given L -provider are independent and have no common neighbors.

We use F as in “fill” since these providers are used to fill up the count of selected neighbors of their portals as far as possible.

- Analogously, for each $i \in [0 \dots \ell^*]$, the vertices $\bar{v}_{i,1}, \dots, \bar{v}_{i,d}$ serve as portals of s_{top} L -providers $\bar{F}_i^1, \dots, \bar{F}_i^{s_{\text{top}}}$.
- For all $i \in [1 \dots \ell^*]$, there is a subgraph X_i defined as follows. It contains a set of vertices $\{x_{i,z,j} \mid z \in [0 \dots t_{\text{top}}], j \in [1 \dots d]\}$. For $z \notin \{0, t_{\text{top}}\}$, these $x_{i,z,j}$ ’s are new vertices, but, for all $j \in [1 \dots d]$, we set $x_{i,0,j} = v_{i-1,j}$ and $x_{i,t_{\text{top}},j} = v_{i,j}$.

Just like the $v_{i,j}$ ’s, for each $z \in [1 \dots t_{\text{top}} - 1]$, the vertices $x_{i,z,1}, \dots, x_{i,z,d}$ serve as portals of s_{top} copies $F_{i,z}^1, \dots, F_{i,z}^{s_{\text{top}}}$ of the given L -provider. We set $F_{i,0}^y = F_{i-1}^y$ and $F_{i,t_{\text{top}}}^y = F_i^y$ for all $y \in [1 \dots s_{\text{top}}]$.

Furthermore, for all $z \in [1 \dots t_{\text{top}}]$, the subgraph X_i contains an L -provider $J_{i,z}$ with portals $u_i, x_{i,z-1,1}, \dots, x_{i,z-1,d-1}$, and $x_{i,z,1}, \dots, x_{i,z,d}$.

- For each $i \in [1.. \ell^*]$, we similarly define a subgraph $\bar{X}_i$ with vertices $\{\bar{x}_{i,z,j} \mid z \in [0.. t_{\text{top}}], j \in [1.. d]\}$, and $\bar{x}_{i,0,j} = \bar{v}_{i-1,j}$, $\bar{x}_{i,t_{\text{top}},j} = \bar{v}_{i,j}$ (for all $j \in [1.. d]$).

For all $z \in [1.. t_{\text{top}} - 1]$, we create s_{top} L -providers $\bar{F}_{i,z}^1, \dots, \bar{F}_{i,z}^{s_{\text{top}}}$, that have the vertices $\bar{x}_{i,z,1}, \dots, \bar{x}_{i,z,d}$ as portals. We additionally set, for all $y \in [1.. s_{\text{top}}]$, $\bar{F}_{i,0}^y = \bar{F}_{i-1}^y$ and $\bar{F}_{i,t_{\text{top}}}^y = \bar{F}_i^y$.

Moreover, for all $z \in [1.. t_{\text{top}}]$, the subgraph $\bar{X}_i$ contains an L -provider $\bar{J}_{i,z}$ with portals $u_i, \bar{x}_{i,z-1,1}, \dots, \bar{x}_{i,z-1,d-1}$, and $\bar{x}_{i,z,1}, \dots, \bar{x}_{i,z,d}$.

- As a last step, we add a relational constraint whose scope is the union of the vertices in X_i with the vertex u_i . This constraint ensures that there is exactly one solution for each state the vertex u_i can obtain. For this, we use the solution which we construct in the following.

We apply the same to the vertices of $\bar{X}_i$ together with u_i .

This completes the construction of G . As a next step, we define the blocks of G .

Partitioning the Graph into Blocks. We show that there is an integer b , and a partition of G into blocks that fulfill the requirements stated in Definition 10.3.1. Note that the construction only depends on β via the definition of ℓ^* and thus, we use the same approach for $\beta = 0$ and $\beta = 1$.

Suppose that δ_L denotes the number of vertices of the L -provider (including the $2d$ portal vertices). Then, we set

$$b := (\ell^* - \ell + 1) \cdot (s_{\text{top}} + 1) \cdot (t_{\text{top}} + 1) \cdot \delta_L. \quad (10.3.1)$$

For all $i \in [1.. \ell - 1]$, the block B_i consists of the vertices of X_i minus the ones from X_{i+1} (and minus u_i). Note that the common vertices of X_i and X_{i+1} are $v_{i,1}, \dots, v_{i,d}$ together with the L -providers $F_i^1, \dots, F_i^{s_{\text{top}}}$. Likewise, the block $\bar{B}_i$ consists of the vertices of $\bar{X}_i$ minus the vertices from $\bar{X}_{i+1}$. The final blocks B_ℓ and $\bar{B}_\ell$ hold all the remaining vertices of $V(G) \setminus U$, that is, B_ℓ consists of all X_i with $i \in [\ell.. \ell^*]$ minus the vertices from $X_{\ell^*} \cap \bar{X}_{\ell^*}$, that is, block B_ℓ does not contain the vertices $v_{\ell^*,1} = \bar{v}_{\ell^*,d}, \dots, v_{\ell^*,d} = \bar{v}_{\ell^*,1}$ together with the L -providers $F_{\ell^*}^1, \dots, F_{\ell^*}^{s_{\text{top}}}$. Then, $\bar{B}_\ell$ consists of all $\bar{X}_i$ with $i \in [\ell.. \ell^*]$.

It is straightforward to verify that these blocks form a partition of the vertices of G excluding the distinguished vertices in U . Each set X_i contains the $s_{\text{top}} \cdot (t_{\text{top}} + 1)$ copies $F_{i,z}^s$ of the L -provider and the t_{top} copies $J_{i,z}$ of the L -providers. So, in total, this gives at most $(s_{\text{top}} + 1) \cdot (t_{\text{top}} + 1)$ copies of the L -provider in X_i . Since each vertex $x_{i,z,j}$ is a portal of at least one of those L -providers, we conclude that X_i contains at most $(s_{\text{top}} + 1) \cdot (t_{\text{top}} + 1) \cdot \delta_L$ vertices. The same bound holds for the subgraphs $\bar{X}_i$ for all $i \in [1.. \ell^*]$.

Since each block B_i or $\bar{B}_i$, where $i \in [1.. \ell]$, contains vertices from at most $(\ell^* - \ell + 1)$ subgraphs X_i or $\bar{X}_i$, it follows that $|B_i| \leq b$ and $|\bar{B}_i| \leq b$ for all $i \in [1.. \ell]$.

Finally, by construction, it follows that $N(u_i) \subseteq B_i \cup \bar{B}_i$ for all $i \in [1.. \ell]$, and there are edges only between B_i and B_{i+1} , between $\bar{B}_i$ and $\bar{B}_{i+1}$, for each $i \in [1.. \ell - 1]$, and between B_ℓ and $\bar{B}_\ell$.

Constructing a Solution. To show that the corresponding graph G together with the distinguished vertices $U = \{u_1, \dots, u_\ell\}$ is an $\mathbb{A}$ -manager when $\beta = 1$ or an $\mathbb{S}$ -manager when $\beta = 0$ of rank ℓ , we additionally need to prove the second property from Definition 10.3.1, that is, there are solutions corresponding to the intended states of the distinguished vertices.

For a given state-vector $x \in \mathbb{A}^\ell$, we define, for a value ℓ^* to be chosen later, a new state-vector $x^* \in \mathbb{A}^{\ell^*}$ with $x^*[i] = x[i]$ for all $i \in [1 \dots \ell]$, where we set the remaining entries later depending on β and x . We iteratively construct (partial) solutions $S_{i,z}$ for all $i \in [1 \dots \ell^*]$ and $z \in [0 \dots t_{\text{top}}]$ such that $S_x := S_{\ell^*, t_{\text{top}}}$ corresponds to the final solution.

For $\beta = 0$, we restrict ourselves to vectors from $\mathbb{S}^\ell$.

Along with these $S_{i,z}$, we define two functions $\mu, \bar{\mu}: [1 \dots \ell^*] \times [0 \dots t_{\text{top}}] \rightarrow [0 \dots d-1]$ such that $\mu(i, z) = k$ if and only if the k vertices $x_{i,z,1}, \dots, x_{i,z,k}$ need (exactly) one more neighbor in $S_{i,z}$, respectively for $\bar{\mu}(i, z) = k$ and the vertices $\bar{x}_{i,z,1}, \dots, \bar{x}_{i,z,k}$. Due to the construction of G , we set $S_{i, t_{\text{top}}} = S_{i+1,0}$, and therefore, $\mu(i, t_{\text{top}}) = \mu(i+1, 0)$ and $\bar{\mu}(i, t_{\text{top}}) = \bar{\mu}(i+1, 0)$ for all $i \in [1 \dots \ell^* - 1]$.

We later choose ℓ^* and x^* such that the partial solutions maintain the following invariant for all $i \in [1 \dots \ell^*]$:

$$\mu(i, t_{\text{top}}) + \bar{\mu}(i, t_{\text{top}}) \equiv_d s_{\text{top}} \cdot \#_{\mathbb{S}}(x^*[1 \dots i]) \quad \text{and} \quad \mu(1, 0) = \bar{\mu}(1, 0) = 0. \quad (10.3.2)$$

The first set $S_{1,0}$ consists of the vertices $x_{1,0,j}$ and $\bar{x}_{1,0,j}$ for all $j \in [1 \dots d]$. Moreover, the L -providers $F_{1,0}^1, \dots, F_{1,0}^{s_{\text{top}}}$ and $\bar{F}_{1,0}^1, \dots, \bar{F}_{1,0}^{s_{\text{top}}}$ give each of these vertices s_{top} neighbors. We directly get $\mu(1, 0) = \bar{\mu}(1, 0) = 0$ which satisfies the invariant.

Now, for each i from 1 to ℓ^* , we iterate over all z from 1 to t_{top} , and apply the following selection process. Unless mentioned otherwise, the portals of the L -providers get state σ_0 .

- Set $S_{i,z}$ consists of all selected vertices from $S_{i,z-1}$. If $x^*[i] \in \mathbb{S}$, then we select vertex u_i . Moreover, we select the vertices $x_{i,z,j}$ and $\bar{x}_{i,z,j}$ for all $j \in [1 \dots d]$. We use the L -providers $F_{i,z}^y$ and $\bar{F}_{i,z}^y$, for all $y \in [1 \dots s_{\text{top}} - 1]$, to add $s_{\text{top}} - 1$ neighbors to each of these vertices.
- Selection process for B_i if $x^*[i] = \sigma_s$.
 - If u_i has less than s neighbors in $B_i \cap S_{i,z-1}$, then we give u_i one more neighbor in $S_{i,z}$. The L -provider $J_{i,z}$ gives state σ_1 to u_i and each of $x_{i,z-1,1}, \dots, x_{i,z-1,\mu(i,z-1)}$ and each of $x_{i,z,\mu(i,z-1)+2}, \dots, x_{i,z,d}$. Hence, the $\mu(i, z-1) + 1$ vertices $x_{i,z,1}, \dots, x_{i,z,\mu(i,z-1)+1}$ need one more neighbor.
 - If $\mu(i-1, z) + 1 = d$, then we use the additional L -provider $F_{i,z}^{s_{\text{top}}}$ to give every vertex one additional neighbor. Therefore, $\mu(i, z) = \mu(i, z-1) + 1 \pmod d$.
 - If u_i has s neighbors in $B_i \cap S_{i,z-1}$, then we do not give it one more neighbor in $S_{i,z}$. The L -provider $J_{i,z}$ gives state σ_0 to u_i , and gives state σ_1 to each of $x_{i,z-1,1}, \dots, x_{i,z-1,\mu(i,z-1)}$ and $x_{i,z,\mu(i,z-1)+1}, \dots, x_{i,z,d}$. Hence, $\mu(i, z) = \mu(i, z-1)$.
- Selection process for $\bar{B}_i$ if $x^*[i] = \sigma_s$.
 - If u_i has less than $s_{\text{top}} - s$ neighbors in $\bar{B}_i \cap S_{i,z-1}$, then we give it one more neighbor in $S_{i,z}$. The L -provider $\bar{J}_{i,z}$ gives state σ_1 to u_i and each of $\bar{x}_{i,z-1,1}, \dots, \bar{x}_{i,z-1,\bar{\mu}(i,z-1)}$ and $\bar{x}_{i,z,\bar{\mu}(i,z-1)+2}, \dots, \bar{x}_{i,z,d}$. Hence, the $\bar{\mu}(i, z-1) + 1$ vertices $\bar{x}_{i,z,1}, \dots, \bar{x}_{i,z,\bar{\mu}(i,z-1)+1}$ need one more neighbor.

- If $\bar{\mu}(i-1, z) + 1 = d$, then we use the additional L -provider $\bar{F}_{i,z}^{\text{stop}}$ to give every vertex one additional neighbor. Therefore, $\bar{\mu}(i, z) = \bar{\mu}(i, z-1) + 1 \pmod d$.
- If u_i has $s_{\text{top}} - s$ neighbors in $\bar{B}_i \cap S_{i,z-1}$, then we do not give it one more neighbor in $S_{i,z}$. The L -provider $\bar{J}_{i,z}$ gives state σ_0 to u_i , and gives state σ_1 to each of $\bar{x}_{i,z-1,1}, \dots, \bar{x}_{i,z-1,\bar{\mu}(i,z-1)}$ and $\bar{x}_{i,z,\bar{\mu}(i,z-1)+1}, \dots, \bar{x}_{i,z,d}$. Hence, $\bar{\mu}(i, z) = \bar{\mu}(i, z-1)$.
 - Selection process for B_i if $x^*[i] = \rho_r$. Note that this case does not occur when $\beta = 0$.
 - If u_i has less than r neighbors in $B_i \cap S_{i,z-1}$, then we give it one more neighbor in $S_{i,z}$. The L -provider $J_{i,z}$ gives state ρ_1 to u_i , and gives state σ_1 to each of $x_{i,z-1,1}, \dots, x_{i,z-1,\mu(i,z-1)}$ and $x_{i,z,\mu(i,z-1)+1}, \dots, x_{i,z,d}$. Since the $\mu(i, z-1)$ vertices $x_{i,z,1}, \dots, x_{i,z,\mu(i,z-1)}$ need one more neighbor, $\mu(i, z) = \mu(i, z-1)$.
 - If u_i has r neighbors in $B_i \cap S_{i,z-1}$, then we do not give it one more neighbor in $S_{i,z}$. The L -provider $J_{i,z}$ gives state ρ_0 to u_i , and gives state σ_1 to each of $x_{i,z-1,1}, \dots, x_{i,z-1,\mu(i,z-1)}$ and $x_{i,z,\mu(i,z-1)+1}, \dots, x_{i,z,d}$. Hence, $\mu(i, z) = \mu(i, z-1)$.
 - Selection process for $\bar{B}_i$ if $x^*[i] = \rho_r$. Note that this case does not occur when $\beta = 0$.
 - If u_i has less than $r_{\text{top}} - r$ neighbors in $\bar{B}_i \cap S_{i,z-1}$, then we give it one more neighbor in $S_{i,z}$. The L -provider $\bar{J}_{i,z}$ gives state ρ_1 to u_i , and gives state σ_1 to each of $\bar{x}_{i,z-1,1}, \dots, \bar{x}_{i,z-1,\bar{\mu}(i,z-1)}$ and $\bar{x}_{i,z,\bar{\mu}(i,z-1)+1}, \dots, \bar{x}_{i,z,d}$. Since the $\bar{\mu}(i, z-1)$ vertices $\bar{x}_{i,z,1}, \dots, \bar{x}_{i,z,\bar{\mu}(i,z-1)}$ need one more neighbor, $\bar{\mu}(i, z) = \bar{\mu}(i, z-1)$.
 - If u_i has $r_{\text{top}} - r$ neighbors in $\bar{B}_i \cap S_{i,z-1}$, then we do not give it one more neighbor in $S_{i,z}$. The L -provider $\bar{J}_{i,z}$ gives state ρ_0 to u_i , and gives state σ_0 to each of $\bar{x}_{i,z-1,1}, \dots, \bar{x}_{i,z-1,\bar{\mu}(i,z-1)}$ and $\bar{x}_{i,z,\bar{\mu}(i,z-1)+1}, \dots, \bar{x}_{i,z,d}$. Hence, $\bar{\mu}(i, z) = \bar{\mu}(i, z-1)$.

Correctness of the Solution. We first show that these steps preserve the invariant in Equation (10.3.2). The claim immediately follows if $x^*[i] = \rho_r$ because, in this case, $\mu(i, z-1) = \mu(i, z)$ for all $z \in [1 \dots t_{\text{top}}]$. Likewise, we get $\bar{\mu}(i, z-1) = \bar{\mu}(i, z)$. Moreover, exactly d portals of the L -providers have state σ_1 and exactly one portal has state ρ_0 or ρ_1 .

For the case $x^*[i] = \sigma_s$, we get that $\mu(i, z) \equiv_d \mu(i, 0) + z$ for all $z \in [1 \dots s]$, and $\mu(i, z) = \mu(i, z-1)$ for all $z \in [s+1 \dots t_{\text{top}}]$. Conversely, we get $\bar{\mu}(i, z) \equiv_d \bar{\mu}(i, 0) + z$ for all $z \in [1 \dots s_{\text{top}} - s]$, and $\bar{\mu}(i, z) = \bar{\mu}(i, z-1)$ for all $z \in [s_{\text{top}} - s + 1 \dots t_{\text{top}}]$. This directly yields

$$\begin{aligned} \mu(i, t_{\text{top}}) + \bar{\mu}(i, t_{\text{top}}) &\equiv_d \mu(i, 0) + \bar{\mu}(i, 0) + s + (s_{\text{top}} - s) \\ &\equiv_d \#_{\mathbb{S}}(x^*[1 \dots i-1]) \cdot s_{\text{top}} + s_{\text{top}} \\ &\equiv_d \#_{\mathbb{S}}(x^*[1 \dots i]) \cdot s_{\text{top}}, \end{aligned}$$

proving that the invariant is preserved. Again, exactly d portals of the L -provider have state σ_1 .

Setting the Parameters. Assume we can set ℓ^* and $x^*[i]$ for all $i \in [\ell + 1 \dots \ell^*]$ such that the solution satisfies

$$\mu(\ell^*, t_{\text{top}}) + \bar{\mu}(\ell^*, t_{\text{top}}) \in \{0, d\}. \quad (10.3.3)$$

We claim that in this case, $S_{\ell^*, t_{\text{top}}}$ is indeed a valid solution.

From $\mu(\ell^*, t_{\text{top}}) = k$, we know that the vertices $v_{\ell^*,1}, \dots, v_{\ell^*,k}$ need one more neighbor. From $\bar{\mu}(\ell^*, t_{\text{top}}) = k'$, we know that the vertices $\bar{v}_{\ell^*,1}, \dots, \bar{v}_{\ell^*,k'}$ need one more neighbor. Recall that $v_{\ell^*,j} = \bar{v}_{\ell^*,d+1-j}$ for all j .

If $\mu(\ell^*, t_{\text{top}}) + \bar{\mu}(\ell^*, t_{\text{top}}) = d$, then $k' = d - k$ and the vertices $\bar{v}_{\ell^*,d-k+1}, \dots, \bar{v}_{\ell^*,d}$ that is, the vertices $v_{\ell^*,k}, \dots, v_{\ell^*,1}$, get a neighbor from $\bar{J}_{\ell^*,t_{\text{top}}}$. Conversely, the vertices $v_{\ell^*,k+1}, \dots, v_{\ell^*,d}$, that is, the vertices $\bar{v}_{\ell^*,d-k}, \dots, \bar{v}_{\ell^*,1}$ get a neighbor from $J_{\ell^*,t_{\text{top}}}$. Thus, none of these vertices need an additional neighbor.

If $\mu(\ell^*, t_{\text{top}}) + \bar{\mu}(\ell^*, t_{\text{top}}) = 0$, then the L -provider $F_{\ell^*}^{\text{stop}} = \bar{F}_{\ell^*}^{\text{stop}}$ already provided one additional neighbor for these vertices.

It remains to set ℓ^* and $x^*[i]$ for all $i \in [\ell+1 .. \ell^*]$ depending on β such that Equation (10.3.3) holds.

- For the case when $\beta = 1$, we set $\ell^* := \ell + d$ and $\ell' := d - (\#_{\mathbb{S}}(x) \bmod d)$. We define the vector x^* as $x^*[\ell+i] := \sigma_0$ for all $i \in [1 .. \ell']$ and as $x^*[\ell+i] := \rho_0$ for all $i \in [\ell'+1 .. d]$.

From the invariant in Equation (10.3.2), we get that

$$\begin{aligned} \mu(\ell^*, t_{\text{top}}) + \bar{\mu}(\ell^*, t_{\text{top}}) &\equiv_d s_{\text{top}} \cdot \#_{\mathbb{S}}(x^*) \\ &\equiv_d s_{\text{top}} \cdot (\#_{\mathbb{S}}(x) + \ell') \\ &\equiv_d s_{\text{top}} \cdot (\#_{\mathbb{S}}(x) + d - (\#_{\mathbb{S}}(x) \bmod d)) \equiv_d 0 \end{aligned}$$

- For the case when $\beta = 0$, we set $\ell^* = \ell + d - (\ell \bmod d)$ and $x^*[\ell+i] = \sigma_0$ for all $i \in [1 .. d - (\ell \bmod d)]$.

By the invariant in Equation (10.3.2) and the fact that $x^* \in \mathbb{S}^{\ell^*}$, we get:

$$\begin{aligned} \mu(\ell^*, t_{\text{top}}) + \bar{\mu}(\ell^*, t_{\text{top}}) &\equiv_d s_{\text{top}} \cdot \#_{\mathbb{S}}(x^*) \\ &\equiv_d s_{\text{top}} \cdot \ell^* \\ &\equiv_d s_{\text{top}} \cdot (\ell + d - (\ell \bmod d)) \equiv_d 0 \end{aligned}$$

We complete the proof of Lemma 10.3.8 by observing that $\ell^* - \ell + 1 \leq d + 1$ in both cases, which means that the value of b from Equation (10.3.1) depends only on s_{top} and r_{top} . $\square$

10.4 Lower Bound for the Intermediate Problem

In the following, we prove the intermediate lower bound for (σ, ρ) -DOMSET^{Rel}, that is, we prove Main Lemma 10.4.3 for the decision version and the corresponding result for the counting version of (σ, ρ) -DOMSET^{Rel} as stated in Main Lemma 10.4.4.

Main Lemma 10.4.3 (Lower Bound for (σ, ρ) -DOMSET^{Rel}). *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite or simple cofinite. Suppose there is an $A \subseteq \mathbb{A}$ that is closed under the inverse with respect to σ, ρ such that there is an A -manager.*

For all $\varepsilon > 0$, there is a constant d such that (σ, ρ) -DOMSET^{Rel} on instances of size n and arity at most d cannot be solved in time $(|A| - \varepsilon)^{p+\mathcal{O}(1)} \cdot n^{\mathcal{O}(1)}$, even if the input is given with a path decomposition of width p , unless MONO-pw-CSPH fails.

Observe that each A -manager from Main Lemma 10.3.7, satisfies the conditions from Main Lemma 10.4.3, that is, each such set A is closed under the inverse with respect to σ, ρ (see Definition 10.1.6 for the definition of the inverse). Hence, we can always think of one of the managers from Main Lemma 10.3.7 as a concrete example for the managers used in the following.

Similar to the lower bound for Λ -FACTOR, we do not reduce from SAT, but from q -CSP- B instead. Moreover, we use some additional assumptions for the given instances such that we can base our lower bound on MONO-pw-CSPH for the decision version and MONO-pw-#CSPH for the counting problem.

We restate Conjecture 4.1.11 here for convenience and refer to Conjecture 4.1.12 for the formal statement of MONO-pw-#CSPH as the corresponding version for the counting problem.

Conjecture 4.1.11 (MONO-pw-CSPH). *For all $B \geq 2$ and $\varepsilon > 0$, there exists a q such that no algorithm can, for a q -CSP- B instance ϕ that is given with*

- *a partition $V_1 \dot{\cup} V_2$ of the variables,*
- *a nice path decomposition of the primal graph of ϕ of width w*
- *where every bag contains at most $\mathcal{O}(B \cdot \log|V_1|)$ variables from V_2 ,*
- *an injective mapping c from the constraints of ϕ to the bags such that the bag contains the variables of the respective constraint,*
- *the promise that every satisfying multi-assignment that is consistent on V_2 is also consistent on V_1 ,*

decide if there is a monotone satisfying multi-assignment that is consistent on V_2 in time

$$(B - \varepsilon)^w \cdot |\phi|^{\mathcal{O}(1)}.$$

With this assumption, the main idea reads as follows; we follow essentially the same ideas as for the lower bound for Λ -FACTOR in Chapter 7.

Intuitive Idea. Denote by ϕ an instance of q -CSP- B with the additional properties stated in Conjecture 4.1.11 where we choose $B \doteq |A|$. Especially, we have access to a path decomposition of the primal graph of ϕ .

For each bag $\beta(j)$ of the decomposition and each variable v appearing in this bag, we create a vertex t_v^j to which we refer as *transmission vertex*. For each bag $\beta(j)$, we create an A -manager and denote it by M^j . We use the vertices $w_\star^j$ as the distinguished vertices of this manager.

We determine the states of a vertex t_v^j depending on the left blocks of M^j , that is, depending on the selection status of vertex t_v^j and the number of selected neighbors it gets in the left block. Then, we define a one-to-one-mapping mapping between the states in A and the values for the variables of the CSP instance. That is, we fix some bijective function $\text{enc}: [1 \dots |A|] \rightarrow A$. Hence, we can directly relate a state of a vertex to an assignment for the corresponding variable.

We first define the relations that model the constraints of the CSP-instance. Without loss of generality, we can assume that for each bag $\beta(j)$ there is a unique constraint C_j of ϕ that is assigned to this bag. Otherwise, we could create a dummy constraint that observes one variable of the bag and always accepts. We define a relational constraint R^j that observes precisely those left blocks from the A -manager M^j that correspond to the variables that appear in C_j . By the above mapping enc, the relation R^j accepts only those states that correspond to an assignment that satisfies constraint C_j .

It remains to ensure that the states (and thus, the encoded assignments) are propagated correctly through the entire graph. For each variable v that appears in the bags $\beta(j)$ and $\beta(j+1)$, we introduce a relational constraint $\bar{R}$ where the scope contains the right block of t_v^j in M^j , the left block of t_v^{j+1} in M^{j+1} , and the two corresponding information vertices. Now the relation $\bar{R}$ ensures that the state of the left block is the inverse of the state from the right block with respect to σ and ρ .

Note that we actually consider monotone assignments.

The pathwidth of the construction does not increase too much as we follow the path decomposition of the formula ϕ .

With this intuition, we move to the formal proof. As we are dealing with the counting version as well, we have to be more careful about the number of (partial) solutions of the providers (see Definition 8.1.10). We are especially interested in L -providers which do not have any “unwanted” (partial) solutions that might change the output. We make this formal by the following definition.

For the decision version, we do not need these modifications.

Definition 10.4.1 (Parsimonious Provider). *Let G be a graph (possibly with relations) together with a set of portals $U \subseteq V(G)$. For a language $L \subseteq \mathbb{A}_{\text{full}}$, we say that (G, U) is a parsimonious L -provider if (G, U) is an L -realizer (as defined in Definition 10.1.5) and, for all $x \in L$, there is a unique partial solution that witnesses x .*

Observe that the requirement of being an L -realizer rules out the possibility of having (partial) solutions which witness an element not contained in L . By Lemma 10.2.2, there is a $\{\sigma_s, \rho_r\}$ -provider for all $s \in \sigma$ and $r \in \rho$. We extend this now to a parsimonious provider.

Lemma 10.4.2. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$. For all $s \in \sigma$ and $r \in \rho$, there is a parsimonious $\{\sigma_s, \rho_r\}$ -provider.*

Proof. Let (G, u) be the $\{\sigma_s, \rho_r\}$ -provider from Lemma 10.2.2 with portal u . Let S_0 be a partial solution corresponding to state ρ_r , and let S_1 be a partial solution corresponding to the state σ_s . We add a relation R to the graph which observes all vertices of G , including u . Relation R accepts only two possible selections of vertices, namely S_0 whenever u is unselected, and S_1 if u is selected. The properties of the parsimonious provider follow immediately. $\square$

Now we have everything ready to prove Main Lemmas 10.4.3 and 10.4.4.

Main Lemma 10.4.3 (Lower Bound for (σ, ρ) -DOMSET^{Rel}). *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite or simple cofinite. Suppose there is an $A \subseteq \mathbb{A}$ that is closed under the inverse with respect to σ, ρ such that there is an A -manager.*

For all $\varepsilon > 0$, there is a constant d such that (σ, ρ) -DOMSET^{Rel} on instances of size n and arity at most d cannot be solved in time $(|A| - \varepsilon)^{p+\mathcal{O}(1)} \cdot n^{\mathcal{O}(1)}$, even if the input is given with a path decomposition of width p , unless MONO-pw-CSPH fails.

Main Lemma 10.4.4. Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite or simple cofinite. Suppose there is an $A \subseteq \mathbb{A}$ that is closed under the inverse with respect to σ, ρ such that there is an A -manager.

For all $\varepsilon > 0$, there is a constant d such that (σ, ρ) -#DOMSET^{Rel} on instances of size n and arity at most d cannot be solved in time $(|A| - \varepsilon)^{k+\mathcal{O}(1)} \cdot n^{\mathcal{O}(1)}$, even if the input is given with a path decomposition of width k , unless MONO-pw-#CSPH fails.

Proof of Main Lemmas 10.4.3 and 10.4.4. We set $B := |A|$ and design a parsimonious reduction from q -CSP- B to (σ, ρ) -DOMSET^{Rel}. Assume we are given a q -CSP- B formula ϕ with n variables $V = V_1 \cup V_2$. We denote the m constraints of ϕ by $C_1, \dots, C_m$.

Moreover, assume that the given path decomposition consists of the bags $\beta(1), \dots, \beta(r)$ and satisfies the constraints from Conjecture 4.1.11. Additionally, assume that we are given an injective mapping from the constraint of ϕ to the bags $\beta(j)$. Without loss of generality, we assume that there is a bijection between the constraints and the bags. We can easily assume this by introducing dummy constraints that observe only one variable and accepts all possible assignments to this variable. Hence, we especially have that $r = m$.

We first define the instance G_ϕ of (σ, ρ) -DOMSET^{Rel} and afterward, establish the necessary properties of G_ϕ . Finally, we put the pieces together to show that a faster algorithm for (σ, ρ) -DOMSET^{Rel} gives an algorithm for q -CSP- B that would contradict MONO-pw-CSPH. We refer to Figure 10.24 on page 390 for an illustration of the following construction.

Construction of G_ϕ . In the following, the indices we use are superscripts if they refer to “columns” of the construction (which usually refers to bags/constraints), and they are subscripts if they refer to “rows” of the construction (which usually refers to variables). Whenever we say that some set of vertices V' is subject to a relation R , we mean that there is a relational constraint C with $\text{scp}(C) = V'$ and $R(C) = R$.

The graph G_ϕ consists of the following vertices, edges and relations.

- For all $v \in V$ and $j \in [\text{intro}(v) .. \text{forget}(v) - 1]$, we create a vertex t_v^j .
- For all $j \in [1 .. m]$, we define $V_j := (V_1 \cap \beta(j) \cap \beta(j+1)) \cup (V_2 \cap \beta(j))$ and create a copy of the given A -manager of rank $|V_j|$, denote it as M^j , and use the vertices in $\{t_v^j \mid v \in V_j\}$ as distinguished vertices.

For each distinguished vertex t_v^j of M^j , we denote by B_v^j and $\bar{B}_v^j$ the left and the right block of manager M_j for vertex t_v^j .

Moreover, we denote by N_v^j and $\bar{N}_v^j$ the neighbors of t_v^j in B_v^j and $\bar{B}_v^j$, respectively.

- For all $v \in V_1$, we create t_{top} vertices, denoted by the set $N_v^{\text{forget}(v)-1}$. We make each of the vertices in $\{t_v^{\text{forget}(v)-1}\} \cup N_v^{\text{forget}(v)-1}$ subject to a parsimonious $\{\sigma_{\text{min}}, \rho_{\text{min}}\}$ -provider which exists by Lemma 10.4.2.
- For all $j \in [1 .. m]$, we write $V(C_j)$ for the scope of constraint C_j , that is, for the set of variables that appear in the constraint C_j . Then, the set of vertices

$$Z^j := \bigcup_{v \in V(C_j)} (\{t_v^j\} \cup N_v^j)$$

The manager avoids those vertices t_v^j where variable $v \in V_1$ is forgotten at bag $j+1$.

is subject to a relation R^j , which we define in a moment.

- For all $v \in V$ and $j \in [\text{intro}(v) .. \text{forget}(v) - 2]$, the vertices in the set

$$Y_v^j := \{t_v^j\} \cup \overline{N}_v^j \cup \{t_v^{j+1}\} \cup N_v^{j+1}$$

are subject to a relation $\overline{R}$ which is defined shortly.

- For all $v \in V_2$, the vertices in the set

$$Y_v^{\text{forget}(v)-1} := \{t_v^{\text{forget}(v)-1}\} \cup \overline{N}_v^{\ell} \cup \{t_v^{\text{intro}(v)}\} \cup N_v^{\text{intro}(v)}$$

are also subject to the relation $\overline{R}$.

Note that this step connects managers from two non-consecutive bags.

This concludes the definition of the vertices, edges and gadgets. It remains to define the relations.

Defining the Relations. In order to define the relations, we first introduce some notation to conveniently denote the states of the information vertices.

Given a selection S of vertices from G_ϕ , for each $v \in V$ and $j \in [\text{intro}(v) .. \text{forget}(v) - 2]$, we define two states a_v^j and $\overline{a}_v^j$ as follows. Let $T_v^j := |S \cap N_v^j|$ be the number of selected neighbors of t_v^j in B_v^j , and let $\overline{T}_v^j := |S \cap \overline{N}_v^j|$ be the number of selected neighbors of t_v^j in $\overline{B}_v^j$.

$$\begin{aligned} \text{If } t_v^j \text{ is selected, then we set } a_v^j &:= \sigma_{T_v^j} \text{ and } \overline{a}_v^j := \sigma_{\overline{T}_v^j}. \\ \text{If } w_v^j \text{ is not selected, then we set } a_v^j &:= \rho_{T_v^j} \text{ and } \overline{a}_v^j := \rho_{\overline{T}_v^j}. \end{aligned} \quad (10.4.1)$$

Analogously, we define the state $a_v^{\text{forget}(v)-1}$ for all $v \in V_1$, and the two states $a_v^{\text{forget}(v)-1}$ and $\overline{a}_v^{\text{forget}(v)-1}$ for $v \in V_2$.

As a last ingredient for the definition of the relations, we define the bijection between the assignments of the variables to the states in A . We partition A according to the σ - and ρ -states, that is, we set $A_\sigma := A \cap \mathbb{S}$ and $A_\rho := A \cap \mathbb{R}$. Moreover, we order the states according to their weight-component (see Definition 9.1.1 for the formal definition) in ascending order. That is, for two states $\tilde{\sigma}_1, \tilde{\sigma}_2 \in A_\sigma$, we have $\tilde{\sigma}_1 < \tilde{\sigma}_2$ if and only if $\vec{w}(\tilde{\sigma}_1) < \vec{w}(\tilde{\sigma}_2)$. We write $\tilde{\sigma}_1, \tilde{\sigma}_2, \dots, \tilde{\sigma}_{|A_\sigma|}$ for the ordered states of A_σ and $\tilde{\rho}_1, \tilde{\rho}_2, \dots, \tilde{\rho}_{|A_\rho|}$ for the ordering of the states in A_ρ .

One of the partitions may even be empty.

Now, we define the function $\text{enc}_0: [1 .. B] \rightarrow A$ where, for all $b \in [1 .. B]$, we set

$$\text{enc}_0(b) := \begin{cases} \tilde{\sigma}_b & b \in [1 .. |A_\sigma|], \\ \tilde{\rho}_{b-|A_\sigma|} & b \in [|A_\sigma| + 1 .. B]. \end{cases} \quad (10.4.2)$$

For example, if $A = \mathbb{R}$, then this approach yields $\text{enc}_0(b) = \rho_{b-1}$.

We define a new function enc based on enc_0 that inverts the result of enc_0 when at least one of the sets σ or ρ is simple cofinite. We use the inverse of a state from Definition 10.1.6 for the definition.

- If σ is simple cofinite, then we set $\text{enc}(b) := \text{inv}^\sigma(\text{enc}_0(b))$.

- If ρ is simple cofinite, then we set $\text{enc}(b) := \text{inv}^\rho(\text{enc}_0(b))$.

Note that both cases might apply as σ and ρ can both be simple cofinite. As, by assumption, the set A is closed under inversion with respect to σ and ρ , we get $\text{enc}(b) \in A$.

By definition, we have $B = |A|$, and thus, function enc_0 is a bijection. Moreover, as the inverse is a bijective function, our final mapping enc is a bijection, too. Hence, we write enc_0^{-1} and enc^{-1} for the inverse of the functions enc_0 and enc , respectively.

Now we have everything ready to define the relations R^j . For this, we fix some $j \in [1..m]$.

- For now, we write $V(C_j) = \{v_1, \dots, v_k\}$ for the $k \leq q$ variables that appear in the constraint C_j .
- Then, the relation R^j accepts the selection $S \subseteq Z^j$ if and only if there is some satisfying assignment for the variables appearing in C_j and this assignment corresponds to the states of the vertices in scope,

Formally, the relation R^j accepts the selection $S \subseteq Z^j$ if and only if there is some assignment π^j for the variables in bag $\beta(j)$ that satisfies C_j such that $\text{enc}(\pi^j[V(C_j)]) = (a_{v_1}^j, \dots, a_{v_k}^j)$.

As the relation R^j observes the set Z^j , and the selection of the vertices from Z^j determines the states a_v^j , the relation has indeed access to the states a_v^j .

Now we define the relation $\bar{R}$ with scope Y_v^j .

- The relation accepts a selection $S \subseteq Y_v^j$ if and only if the state a_v^{j+1} complements the state $\bar{a}_v^j$.

Formally, the relation accepts if and only if $a_v^{j+1} = \text{inv}(\bar{a}_v^j)$. Note that therefore, vertex t_v^j is selected if and only if vertex t_v^{j+1} is selected.

This concludes the construction of G_ϕ . Next, we first prove the correctness and then, analyze the graph G_ϕ .

Correctness of the Construction. We prove both directions of the correctness separately.

Claim 10.4.4.1. *Every satisfying multi-assignment π for ϕ that is consistent on V_2 can be extended to a solution S_π for G_ϕ .*

Proof of Claim. Let π be a satisfying multi-assignment for ϕ that is consistent on V_2 . By assumption, we know that π must also be consistent on V_1 . Hence, the multi-assignment is an assignment.

We select the following vertices for the solution S_π .

- For all $j \in [1..m]$, we select the vertices of the manager M^j such that block B_v^j gives state $\text{enc}(\pi[v])$ to vertex t_v^j . Hence, the block $\bar{B}_v^j$ gives state $\text{inv}(\text{enc}(\pi[v]))$ to vertex t_v^j . Such a unique solution exists by the definition of the A -manager.
- For all $v \in V_1$, we select the vertices in $N_v^{\text{forget } v-1}$ according to the state $\text{enc}(\pi[v])$.

- We extend this selection to the managers and providers of G_ϕ . Especially, if a vertex in $N_v^{\text{forget } v-1}$ is selected, we use the $\sigma_{\min}$ -state and otherwise, we use the $\rho_{\min}$ -state.

In order to show that this selection of vertices is a feasible solution of (σ, ρ) -DOMSET^{Rel}, we have to verify that

1. all vertices of G_ϕ have a feasible number of selected neighbors, according to their own selection status,
2. for each $j \in [1 .. m]$, the relation R^j is satisfied,
3. all relations $\bar{R}$ are satisfied.

Let z be a vertex in G_ϕ . If z is a non-portal vertex of a $\{\sigma_{\min}, \rho_{\min}\}$ -provider or a manager, then, by the definition of such providers and managers (see Definitions 8.1.10 and 10.3.1), vertex z has a feasible number of selected neighbors and, as it is not a portal or distinguished vertex, it does not have any neighbors outside the respective gadget.

If vertex z is a vertex from some set N_v^j , then it is a portal of a $\{\sigma_{\min}, \rho_{\min}\}$ -provider. As such a vertex does not have any other neighbors (except for relations), it obtains a valid number of neighbors.

In the remaining case, vertex z corresponds to some distinguished vertex w_v^j of a manager M^j . By the definition of a manager from Definition 10.3.1, the distinguished vertices obtain a feasible number of neighbors within M^j . This concludes the first property.

Now, consider a constraint C_j of ϕ for $j \in [1 .. m]$. By our definition of the solution, the vertices t_v^j obtain state $\text{enc}(\pi[v])$ from the manager. Since π is a satisfying assignment, the assignment $\pi[\beta(j)]$ satisfies C_j and proves the second property.

By our selection of the vertices, we know that for each variable v , the states a_v^j are identical for all j . Hence, we especially have that $\bar{a}_v^j = \text{inv}(a_v^j)$. Therefore, the relations $\bar{R}$ are satisfied as they check precisely this requirement. Moreover, this implies that the vertices t_v^j are either all selected to all not selected. $\triangleleft$

As a next step, we show the reverse direction of the correctness.

Claim 10.4.4.2. *For every solution for G_ϕ , there is a unique monotone satisfying multi-assignment for ϕ that is consistent on V_2 .*

Proof of Claim. Consider a solution S for the instance G_ϕ of (σ, ρ) -DOMSET^{Rel}. Recall that the vertex set S determines the numbers T_v^ℓ and $\bar{T}_v^\ell$ and the states a_v^ℓ and $\bar{a}_v^\ell$ as described in Equation (10.4.1).

We define the following multi-assignment π for ϕ . For all $j \in [1 .. m]$ and all variables $v \in \beta(j)$, we set $\pi(j, v) := \text{enc}^{-1}(a_v^j)$.

We first claim that π satisfies ϕ . Indeed, by the definition of the relations R^j , the selected vertices in S are only accepted if the states $a_\star^j$ correspond to an accepting assignment for C_j .

It remains to check that π is monotone on all variables and consistent on V_2 . We fix a variable $v \in V$ for this. By the definition of the relations $\bar{R}$, we know that

$$a_v^{j+1} = \text{inv}(\bar{a}_v^j) \quad \text{for all } j \in [\text{intro}(v) .. \text{forget}(v) - 2].$$

This especially implies that t_v^j is selected if and only if t_v^{j+1} is selected.

Now assume that t_v^j is selected and σ is finite. The case when t_v^j is not selected and ρ is finite, follows analogously. As the set σ is finite, we also know that t_v^j cannot be adjacent to more than s_{top} selected neighbors. Therefore, we have

$$T_v^j + \bar{T}_v^j \leq s_{\text{top}} \quad \text{for all } j \in [\text{intro}(v) .. \text{forget}(v) - 2] \quad (10.4.3)$$

which is equivalent to $T_v^j \leq s_{\text{top}} - \bar{T}_v^j$. By the definition of enc based on enc_0 as a monotone function when restricted to the σ -states, we know from Equation (10.4.2) and our previous observation that

$$\begin{aligned} \pi(j, v) &= \text{enc}_0^{-1}(a_v^j) = \text{enc}_0^{-1}(\sigma_{T_v^j}) \\ &\leq \text{enc}_0^{-1}(\sigma_{s_{\text{top}} - \bar{T}_v^j}) = \text{enc}_0^{-1}(\text{inv}(\sigma_{\bar{T}_v^j})) \\ &= \text{enc}_0^{-1}(\text{inv}(\bar{a}_v^j)) = \text{enc}_0^{-1}(a_v^{j+1}) \\ &= \pi(j+1, v). \end{aligned}$$

Now assume that t_v^j is not selected and ρ is simple cofinite. The case when t_v^j is selected and σ is simple cofinite follows analogously. As the set ρ is simple cofinite, vertex t_v^j must be adjacent to at least $\min \rho$ selected vertices. Moreover, we know that $\min \rho = r_{\text{top}}$ because the set is *simple* cofinite. Therefore, we have

$$T_v^j + \bar{T}_v^j \geq r_{\text{top}} \quad \text{for all } j \in [\text{intro}(v) .. \text{forget}(v) - 2] \quad (10.4.4)$$

which is equivalent to $r_{\text{top}} - T_v^j \leq \bar{T}_v^j$. Now we observe that enc differs from enc_0 because of the set ρ . More precisely, we know that $\text{enc}^{-1}(a_v^\ell) = \text{enc}_0^{-1}(\text{inv}(a_v^\ell))$. Combined with the definition of enc_0 from Equation (10.4.2) as a monotone function when restricted to the ρ -states and our previous observation, we have

$$\begin{aligned} \pi(\ell, v) &= \text{enc}^{-1}(a_v^\ell) = \text{enc}_0^{-1}(\text{inv}(a_v^\ell)) \\ &= \text{enc}_0^{-1}(\text{inv}(\rho_{T_v^\ell})) = \text{enc}_0^{-1}(\rho_{r_{\text{top}} - T_v^\ell}) \\ &\leq \text{enc}_0^{-1}(\rho_{\bar{T}_v^\ell}) = \text{enc}_0^{-1}(\bar{a}_v^\ell) \\ &= \text{enc}_0^{-1}(\text{inv}(a_v^{\ell+1})) = \text{enc}^{-1}(a_v^{\ell+1}) \\ &= \pi(\ell+1, v). \end{aligned}$$

Together with the first case, this proves that the assignment is monotone on all variables.

It remains to argue that the assignment is consistent for all $v \in V_2$. To see this, we observe that there is also a relation $\bar{R}$ where the scope contains $\{t_v^{\text{forget}(v)-1}\} \cup N_v^{\text{forget}(v)-1}$ together with $\{t_v^{\text{intro}(v)}\} \cup N_v^{\text{intro}(v)}$. Hence, we additionally obtain $a_v^{\text{intro}(v)} = \text{inv}(\bar{a}_v^{\text{forget}(v)-1})$. Together with the monotonicity of the variables, we conclude that π must be consistent on V_2 . $\blacktriangleleft$

By assumption, we know that every satisfying monotone multi-assignment for ϕ that is consistent on V_2 is also consistent on V_1 . Combined with the construction of the solutions in the previous proofs, we obtain that the construction is parsimonious.

Claim 10.4.4.3. *There is a one-to-one correspondence between the satisfying multi-assignment for ϕ that are consistent on V_2 and the solutions for G_ϕ .*

Moreover, in all solutions for G_ϕ , each vertex t_v^j has exactly s_{top} neighbors if it is selected, and exactly r_{top} neighbors if it is not selected.

This finishes the correctness of the construction. As a last step, we analyze the graph and its pathwidth. Recall that we denote by q the size of the largest scope of a constraint of ϕ .

Claim 10.4.4.4. *There is some function f that depends only on s_{top} and r_{top} such that the graph G_ϕ has size at most $f(t_{\text{top}}) \cdot |\phi|^{\mathcal{O}(1)}$ vertices and arity at most $q \cdot f(t_{\text{top}})$. Moreover, we can efficiently construct a path decomposition of G_ϕ of width $\text{pw}(\phi) + f(t_{\text{top}}) \cdot |A| \cdot q \cdot \log |V_1|$.*

Proof of Claim. Every vertex in G_ϕ is part of (at least)

- one of the m different A -managers of rank at most $\text{pw}(\phi) + 1$, or
- one of the $|V_1| \cdot (t_{\text{top}} + 1)$ copies of a $\{\sigma_{s_{\text{min}}}, \rho_{r_{\text{min}}}\}$ -provider.

We analyze the number of vertices in these components.

- By Definition 10.3.1, the size of an A -manager of rank k is bounded by $k \cdot b$, where b is an upper bound on the size of the blocks. Especially, the value of b depends only on s_{top} and r_{top} .
- By Lemma 10.4.2, the size of the parsimonious $\{\sigma_{s_{\text{min}}}, \rho_{r_{\text{min}}}\}$ -providers depends only on s_{min} and r_{min} .

Recall that the arity of the graph is defined as the largest degree of any relation in the graph. Hence, the arity is upper bounded by the size of the sets Z^j (as we can assume that $q \geq 2$) or some function depending on the size of the blocks from the manager. Recall that

$$Z^j = \bigcup_{v \in V(C_j)} (\{t_v^j\} \cup N_v^j).$$

The size of the sets N_v^j is bounded by that of the blocks of the A -manager of rank at most $\text{pw}(\phi) + 1$. So, once again, their size depends only on s_{top} and r_{top} . So, there is some function f' such that, for each i and j , we have $|Z^j| \leq q \cdot f'(s_{\text{top}}, r_{\text{top}})$. This proves the desired bound on the arity of G_ϕ .

So for the bound on the size of G_ϕ it suffices to bound the number of vertices plus the number of relations. We have already bounded the number vertices. For the number of relations, note that there are m relations of the form R^j , at most $m \cdot (\text{pw}(\phi) + 1)$ relations $\bar{R}$, and at most $|V_1| \cdot (t_{\text{top}} + 1)$ relations from the parsimonious $\{\sigma_{s_{\text{min}}}, \rho_{r_{\text{min}}}\}$ -providers. This proves the bound on the size of G_ϕ .

We use a node search strategy to bound the pathwidth of G_ϕ (see, for example, Section 7.5 in [36] for further details).

For a bag $j \in [1..r]$, we write $v_{j,1}, \dots, v_{j,|\beta(j)|}$ for an ordering of the vertices in $\beta(j)$ according to the ordering of the corresponding blocks of the manager M^j . For ease of notation, we set $v_j := v_{j,|\beta(j)|}$ for the last vertex in this sequence.

We assume that the first bag $\beta(0)$ is empty. Hence, we have vacuously placed searchers on the vertices $\{t_v^0\}_{v \in \beta(0)}$.

Then, we clean the rest of the graph in stages where we start with stage 1. Stage j starts if the searchers are placed on the vertices $\{t_v^{j-1}\}_{v \in \beta(j-1)} \cup B_{v_{j-1}}^{j-1}$. We first place searchers on the vertices in Z^j when constraint C_j is assigned to bag $\beta(\ell)$. Next, we iterate over all $i \in [1 \dots |\beta(j)|]$. We place a searcher on $t_{v_{j,i}}^j$ and place searchers on $\bar{B}_{v_{j,i}}^{j-1}$ and $B_{v_{j,i}}^j$. Note that this covers all vertices in the scope $Y_{v_{j,i}}^{j-1}$ of relation $\bar{R}$. Then, we remove the searchers from $\bar{B}_{v_{j,i-1}}^{j-1}$ and $B_{v_{j,i-1}}^j$, and place them on $\bar{B}_{v_{j,i+1}}^{j-1}$ and $B_{v_{j,i+1}}^j$. Then, we remove the searcher from $t_{v_{j,i}}^j$ unless $v_{j,i} \in V_2$ and $j = \text{intro}(v_{j,i})$. However, if $v_{j,i} \in V_2$ and $j = \text{forget}(v_{j,i}) - 1$, then we also remove the searcher from $j_{v_{j,i}}^{\text{intro}(v_{j,i})}$.

Then, we proceed to round $i + 1$. Once we reach $i + 1 > |\beta(j)|$, we do not remove the searchers from $B_{v_j}^j$ but remove the searchers from $B_{v_{j-1}}^{j-1}$. Additionally, we remove the searchers from Z^j and proceed to the next stage $j + 1$.

By assumption, each bag of the primal graph of ϕ contains at most $\mathcal{O}((|A| + 1) \cdot \log|V_1|)$ vertices from V_2 . Hence, the pathwidth of the instance is bounded by

$$(\text{pw}(\phi) + 1) + \mathcal{O}(b) + \mathcal{O}(B \cdot \log|V_1| \cdot b).$$

As the size b of a block only depend on t_{top} , the claim follows.

We observe that for each relational constraint, there is some round in some stage where the entire scope of the constraint is covered (that is, all vertices in the scope are in the same bag of the decomposition). $\blacktriangleleft$

No we have everything ready to conclude the proof.

Finalizing the Proof. We now use the previous construction to solve q -CSP- B faster assuming an algorithm for (σ, ρ) -DOMSET^{Rel} is given that beats the lower bound stated in Main Lemma 10.4.3. We fix an $\varepsilon > 0$ for this and denote by q the value from Conjecture 4.1.11 such that q -CSP- $|A|$ cannot be solved in time $(|A| - \varepsilon)^w \cdot |\psi|^{\mathcal{O}(1)}$ for an instance ψ that is given with a path decomposition of width w and the additional assumptions as stated in Conjecture 4.1.11.

Now consider precisely such a q -CSP- $|A|$ instance ϕ and denote by G_ϕ the graph obtained by our earlier construction.

Assume there is an algorithm that solves (σ, ρ) -DOMSET^{Rel} on instances of size N given a path decomposition of width at most w in time $(|A| - \varepsilon)^{w + \mathcal{O}(1)} \cdot N^{\mathcal{O}(1)}$.

We apply this algorithm to the instance G_ϕ which we just constructed and recover the result for the CSP-instance ϕ . Including the construction of G_ϕ , by Claim 10.4.4.4, this procedure takes time

$$\begin{aligned} & |\phi|^{\mathcal{O}(1)} + (|A| - \varepsilon)^{\text{pw}(G_\phi)} \cdot |G_\phi|^{\mathcal{O}(1)} \\ & \leq (|A| - \varepsilon)^{\text{pw}_\phi + f(t_{\text{top}}) \cdot |A| \cdot q \cdot \log|V_1|} \cdot |\phi|^{\mathcal{O}(1)} \\ & \leq (|A| - \varepsilon)^{\text{pw}_\phi} \cdot |V_1|^{f(t_{\text{top}}) \cdot |A| \cdot q \cdot \log|A|} \cdot |\phi|^{\mathcal{O}(1)}. \end{aligned}$$

We observe that $|A| \leq |\mathbb{A}|$, and thus, $|A|$ can be bounded in terms of s_{top} and r_{top} . Moreover, the value of q depends only on ε , and the term t_{top} depends only on σ and ρ . Therefore, the

We keep the searcher on $t_{v_{j,i}}^{\text{intro}(v_{j,i})}$ until we forget the variable $v_{j,i}$. This is needed to avoid that vertices get contaminated again as the vertices associated to the variables in V_2 form a cycle in the graph.

contribution to the running time can be bounded by the polynomial term $|\phi|^{\mathcal{O}(1)}$, and we can solve q -CSP- $|A|$ in time

$$(|A| - \varepsilon)^{\text{pw}_\phi} \cdot |\phi|^{\mathcal{O}(1)}.$$

This running time directly contradicts MONO-pw-CSPH for the decision version and MONO-pw-#CSPH for the counting version as stated in Conjecture 4.1.11 and Conjecture 4.1.12, respectively, and thus, concludes the proof of Main Lemma 10.4.3 and Main Lemma 10.4.4. $\square$

10.5 Counting Version: Extension to Cofinite Sets

The intermediate lower bounds from Section 10.4 cover the case when σ and ρ are finite or *simple* cofinite sets. In the following we extend this result to arbitrary cofinite sets in the case of the counting version. Formally, our goal is to prove Main Lemma 10.5.8.

Main Lemma 10.5.8. *Consider two non-empty sets σ and ρ that are finite or cofinite with $\rho \neq \{0\}$.*

For all sets $\hat{\sigma}$ that are finite with $\sigma = \hat{\sigma}$ or simple cofinite with $\text{top } \sigma = \text{top } \hat{\sigma}$, and for all sets $\hat{\rho}$ that are finite with $\rho = \hat{\rho}$ or simple cofinite with $\text{top } \rho = \text{top } \hat{\rho}$, the following holds:

There is a pa-reduction from $(\hat{\sigma}, \hat{\rho})$ -#DOMSET^{Rel} to (σ, ρ) -#DOMSET^{Rel}.

We prove this reduction in multiple steps that mainly consist of small and local changes to the graph. Additionally, we exploit the expressiveness of the counting version to include weights via interpolation techniques. These steps closely follow the ideas from Section 7.4 for a similar result for #GENFAC.

We consider the following steps and refer to Figure 10.8 for an illustration of this sequence.

For the first two steps, we use weighted relations. That is, the relations do not only accept (or reject) but they also contribute by some weight to the total solution. We formalize this concept in Definitions 10.5.1 and 10.5.2.

1. As a first step, we allow that σ can be an arbitrary cofinite set. We make each vertex v a portal of s_{top} copies of a $\{\sigma_0, \sigma_1, \rho_0\}$ -provider and add a weighted relation that observes v and the providers. Depending on the number of selected neighbors which vertex v already has, there are only limited number of extensions to the providers. As these extensions provide a “fingerprint” for the number of neighbors, the relation ensures that only those extensions are valid that are valid in the original graph.

We formalize this idea by the pathwidth-preserving reduction in Lemma 10.5.3.

2. For the second step, we allow that ρ can now be an arbitrary cofinite set. We use essentially the same approach as for the first step except that we attach a different provider. More precisely, we use a $\{\sigma_0, \rho_0, \rho_1\}$ -provider instead.

We formalize this step in Lemma 10.5.4.

3. For the third step, we introduce a vertex-weighted version of (σ, ρ) -#DOMSET^{Rel}. For this problem, a vertex contributes by its weight to the solution whenever it is selected. We provide the formal definition in Definition 10.5.5.

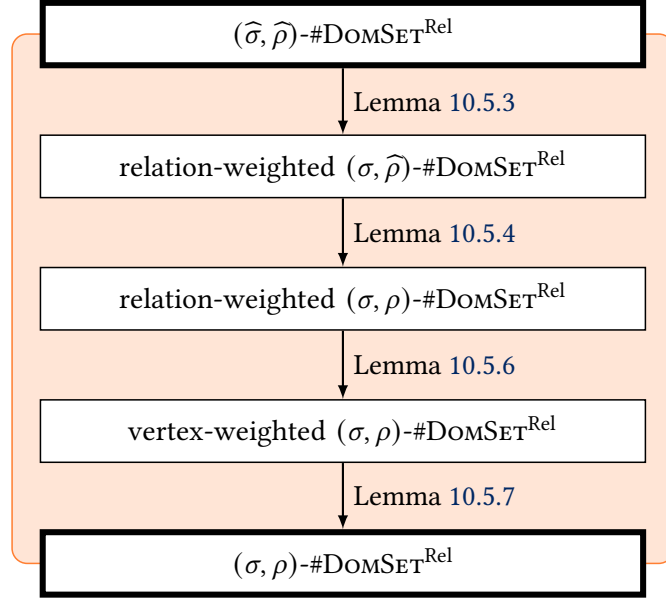


Figure 10.8: The chain of reductions used in the proof of Main Lemma 10.5.8, that is, the reduction from $(\widehat{\sigma}, \widehat{\rho})\text{-}\#\text{DOMSET}^{\text{Rel}}$ where $\widehat{\sigma}$ and $\widehat{\rho}$ are simple cofinite sets to $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{Rel}}$ where σ and ρ are cofinite sets that are not simple cofinite with $\text{top } \sigma = \text{top } \widehat{\sigma}$ and $\text{top } \rho = \text{top } \widehat{\rho}$. The other cases where at least one set is finite or simple cofinite are easier as the first and/or second step of the reduction can be omitted.

The idea of the reduction is quite simple; for each weight of a weighted relation, we create a weighted vertex with precisely this weight. Then, we ensure that this vertex is selected whenever the relations accepts with the corresponding weight.

We formalize this reduction in Lemma 10.5.6.

4. As last step, we remove the weighted vertices. For this step, we exploit that for the counting version two solutions of weight 1 each behave essentially the same as a single solution with weight 2. We define gadgets that simulate some weight and then use polynomial interpolation techniques to recover the original solution.

We formalize this Turing-reduction in Lemma 10.5.7.

To formally state the reductions, we first generalize the definition of graphs with relations to include weights.

Definition 10.5.1. A tuple $G = (V, E, \mathcal{C})$ is a graph with weighted relations if G is defined as in Definition 10.1.1 except that, each constraint $C \in \mathcal{C}$ is now a function $f_C: 2^{\text{scp}(C)} \rightarrow \mathbb{Q}$.

The definitions of pathwidth and arity of a graph with weighted relations transfer directly from Definition 10.1.1.

With this notation, we define the *relation-weighted* version of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{Rel}}$ as follows.

Definition 10.5.2 (q -Relation-Weighted (σ, ρ) -#DOMSET^{Rel}). Let q be a non-negative integer. In the q -relation-weighted (σ, ρ) -#DOMSET^{Rel} problem, we are given a graph with weighted relations $G = (V, E, \mathcal{C})$ such that $|\{f_C(U) \mid C \in \mathcal{C}, U \subseteq \text{scp}(C)\}| \leq q$, that is, the constraints of G use at most q different weights.

Let Ω be the set of (σ, ρ) -sets of G . The task is to compute

$$\sum_{U \in \Omega} \prod_{C \in \mathcal{C}} f_C(U \cap \text{scp}(C)).$$

Note that, for $f_C: 2^{\text{scp}(C)} \rightarrow \{0, 1\}$, this corresponds to the unweighted (σ, ρ) -#DOMSET^{Rel} problem. Therefore, unweighted (σ, ρ) -#DOMSET^{Rel} trivially reduces to 2-relation weighted (σ, ρ) -#DOMSET^{Rel}. As we allow negative weights, it is possible that solutions cancel out each other. We exploit this fact to remove unwanted solutions from the sum.

As we consider the counting version, we additionally have to ensure that we do not introduce additional solutions that might perturb the result. Therefore, we use the parsimonious providers from Lemma 10.4.2 in our constructions.

Lemma 10.4.2. Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$. For all $s \in \sigma$ and $r \in \rho$, there is a parsimonious $\{\sigma_s, \rho_r\}$ -provider.

Step 1: Cofinite σ . Now, we have everything ready to formally state the first reduction where we allow that the set σ can be arbitrary cofinite.

Lemma 10.5.3. Fix a non-empty set $\rho \subseteq \mathbb{N}$ that is finite with $\rho \neq \{0\}$. Fix two non-empty set $\sigma, \hat{\sigma} \subseteq \mathbb{N}$ where σ is cofinite and $\hat{\sigma}$ is simple cofinite such that $\text{top } \sigma = \text{top } \hat{\sigma}$. There is a non-negative, constant integer q_0 such that the following holds.

For all constants q , there is a pa-reduction from q -relation-weighted $(\hat{\sigma}, \rho)$ -#DOMSET^{Rel} to $(q + q_0)$ -relation-weighted (σ, ρ) -#DOMSET^{Rel}.

Proof. We can assume that $\sigma \neq \hat{\sigma}$ as otherwise the reduction is trivial. It follows that $\sigma \neq \mathbb{N}$, and consequently $s_{\text{top}} > 0$. We use the following auxiliary provider in the proof.

Claim 10.5.3.1. Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ with $\rho \neq \{0\}$. If σ is cofinite, then there is a parsimonious $\{\sigma_0, \sigma_1, \rho_0\}$ -provider which uses relations.

Proof of Claim. We use a set of vertices $V = \{u, w, v_1, \dots, v_{t_{\text{top}}}\}$, where u is the portal vertex, and u is adjacent only to w . For each $i \in [1 .. t_{\text{top}}]$, vertex w is adjacent to v_i , and v_i serves as the portal of a new parsimonious $\{\sigma_{s_{\text{top}}}, \rho_{r_{\text{min}}}\}$ -provider J_i , which exists by Lemma 10.4.2. We show that there is a partial solution for each state.

We set $r := \min(\rho \setminus \{0\})$, that is, $r = r_{\text{min}}$ if $0 \notin \rho$.

ρ_0 : For each $i \in [1 .. r]$, the vertex v_i is selected, which is extended to the $\sigma_{s_{\text{top}}}$ -state of J_i . The remaining vertices of V are unselected, and this selection is extended to the $\rho_{r_{\text{min}}}$ -states of $J_{r+1}, \dots, J_{t_{\text{top}}}$. Hence, the vertices $v_1, \dots, v_r$ obtain s_{top} selected neighbors from the attached $\{\sigma_{s_{\text{top}}}, \rho_{r_{\text{min}}}\}$ -providers, and the vertices $v_{r_{\text{top}}+1}, \dots, v_{t_{\text{top}}}$ obtain r_{min} selected neighbors. Moreover, w is unselected and has r selected neighbors $v_1, \dots, v_r$ and $r \in \rho$.

σ_0 : For each $i \in [1 \dots r-1]$, the vertex v_i is selected, which is extended to the $\sigma_{s_{\text{top}}}$ -state of J_i . The vertex u is also selected. The remaining vertices in V are unselected, and this is extended to the $\rho_{r_{\text{min}}}$ -states of $J_r, \dots, J_{t_{\text{top}}}$. The vertex w is unselected and has r selected neighbors, that is, the vertices $v_1, \dots, v_{r-1}$ and u . The vertices $v_1, \dots, v_{t_{\text{top}}}$ again obtain a feasible number of selected neighbors from the attached $\{\sigma_{s_{\text{top}}}, \rho_{r_{\text{min}}}\}$ -providers.

σ_1 : Each vertex in V is selected, and this is extended to the $\{\sigma_{s_{\text{top}}}, \rho_{r_{\text{min}}}\}$ -providers $J_1, \dots, J_{t_{\text{top}}}$. Each vertex in $V \setminus \{u\}$ is selected with $s_{\text{top}} + 1$ selected neighbors, which is feasible since σ is cofinite.

The state ρ_1 could also be obtained by selecting all of $V \setminus \{u\}$.

We add a relation that observes all vertices of V . The relation ensures that exactly three partial solutions are feasible, each corresponding to one of the mentioned states. As the $\{\sigma_{s_{\text{top}}}, \rho_{r_{\text{min}}}\}$ -providers are also parsimonious, all other potential partial solutions are ruled out. $\triangleleft$

Let G be the graph from an instance of $(\hat{\sigma}, \rho)$ -#DOMSET^{Rel}. Observe that every $(\hat{\sigma}, \rho)$ -set is also a (σ, ρ) -set, but the converse is not necessarily true as in a (σ, ρ) -set a selected vertex might also have fewer than s_{top} selected neighbors. We modify G with the goal of modeling the situation where no selected vertex can receive fewer than s_{top} selected neighbors. To this end, we intend to use weighted relations to cancel out unwanted selections. The underlying idea is to add neighbors to each vertex $v \in V(G)$ that can be selected almost arbitrarily. Based on the set σ , some selections in which v receives fewer than s_{top} selected neighbors are allowed and some are not. We add weights based on the number of selected neighbors of v such that in total the vertex behaves as if it had at least s_{top} selected neighbors.

Formally, we apply the following modification for each vertex $v \in V(G)$. To simplify notation, set $Z := \{\sigma_0, \sigma_1, \rho_0\}$. Note that, by Claim 10.5.3.1, a Z -provider exists if we allow additional relations which is the case in our setting. We introduce s_{top} copies of the Z -provider with v as the portal vertex. Let $W = \{w_1, \dots, w_{s_{\text{top}}}\}$ be the set of neighbors of v in these copies, that is, the set W contains the copies of vertex w from each Z -provider.

We add a relation R observing $v, w_1, \dots, w_{s_{\text{top}}}$ that only allows the following selections: if v is unselected, then all vertices in W are unselected; if v is selected, then R allows a selection for each $\gamma \in [0 \dots s_{\text{top}}]$ such that $w_1, \dots, w_\gamma$ are selected and $w_{\gamma+1}, \dots, w_{s_{\text{top}}}$ are unselected. Moreover, if exactly the vertices $v, w_1, \dots, w_\gamma$ are selected, then the relation accepts with a weight f_γ which we define in a moment.

Let us first check that selecting vertices according to R actually can be extended to partial solutions. If v and all vertices in W are unselected, then we obtain a partial solution if all s_{top} copies of the Z -provider are in state ρ_0 . Similarly, if v is selected, the first γ w_i 's are selected, and the remaining w_i 's are unselected, then we obtain a partial solution if the first γ Z -providers are in state σ_1 , and the remaining Z -providers are in state σ_0 .

Now let us define the weights. Note that if v is selected and has α selected neighbors in the original graph (ignoring the vertices in W), then by the σ -constraint and the definition of R , there is some γ with $\alpha + \gamma \in \sigma$ such that from W precisely $w_1, \dots, w_\gamma$ are selected. We

now aim to define weights f_i , for $i \in [0 \dots s_{\text{top}}]$, such that valid weights sum up to 0 whenever $\alpha < s_{\text{top}}$. Formally, we want, for all $\alpha \in [0 \dots s_{\text{top}}]$, that

$$\sum_{\substack{\gamma=0: \\ \alpha+\gamma \in \sigma}}^{s_{\text{top}}} f_{\gamma} = \begin{cases} 1 & \text{if } \alpha \geq s_{\text{top}}, \\ 0 & \text{else.} \end{cases} \quad (10.5.1)$$

Claim 10.5.3.2. *We can efficiently find the values for the f_i 's such that they satisfy the constraints from Equation (10.5.1).*

Proof of Claim. Observe that there are $s_{\text{top}} + 1$ unknown values and $s_{\text{top}} + 1$ constraints. Recall that $s_{\text{top}} > 0$, and therefore, $s_{\text{top}} - 1 \notin \sigma$. From Equation (10.5.1), the sums for $\alpha = s_{\text{top}}$ and $\alpha = s_{\text{top}} - 1$ differ by exactly one weight, namely f_0 . Hence, we can determine the value for f_0 and eliminate it from the constraints. Repeating this procedure yields a solution for the equations and provides the values for all f_i . $\triangleleft$

Note that there are at most $s_{\text{top}} + 1$ different weights. Hence, the above reduction constructs an instance of $q + s_{\text{top}} + 1$ -relation weighted (σ, ρ) -#DOMSET^{Rel} whenever the input was an instance of q -relation weighted $(\hat{\sigma}, \rho)$ -#DOMSET^{Rel}. Thus, we have $q_0 := s_{\text{top}} + 1$.

It remains to show that the reduction is pathwidth-preserving. The relation R has a scope of size $s_{\text{top}} + 1$. Hence, we can duplicate one bag containing v and add all s_{top} vertices $w_1, \dots, w_{s_{\text{top}}}$ to that bag. Then, the scope of the relation R is contained in one bag. As σ is fixed, s_{top} is a constant, and hence, the pathwidth increases by a constant only.

Moreover, the reduction is arity-preserving. Indeed, the existing relations are not changed and the relation R has arity at most $s_{\text{top}} + 1$ which is a constant as σ is fixed. $\square$

Step 2: Cofinite ρ . Similarly to the previous result, we show the following dual result, that is, we allow ρ to be cofinite.

Lemma 10.5.4. *Fix a non-empty set $\sigma \subseteq \mathbb{N}$. Fix two non-empty set $\rho, \hat{\rho} \subseteq \mathbb{N}$ where ρ is cofinite and $\hat{\rho}$ is simple cofinite such that $\text{top } \rho = \text{top } \hat{\rho}$. There is a non-negative constant q_0 such that the following holds.*

For all constants q , there is a pa-reduction from q -relation-weighted $(\sigma, \hat{\rho})$ -#DOMSET^{Rel} to $(q + q_0)$ -relation-weighted (σ, ρ) -#DOMSET^{Rel}.

Proof. We use the following auxiliary provider in the proof which complements the result from Claim 10.5.3.1.

Claim 10.5.4.1. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$. If ρ is cofinite, then there is a parsimonious $\{\sigma_0, \rho_0, \rho_1\}$ -provider which uses relations.*

Proof of Claim. Let u be the portal vertex, which is adjacent to a new vertex w which serves as the portal of a parsimonious $\{\sigma_{s_{\text{min}}}, \rho_{r_{\text{top}}}\}$ -provider from Lemma 10.4.2. For states σ_0 and ρ_0 , vertex w is unselected which we can extend to the $\{\sigma_{s_{\text{min}}}, \rho_{r_{\text{top}}}\}$ -provider. Hence, vertex w has at least r_{top} neighbors and ρ is cofinite. For state ρ_1 , vertex w is selected which we can again extend to the $\{\sigma_{s_{\text{min}}}, \rho_{r_{\text{top}}}\}$ -provider. Hence, vertex w has s_{min} neighbors as u is not selected.

We add a relation which ensures that there are exactly three partial solutions, each corresponding to one of the mentioned states. By this, all other potential partial solutions are ruled out. $\triangleleft$

The proof follows precisely the same idea as before. The most notable difference is that we now use Z -providers with $Z = \{\sigma_0, \rho_0, \rho_1\}$, that is, the state σ_1 is not allowed, but the state ρ_1 is allowed. Observe that this is not an issue as we never need to add neighbors to v if the vertex is selected, and actually the relation R forbids to add selected neighbors to v in this case.

Then, the remainder of the proof is analogous to that of Lemma 10.5.3. $\square$

Step 3: From Weighted Relations to Weighted Vertices. In the next step, we remove the weighted relations by allowing weighted vertices. In this setting, the vertices contribute with their weight to the solution whenever they are selected. We refer to this new variant as *vertex-weighted* (σ, ρ) -#DOMSET^{Rel}.

Definition 10.5.5 (q -Vertex-Weighted (σ, ρ) -#DOMSET^{Rel}). *Let q be a non-negative integer. In the q -vertex-weighted (σ, ρ) -#DOMSET^{Rel} problem, we are given a graph with relations $G = (V, E, \mathcal{C})$ and a weight function $\text{wt}: V \rightarrow \mathbb{Q}$ such that $|\{\text{wt}(v) \mid v \in V\}| \leq q$, that is, there are at most q different vertex weights.*

Let Ω be the set of (σ, ρ) -sets of G . The task is to compute

$$\sum_{U \in \Omega} \prod_{v \in U} \text{wt}(v).$$

Recall that a (σ, ρ) -set of G has to satisfy the relations of G . Therefore, this requirement appears only implicitly in the formula. Note that to obtain the original (σ, ρ) -#DOMSET^{Rel} problem we can set $\text{wt}(v) = 1$ for all $v \in V$.

Now, we can formally state the next step of the reduction. This step crucially relies on the fact that only a constant number of different weights are used, as otherwise the pathwidth of the used construction would be too large.

Lemma 10.5.6. *For all constants q , there is a pa-reduction from q -relation-weighted (σ, ρ) -#DOMSET^{Rel} to q -vertex-weighted (σ, ρ) -#DOMSET^{Rel}.*

Proof. For a fixed q , let $w_1, \dots, w_{q'}$ be the $q' \leq q$ different weights used by the relations of a q -relation-weighted (σ, ρ) -#DOMSET^{Rel} instance G . Formally, we have $\{w_1, \dots, w_{q'}\} = \{f_C(U) \mid C \in \mathcal{C}, U \subseteq \text{scp}(C)\}$. We assume without loss of generality that $q' = q$ in the following.

We apply the following modification for all weighted constraints in $\mathcal{C}$. Let f_C be the weight function of some constraint $C \in \mathcal{C}$. We introduce q new vertices $v_1, \dots, v_q$. Each of these vertices is the portal of a parsimonious $\{\sigma_{\min}, \rho_{\min}\}$ -provider with relations, which exists by Lemma 10.4.2. For all $i \in [1 \dots q]$, we assign weight w_i to vertex v_i . Moreover, we replace the constraint C by an (unweighted) relational constraint C' with $\text{scp}(C') := \text{scp}(C) \cup \{v_1, \dots, v_q\}$. The new relation $R(C')$ accepts a selection $U \subseteq \text{scp}(C')$ if and only if

- the corresponding weight $f_C(U \cap \text{scp}(C))$ is nonzero,
- the vertex v_i with $w_i = f_C(U \cap \text{scp}(C))$ is selected, and

- all v_j with $i \neq j$ are unselected.

It remains to argue that the reduction increases the pathwidth and the arity only by a constant. By assumption, there is a bag containing $\text{scp}(C)$. We duplicate this bag, make it a neighbor of the original bag in the path decomposition, and add the new vertices $v_1, \dots, v_q$ to the copy which increases the size of the bag by q . For each v_i (one after the other), we duplicate this new bag and also add all the vertices from its attached $\{\sigma_{\min}, \rho_{\min}\}$ -provider. Observe that the size of each parsimonious $\{\sigma_{\min}, \rho_{\min}\}$ -provider with relations is bounded by a constant, as the size depends only on σ and ρ , and these two sets are fixed. Hence, the width of the decomposition increases by at most an additive constant depending only on σ , ρ , and q .

To bound the arity of the new graph, observe that the arity of each relation increases by at most q , which is a constant. Moreover, the arity of the relations in the parsimonious $\{\sigma_{\min}, \rho_{\min}\}$ -provider with relations is bounded by a constant as it is clearly bounded by the size of such a provider. Hence, the reduction increases the arity of the graph at most by an additive constant. $\square$

Step 4: Simulating Weighted Vertices. The last step now finally removes the weights from the vertices, and thus, also from the instance. This procedure makes use of the following known fact about recovering multivariate polynomials.

Fact 2.1.7 ([33, Lemma 2.5]). *Let $P \in \mathbb{Q}[x_1, \dots, x_q]$ be a multivariate polynomial such that, for all $i \in [1 \dots q]$, the degree of x_i in P is bounded by $d_i \in \mathbb{N}$. Furthermore, assume we are given sets $\Xi_i \subseteq \mathbb{Q}$ for $i \in [1 \dots q]$ such that $|\Xi_i| = d_i + 1$ for all $i \in [1 \dots q]$. Consider the Cartesian product of these sets, that is,*

$$\Xi := \Xi_1 \times \dots \times \Xi_q.$$

Then, we can compute the coefficients of P with $\mathcal{O}(|\Xi|^3)$ arithmetic operations when given as input the set $\{(\xi, P(\xi)) \mid \xi \in \Xi\}$.

With this result we prove the last intermediate step.

Lemma 10.5.7. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$. For all constants q , there is a pa-reduction from q -vertex-weighted (σ, ρ) -#DOMSET^{Rel} to unweighted (σ, ρ) -#DOMSET^{Rel}.*

Proof. We use known interpolation techniques to remove the weights. Let $G = (V, E, C)$ be the graph with relations and wt the vertex weight function of a given q -vertex-weighted (σ, ρ) -#DOMSET^{Rel} instance I . Let $w_1, \dots, w_q$ be the distinct values of the vertex weights, that is, $\{w_1, \dots, w_q\} = \{\text{wt}(v) \mid v \in V\}$.

We replace each weight w_i by a variable x_i , and treat the weighted sum of solutions of I as a (unknown) polynomial P in the q variables $x_1, \dots, x_q$. Observe that there are at most n vertices, and hence, the maximum degree is at most n for each variable. Hence, if we can realize all combinations of $n + 1$ different weights for each variable x_i , then we can use Fact 2.1.7 to recover the coefficients of P in time $\mathcal{O}((n + 1)^{3q})$ which is polynomial in n as q is a constant. Then, we can output $P(w_1, \dots, w_q)$ to recover the original solution.

It remains to realize $n + 1$ different weights for each variable. For this, it suffices to realize weights of the form 2^ℓ .

Let v be the vertex for which we want to realize weight $\text{wt}(v) = 2^\ell$. For this, we introduce ℓ new vertices $v_1, \dots, v_\ell$ which are all portals of a parsimonious $\{\sigma_{\min}, \rho_{\min}\}$ -provider with relations, which exists by Lemma 10.4.2. Moreover, for each $j \in [1 \dots \ell]$, we add a relational constraint C_j with scope $\{v, v_j\}$ that ensures that if v is unselected, then v_j must also be unselected, but if v is selected, then v_j can be either selected or unselected. Whenever v is selected, this construction contributes a factor of 2^ℓ to the solution, whereas if v is unselected, it only contributes a factor of 1.

Observe that the arity of the resulting graph increases by at most a constant (or stays unchanged if it is already larger) as C_j has arity 2 and the degrees of the relations in the parsimonious $\{\sigma_{\min}, \rho_{\min}\}$ -provider are bounded by constants as the size of the provider is constant.

It is straightforward to see that this modification does not change the pathwidth too much. Indeed, for each weighted vertex v where we realize weight 2^ℓ , we pick an arbitrary bag containing v and duplicate this bag for each $j \in [1 \dots \ell]$. Then, we add v_j and the corresponding $\{\sigma_{\min}, \rho_{\min}\}$ -provider together with the relations to the j th copy of the bag containing v . By the size bound for the $\{\sigma_{\min}, \rho_{\min}\}$ -provider, the claim follows. $\square$

Combining the Results. Combining all the previous results, we obtain the reduction from (σ, ρ) -#DOMSET^{Rel} with sets that are finite or simple cofinite to the version where the sets are finite or cofinite. That is, the combination of Lemmas 10.5.3, 10.5.4, 10.5.6 and 10.5.7 directly proves Main Lemma 10.5.8.

Main Lemma 10.5.8. *Consider two non-empty sets σ and ρ that are finite or cofinite with $\rho \neq \{0\}$.*

For all sets $\hat{\sigma}$ that are finite with $\sigma = \hat{\sigma}$ or simple cofinite with $\text{top } \sigma = \text{top } \hat{\sigma}$, and for all sets $\hat{\rho}$ that are finite with $\rho = \hat{\rho}$ or simple cofinite with $\text{top } \rho = \text{top } \hat{\rho}$, the following holds:

There is a pa-reduction from $(\hat{\sigma}, \hat{\rho})$ -#DOMSET^{Rel} to (σ, ρ) -#DOMSET^{Rel}.

Proof. If both σ and ρ are finite or simple cofinite, then the results follows directly as we do not apply any reduction.

So, let us assume that at least one of σ or ρ is cofinite but not simple cofinite. In fact, let us assume that both σ and ρ are both cofinite but not simple cofinite as the other cases can be treated analogously.

Let $\hat{\sigma} \subseteq \sigma$ and $\hat{\rho} \subseteq \rho$ be simple cofinite sets such that $\text{top } \sigma = \text{top } \hat{\sigma}$ and $\text{top } \rho = \text{top } \hat{\rho}$.

Let G be a $(\hat{\sigma}, \hat{\rho})$ -#DOMSET^{Rel} instance with arity at most d_0 . Note that, by definition, G is an instance of 2-relation weighted $(\hat{\sigma}, \hat{\rho})$ -#DOMSET^{Rel}. We sequentially use the pa-reductions from Lemmas 10.5.3, 10.5.4, 10.5.6 and 10.5.7 to obtain polynomial (in n) many instances H_i of (σ, ρ) -#DOMSET^{Rel} whose solutions can be used to compute a solution for $(\hat{\sigma}, \hat{\rho})$ -#DOMSET^{Rel} on input G .

First observe that, for each intermediate problem involving weights, the number of different weights is bounded by a constant. Since the pa-reductions are pathwidth-preserving (see Observation 2.2.5), we have $\text{pw}(H_i) \leq \text{pw}(G) + \mathcal{O}(1)$ and $|H_i| \leq |G|^{\mathcal{O}(1)}$. Similarly, since all reductions are also arity-preserving, there is a constant d' , such that, for each i , we have $\text{ar}(H_i) \leq \text{ar}(G) + d'$.

We also observe that the entire reduction is operating in polynomial time and thus, is a pa-reduction which concludes the proof. $\square$

10.6 Reducing to One Type of Relations

In Sections 10.4 and 10.5, we establish (conditional) lower bounds for the intermediate problem (σ, ρ) -DOMSET^{Rel} and its counting version. In order to show that these lower bounds transfer to the original problem versions (without the relations), we show, as a first step, how to express arbitrary relations by a single type of relation.

We prove a parsimonious reduction since we use it both in the decision setting and in the counting setting. A crucial requirement for this reduction is the restriction to relations whose arity is bounded by a constant. This is also the reason why we establish the lower bounds in Main Lemmas 10.4.3 and 10.4.4 for this setting.

Main Lemma 10.6.7. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ with $\rho \neq \{0\}$. For all constants d , there is a parsimonious pathwidth-preserving reduction from (σ, ρ) -DOMSET^{Rel} on instances of arity at most d to (σ, ρ) -DOMSET^{HW=1} on instances of arity at most $2^d + 1$.*

Afterward, in Sections 10.7 to 10.9, we show how to realize HW₌₁-relations.

The idea of this reduction is as follows. For a given d -ary relation R , we wish to construct a graph with portals whose compatible language is *equivalent* to R . We call such a graph a *realization* of R , formally defined in Definition 10.6.2. Crucially, we wish to add (a realization of) a relation to a set of vertices, without invalidating the σ - and ρ -constraints of these vertices. Hence, we require that a realization does not add any *selected* neighbors to the portal vertices.

In a first step, we show how to replace arbitrary relations by gadgets using only HW₌₁-relations and EQ-relations. We formalize this step in Lemma 10.6.3. In a second step, we then remove the EQ-relations by modeling them using only HW₌₁-relations. We formalize the second step in Lemma 10.6.5.

We repeat the formal definition of these relations here for convenience.

Definition 2.1.6. *For a k -dimensional vector $x \in \{0, 1\}^k$, we define $\text{hw}(x)$ as the number of 1s in x , that is, we define $\text{hw}(x)$ as the Hamming weight of x with $\text{hw}(x) := \sum_{i=1}^k x[i]$.*

For two positive integers $i, k \in \mathbb{N}$ and a set $S \subseteq \mathbb{N}$, we define the relations

$$\text{HW}_{\in S}^{(k)} := \{x \in \{0, 1\}^k \mid \text{hw}(x) \in S\}, \quad \text{HW}_{=i}^{(k)} := \text{HW}_{\{i\}}^{(k)} \quad \text{and} \quad \text{EQ}_k := \text{HW}_{\{0, k\}}^{(k)}.$$

We say that EQ_k is the equality relation on k inputs.

We use $\text{HW}_{\in S}$ to denote the relation $\text{HW}_{\in S}^{(k)}$ when the arity k of the relation is clear from the context. Additionally, we also write $\text{HW}_{\in S}$ to denote the set of the relations $\text{HW}_{\in S}^{(k)}$ for all $k \in \mathbb{N}$.

We transfer this abuse of notation to $\text{HW}_{=i}$ and EQ.

Recall from Definition 10.1.7, that, for some set of relations $\mathcal{R}$, we write (σ, ρ) -DOMSET ^{$\mathcal{R}$} for the restriction of (σ, ρ) -DOMSET^{Rel} where only relations in $\mathcal{R}$ are used. Slightly abusing notation, we write (σ, ρ) -DOMSET^{HW=1, EQ} for the restriction to instances in which all relations are either EQ-relations or HW₌₁-relations (of any arity).

In order to ensure that certain vertices can obtain a feasible number of selected neighbors, we again use (parsimonious) $\{\sigma_s, \rho_r\}$ -providers for some $s \in \sigma$ and $r \in \rho$. Recall that said providers exist by Lemma 10.4.2. However, the proof of Lemma 10.4.2 uses arbitrary relations to obtain a parsimonious provider. As we intend to remove just these arbitrary relations, we first show in Lemma 10.6.1 how to obtain parsimonious $\{\sigma_s, \rho_r\}$ -providers that use only $\text{HW}_{=1}$ -relations. The proof is based on Lemma 10.2.1, which we restate for convenience.

Lemma 10.2.1. *Fix two non-empty sets $\sigma, \rho \subseteq \mathbb{N}$. For all $s \in \sigma$ and $r \in \rho$ with $r \geq 1$, there is a $\{\sigma_s, \rho_r\}$ -provider $(G, \{u\})$ that has disjoint (σ, ρ) -sets X and Y with $X \cup Y = V(G)$, where $u \in X$ with $|N(u) \cap X| = s$ and $|N(u) \cap Y| = r$.*

Intuitively, we introduce a complete biclique of $\text{HW}_{=1}$ -relations between the two solutions to construct the provider.

Lemma 10.6.1. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ with $\rho \neq \{0\}$. For all $s \in \sigma$ and $r \in \rho$ with $r \geq 1$, there is a parsimonious $\{\sigma_s, \rho_r\}$ -provider that uses only $\text{HW}_{=1}^{(2)}$ -relations.*

Proof. Pick arbitrary $s \in \sigma$ and $r \in \rho$ with $r \geq 1$. This is possible as both σ and ρ are non-empty, and additionally $\rho \neq \{0\}$. Then, by Lemma 10.2.1, there is a graph G with a vertex w such that there are two solutions S_0 and S_1 partitioning the vertex set of G and the first solution S_0 satisfies $w \notin S_0$ and $|N(w) \cap S_0| = r$, and the second solution S_1 satisfies $w \in S_1$ and $|N(w) \cap S_1| = s$.

Then, we make each pair of vertices $v \in S_0$ and $u \in S_1$ subject to a relation $\text{HW}_{=1}^{(2)}$. These relations ensure that in each partial solution either all vertices from S_0 are selected and none from S_1 , or all vertices from S_1 are selected and none from S_0 .

The gadget does not contain any other vertices and thus, it has precisely these two partial solutions. $\square$

Handling Arbitrary Relations. Now our goal is to replace arbitrary relations by graphs using only $\text{HW}_{=1}$ - and EQ -relations such that the “behavior of the graph” does not change. For this, we formally introduce the notion of a realization of a relation based on a graph with portals (and possibly some restricted set of relations) that “simulates” the relation.

Definition 10.6.2 (Realization of a Relation). *For a set of vertices S with $d = |S|$, let $R \subseteq 2^S$ denote a d -ary relation. For an element $r \in R$, we write x_r for the length- d vector that is σ_0 at every position $v \in r$, and ρ_0 at the remaining positions, that is,*

$$x_r[v] := \begin{cases} \sigma_0 & \text{if } v \in r, \\ \rho_0 & \text{otherwise.} \end{cases}$$

We set $L_R := \{x_r \mid r \in R\}$. Let $G = (V, E, \mathcal{C})$ be a graph with relations. Let $U = \{u_1, \dots, u_d\} \subseteq V$ be a set of portals of G .

Slightly overloading notation, we say that H realizes R if H realizes L_R (as defined in Definition 10.1.5). We say that R is realizable if there is a graph with d portals that realizes R . Such a realization is parsimonious if, for each $x \in L_R$, there is exactly one partial solution in H that witnesses x .

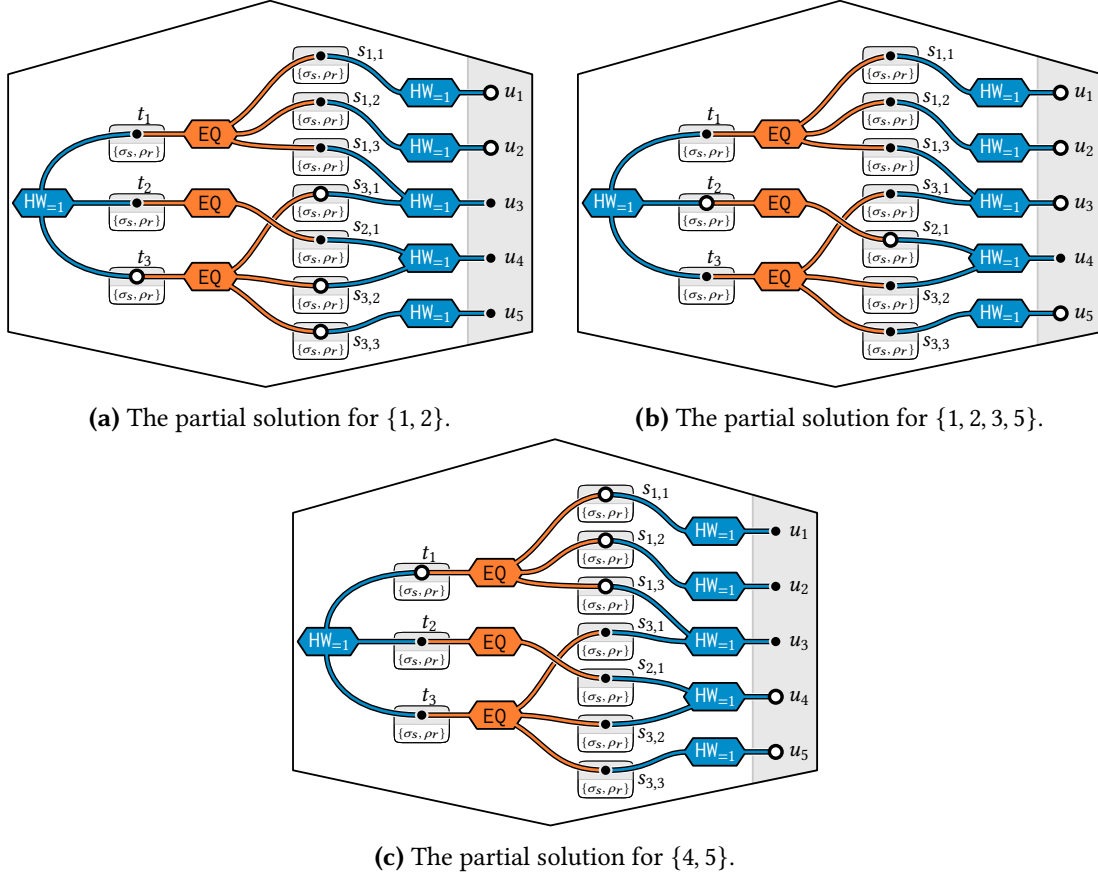


Figure 10.9: The realization of $Q = \{\{1, 2\}, \{1, 2, 3, 5\}, \{4, 5\}\}$ according to Lemma 10.6.3. We depict only the portal vertices of the parsimonious $\{\sigma_s, \rho_r\}$ -provider from Lemma 10.6.1. Further, hexagons depict relations that are realized between vertices.

With this definition at hand, we first show that we can realize arbitrary relations using $\text{HW}_{=1}$ - and EQ -relations.

We follow the basic idea of the construction by Curticapean and Marx [35, Lemma 3.3] which we also use for the corresponding result for GENFAC in Main Lemma 7.5.3. However, we need to modify the construction to account for the fact that we are selecting vertices rather than edges.

Lemma 10.6.3. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ with $\rho \neq \{0\}$, and let $Q \subseteq 2^{[1 \dots d]}$ denote an arbitrary relation. Then, Q is parsimoniously realizable by a graph G with relations and portals such that*

- *all relations used in G are $\text{HW}_{=1}$ - or EQ -relations of arity at most $2^d + 1$, and*
- *the size of G is $\mathcal{O}(2^d \cdot d)$.*

Proof. We write $Q = \{q_1, \dots, q_{|Q|}\}$ where $q_i \subseteq [1 \dots d]$. We construct a graph with portals (G, U) with $U = \{u_1, \dots, u_d\}$ as follows.

For each set q_i , we add $k_i := |[1..d] \setminus q_i|$ independent copies $(H_{i,j}, \{s_{i,j}\})$ for $j \in [1..d] \setminus q_i$ plus an extra copy $(H_i, \{t_i\})$ of a parsimonious $\{\sigma_s, \rho_r\}$ -provider with $\text{HW}_{=1}^{(2)}$ -relations to G . These providers exist by Lemma 10.6.1.

We connect the providers as follows. We put an EQ_{k_i+1} -relation between the vertices $\{s_{i,j} \mid j \in [1..d] \setminus q_i\}$ and t_i . Additionally, we put a $\text{HW}_{=1}^{(|Q|)}$ relation between the vertices $\{t_1, \dots, t_{|Q|}\}$. Finally, for each $j \in [1..d]$, we put a $\text{HW}_{=1}$ relation between u_j and all vertices of the form $s_{\star,j}$. Consult Figure 10.9 for a visualization of an example.

Claim 10.6.3.1. *The constructed graph G parsimoniously realizes Q .*

Proof of Claim. Consider some $q_i \in Q$. Then, the following selection is a solution that “witnesses” q_i . We select the portal vertices u_j with $j \in q_i$. In addition, we select t_i and the vertices $s_{i,j}$ for all $j \in [1..d] \setminus q_i$. We do not select any other vertices $u_{\star}, s_{\star,\star}$ or $t_{\star}$. Observe that, for each i and j , the parsimonious $\{\sigma_s, \rho_r\}$ -provider $(H_{i,j}, \{s_{i,j}\})$ ensures that both selecting or not selecting $s_{i,j}$ can be extended to satisfy the σ - or ρ -constraints of $s_{i,j}$, respectively. The same holds for the vertex t_i . So, vertices from the providers are selected such that all the corresponding σ - and ρ -constraints are satisfied.

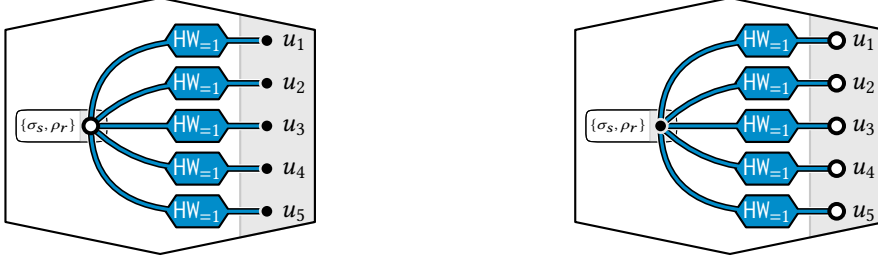
It is clear that this selection witnesses q_i . Next, we show that the selection is a solution, and for this, we need to show that our selection satisfies all relation constraints on the vertices of G that we introduced.

The vertices $\{t_1, \dots, t_{|Q|}\}$ are subject to the relation $\text{HW}_{=1}^{(|Q|)}$, and this is fine since we select only t_i from this set. Since we also select the vertices $s_{i,j}$ for $j \in [1..d] \setminus q_i$, the EQ-relation on these vertices and t_i is satisfied. For all remaining EQ relations, all corresponding vertices are unselected (which in turn also satisfies said EQ relations). Finally, for each $j \in [1..d]$, we selected precisely one of u_j if $j \in q_i$, or we select $s_{i,j}$ if $j \in [1..d] \setminus q_i$. Thus, the remaining $\text{HW}_{=1}$ relations are also satisfied. In total, we conclude that our selection is a valid partial solution for G .

In order to show the claim, it remains to show that the realization is parsimonious. Since all providers are parsimonious and the selection status of all vertices outside the providers is determined by the relations and the selection of vertices from U , the constructed partial solution is the only one that witnesses q_i .

Finally, we show that there are no partial solutions that extend selections from U that do not belong to Q . For this, assume that we are given some partial solution. As the relations are satisfied, there is exactly one vertex t_i that is selected. By the EQ relations, we also get that all vertices $s_{i,j}$ are selected for $j \in [1..d] \setminus q_i$, and that all remaining vertices $s_{\star,\star}$ are unselected. Finally, from the remaining $\text{HW}_{=1}$ relations, we get that, for each $j \in [1..d] \setminus q_i$, the portal u_j is unselected as vertex $s_{i,j}$ is already selected. Moreover, for each $j \in q_i$, the portal u_j is selected as all vertices $s_{i',j}$ are unselected in this case, as $i' \neq i$. In particular, the indices of the selected portals correspond to q_i , which completes the proof of the claim. $\blacktriangleleft$

For the bound on the size of G , observe that $|Q| \leq 2^d$, and the size of a parsimonious $\{\sigma_s, \rho_r\}$ -provider is upper-bounded by a function in r_{top} and s_{top} because of Lemma 10.6.1, which is a constant. As there are at most $2^d \cdot d$ vertices $s_{i,j}$ which are all the portal of some $\{\sigma_s, \rho_r\}$ -provider, the size of the graph is bounded by $\mathcal{O}(2^d \cdot d)$.



(a) The partial solution where no portal vertex is selected.

(b) The partial solution where all portal vertices are selected.

Figure 10.10: The realization of EQ_k from Lemma 10.6.5 here shown for $k = 5$. For the parsimonious $\{\sigma_s, \rho_r\}$ -provider we only depict the portal vertex.

Moreover, the arity of the graph is bounded by $2^d + 1$ as the $\text{HW}_{=1}$ -relations have arity at most $|Q| + 1 \leq 2^d + 1$ and the EQ -relations have arity at most $d + 1$. $\square$

We lift the previous result to the following reduction, where we replace each relation that is neither EQ nor $\text{HW}_{=1}$ by the respective realization according to Lemma 10.6.3.

Lemma 10.6.4. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ with $\rho \neq \{0\}$. For all constants d , there is a parsimonious pathwidth-preserving reduction from $(\sigma, \rho)\text{-DOMSET}^{\text{Rel}}$ on instances of arity at most d to $(\sigma, \rho)\text{-DOMSET}^{\text{HW}_{=1}, \text{EQ}}$ on instances of arity at most $2^d + 1$.*

Proof. Let G denote an instance of $(\sigma, \rho)\text{-DOMSET}^{\text{Rel}}$. We replace each relation in G that is neither a $\text{HW}_{=1}$ -relation nor an EQ -relation with the corresponding realizer from Lemma 10.6.3. Let H be the resulting $(\sigma, \rho)\text{-DOMSET}^{\text{HW}_{=1}, \text{EQ}}$ instance. Indeed, by Lemma 10.6.3, H contains only $\text{HW}_{=1}$ - or EQ -relations. Moreover, by the construction of the realization, the number of solutions is preserved, and hence, the reduction is parsimonious.

We first analyze the arity of H . We know that the arity of G is at most d . From Lemma 10.6.3, we know that the arity for each realization is at most $2^d + 1$. Hence, the arity of H itself is at most $2^d + 1$.

It remains to argue about the pathwidth of H . Assume that we are given a path decomposition of G . Let R be a relation in G which we replaced by a realization. By definition, there is a bag β in the path decomposition of G containing all vertices in the scope of R . We duplicate bag β to obtain a copy β_R and add all vertices of the realization of R to bag β_R . Observe that this is a valid path decomposition of the resulting graph of width at most $\text{pw}(H) \leq \text{pw}(G) + \mathcal{O}(2^d d) \leq \text{pw}(G) + \mathcal{O}(1)$ as d is a constant.

Note that copying the original bag for each relation separately is crucial. Indeed, we cannot simply add the vertices of all realizations to the same bag as this could increase the size of the bag by more than a constant since there might be many relations that observe the same vertex. $\square$

Handling Equality Relations. The next step is to remove the EQ -relations and replace them by realizers which use only $\text{HW}_{=1}$ -relations.

Lemma 10.6.5. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ with $\rho \neq \{0\}$. Then, for any $k \geq 1$, there is a graph G with relations and portals that parsimoniously realizes EQ_k such that*

- *all relations used in G are $\text{HW}_{=1}^{(2)}$ -relations,*
- *the size of G is $\mathcal{O}(k)$, and*
- *$G \setminus U$ has pathwidth $\mathcal{O}(1)$.*

Proof. First, observe that we can trivially realize EQ_1 by a graph that consists of a single portal vertex.

Suppose that $k \geq 2$. We construct a graph with portals (G, U) with $U = \{u_1, \dots, u_k\}$. We create a single parsimonious $\{\sigma_s, \rho_r\}$ -provider $(H, \{v\})$ with $\text{HW}_{=1}^{(2)}$ -relations, which exists by Lemma 10.6.1. Finally, we add a $\text{HW}_{=1}^{(2)}$ relation pairwise between v and each of $u_1, \dots, u_k$. We refer to Figure 10.10 for a visualization.

To see that G realizes EQ_k , first observe that if at least one portal vertex u_i is selected, then v is forced to be unselected by the $\text{HW}_{=1}^{(2)}$ -relation that observes v and u_i . This forces the remaining portal vertices to be selected. On the other hand, if no portal u_i is selected, then we can get a valid solution by selecting v .

The size bound follows trivially as we added only one parsimonious $\{\sigma_s, \rho_r\}$ -provider $(H, \{v\})$ with $\text{HW}_{=1}^{(2)}$ -relations, and k $\text{HW}_{=1}^{(2)}$ -relations. Likewise, the bound on arity and pathwidth follow. $\square$

Similar to Lemma 10.6.4, we lift this realization of EQ to the following reduction, where we replace all EQ-relations by their realization.

Lemma 10.6.6. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ with $\rho \neq \{0\}$. Then, there is a parsimonious pa-reduction from $(\sigma, \rho)\text{-DOMSET}^{\text{HW}_{=1}, \text{EQ}}$ to $(\sigma, \rho)\text{-DOMSET}^{\text{HW}_{=1}}$.*

Proof. Let G denote an instance of $(\sigma, \rho)\text{-DOMSET}^{\text{HW}_{=1}, \text{EQ}}$. We replace each EQ-relation in G by the appropriate realization, which uses $\text{HW}_{=1}$ -relations, from Lemma 10.6.5. Let H be the obtained instance of $(\sigma, \rho)\text{-DOMSET}^{\text{HW}_{=1}}$.

By Lemma 10.6.5, all relations used in H are $\text{HW}_{=1}$ -relations. Moreover, the realizations ensure that the number of solutions is preserved, and thereby, the reduction is parsimonious. Note that this reduction is now also arity-preserving as the realization of the EQ-relations has arity at most 2.

It remains to argue about the pathwidth of H . Assume we are given a path decomposition of G . Let R be a relation in G which we replaced by a realization. By the definition of the path decomposition, there is a bag β containing all vertices in the scope of R . Let $\beta(R)_1', \dots, \beta(R)_\ell$ be the bags of width at most $\mathcal{O}(1)$ from the path decomposition of the realization of R . We duplicate the bag β ℓ times and let $\beta, \beta_1, \dots, \beta_\ell$ denote the resulting copies. We combine the bags β_i and $\beta(R)_i$ into a new bag of the path decomposition of H . Observe that this is a valid path decomposition of width at most $\text{pw}(H) \leq \text{pw}(G) + \mathcal{O}(1)$. $\square$

We combine Lemmas 10.6.4 and 10.6.6 as follows to prove the reduction from $(\sigma, \rho)\text{-DOMSET}^{\text{Rel}}$ to $(\sigma, \rho)\text{-DOMSET}^{\text{HW}_{=1}}$.

Main Lemma 10.6.7. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ with $\rho \neq \{0\}$. For all constants d , there is a parsimonious pathwidth-preserving reduction from (σ, ρ) -DOMSET^{Rel} on instances of arity at most d to (σ, ρ) -DOMSET^{HW=1} on instances of arity at most $2^d + 1$.*

Proof. The reduction follows directly from Lemmas 10.6.4 and 10.6.6 and the properties of the respective reductions. $\square$

To replace the HW₌₁-relations by their realizations, we again distinguish between the decision version in Section 10.7 and the counting version in Sections 10.8 and 10.9.

10.7 Decision Version: Realizing Relations

In this section, we replace HW₌₁-relations by gadgets which do not use any relations at all. These gadgets ensure the existence of a solution that respects the relations. As we consider the decision version, the gadgets do not necessarily have to be parsimonious. Formally, we prove Main Lemma 10.7.3

Main Lemma 10.7.3. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite with $0 \notin \rho$. For all constants d , there is a pathwidth-preserving reduction from (σ, ρ) -DOMSET^{HW=1} on instances of arity at most d to (σ, ρ) -DOMSET.*

For this reduction, we again make use of the notion of a realization from Definition 10.6.2. Crucially, however, while Definition 10.6.2 defines realizers and providers as graphs with relations and portals, in this section, the constructed providers and realizers do not use relations, that is, the providers and realizers are simply graphs with portals. We refrain from stating this in each result as it is easily verifiable as the constructions do not use relations.

We start with a helpful gadget to ensure that a vertex v is selected and receives s_{top} selected neighbors in any partial solution, that is, we realize the language $\{\sigma_{s_{\text{top}}}\}$. Intuitively, this realizer forces a vertex to be selected and ensures that no neighbor of this vertex (outside the gadget) is selected.

Lemma 10.7.1. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite with $0 \notin \rho$. Then, there is a $\{\sigma_{s_{\text{top}}}\}$ -realizer.*

Proof. We start with a useful helper gadget which ensures that the portal is always selected and receives at least s_{min} selected neighbors in the gadget.

Claim 10.7.1.1. *There is a $\{\sigma_{s_{\text{min}}}\}$ -provider $(H, \{u\})$ such that $L(H, \{u\}) \subseteq \{\sigma_i \in \mathbb{S} \mid i \geq s_{\text{min}}\}$.*

Proof of Claim. We construct a graph H with a single portal vertex u that satisfies $\{\sigma_{s_{\text{min}}}\} \subseteq L(H, \{u\}) \subseteq \{\sigma_i \in \mathbb{S} \mid i \geq s_{\text{min}}\}$. We consider three cases and refer to Figure 10.11 for an illustration of the different constructions.

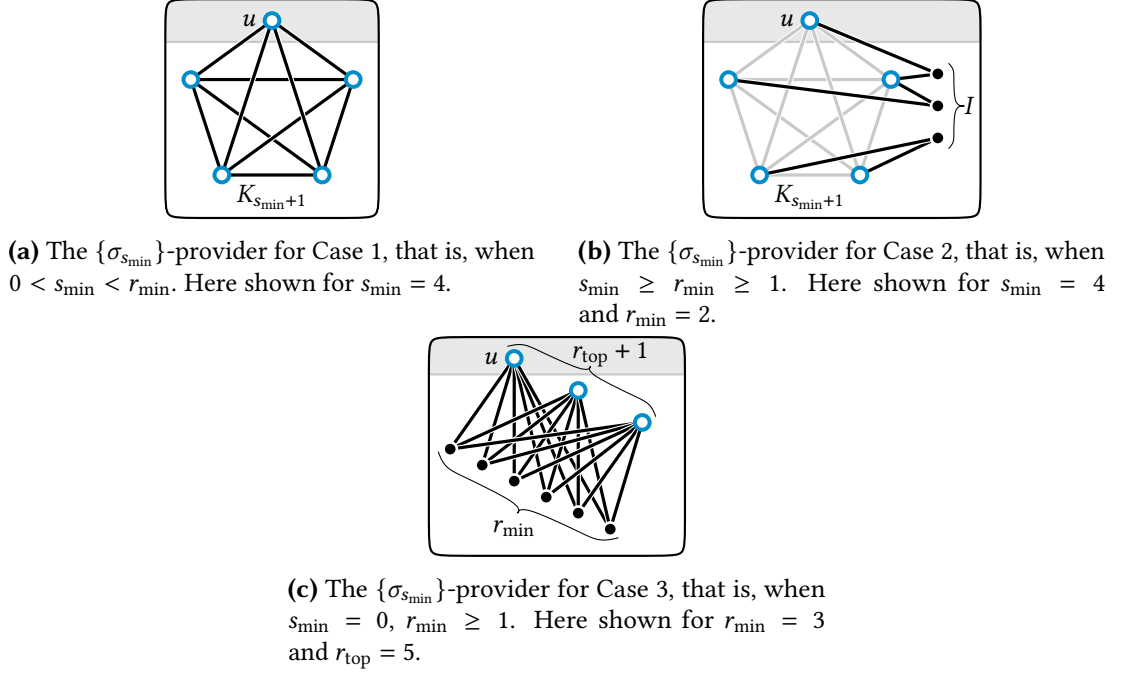


Figure 10.11: The construction of the gadgets from Claim 10.7.1.1 for the three different cases.

Case 1: $0 < s_{\min} < r_{\min}$. We choose H to be a clique of with $s_{\min} + 1$ vertices and declare any of the vertices to be the portal u . Consult Figure 10.11a for a visualization of this construction.

Now, suppose that one vertex $v \neq u$ of H is unselected; this vertex exists as $s_{\min} + 1 \geq 2$. Then, vertex v needs at least $r_{\min}$ selected neighbors. However, vertex v can have at most $s_{\min} < r_{\min}$ selected neighbors, and thus, vertex v cannot be unselected. Since vertex v is selected, it needs at least $s_{\min}$ selected neighbors, which implies that the remaining $s_{\min}$ vertices of the clique (including u) must be selected as well.

Case 2: $s_{\min} \geq r_{\min} \geq 1$. We construct H as follows. We start with a clique C of $s_{\min} + 1$ vertices and an independent set I of $\lceil (s_{\min} + 1)/r_{\min} \rceil$ vertices. We add $r_{\min} \cdot \lceil (s_{\min} + 1)/r_{\min} \rceil$ edges between C and I such that each vertex in I has a degree of exactly $r_{\min}$, and every vertex in C is adjacent to at least one vertex in I . By the choice of the number of edges, it directly follows that this is always possible. Finally, we pick any vertex from C to be the portal vertex u . Consult Figure 10.11b for a visualization of this construction.

Consider an arbitrary vertex v of the clique C . Assume that v is unselected. Let v' be a neighbor of v in the independent set I . If v' is selected, then v' needs at least $s_{\min} \geq r_{\min}$ selected neighbors in C . If v' is unselected, then it needs at least $r_{\min}$ selected neighbors in C . By our assumption that v is unselected and the construction of H , vertex v' can have at most $r_{\min} - 1$ selected neighbors in C , which cannot be turned into a valid solution. Hence, vertex v must be selected. As this argument applies to each vertex v of the clique C , the whole clique C has to be selected, which gives $r_{\min}$ selected neighbors to every vertex of I .

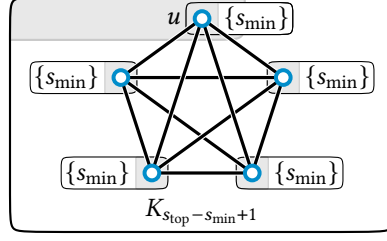


Figure 10.12: The main construction of the $\{\sigma_{s_{\text{top}}}\}$ -provider from Lemma 10.7.1. Here shown for $s_{\text{top}} - s_{\text{min}} = 4$, for example, when $s_{\text{min}} = 3$ and $s_{\text{top}} = 7$. Observe that u has exactly $s_{\text{min}} + (s_{\text{top}} - s_{\text{min}}) = s_{\text{top}}$ selected neighbors. We depict only the portal vertex of the $\{\sigma_{s_{\text{min}}}\}$ -providers from Claim 10.7.1.1.

Case 3: $s_{\text{min}} = 0$ and $r_{\text{min}} \geq 1$. We define H as a complete bipartite graph with bipartition (A, B) such that $|A| = r_{\text{min}}$ and $|B| = r_{\text{top}} + 1$. We pick an arbitrary vertex u in A as the portal vertex u . Consult Figure 10.11c for a visualization of this construction.

Suppose that u is not selected. Then, every vertex in B can have at most $r_{\text{min}} - 1$ selected neighbors, which means that every vertex in B must be selected. In this case, u would have $r_{\text{top}} + 1$ selected neighbors, which is not feasible. Therefore, u must be selected. If u is indeed selected, then selecting every vertex in A , and leaving every vertex in B unselected yields a solution for which u has $s_{\text{min}} = 0$ selected neighbors. $\triangleleft$

Finally, we use Claim 10.7.1.1 to construct a $\{\sigma_{s_{\text{top}}}\}$ -realizer $(G, \{u\})$ as follows. As our graph G , we use $s_{\text{top}} - s_{\text{min}} + 1 \geq 1$ independent copies $(H_i, \{u_i\})$ of the provider from Claim 10.7.1.1, and connect the vertices u_i to form an $(s_{\text{top}} - s_{\text{min}} + 1)$ -clique. We choose any of the vertices u_i to be the portal vertex u .

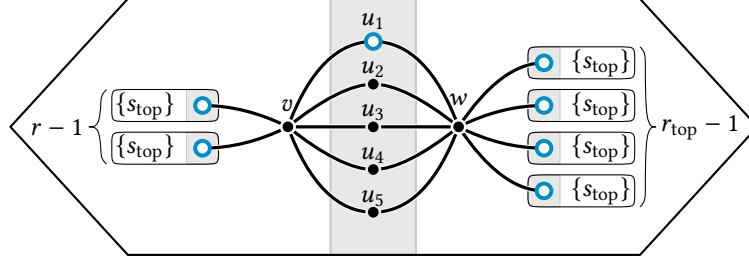
By Claim 10.7.1.1, all vertices u_i are selected in any partial solution for $(G, \{u\})$ with the additional guarantee that each vertex u_i already has at least s_{min} selected neighbors inside the corresponding helper gadgets. Hence, each vertex u_i (and in particular u) has at least s_{top} selected neighbors. More precisely, as σ is finite, each such vertex u_i has exactly s_{top} selected neighbors and, in particular, exactly s_{min} selected neighbors in H_i , which is a valid solution by Claim 10.7.1.1. This completes the proof. $\square$

As a next step, we realize $\text{HW}_{=1}$ -relations. We refer to Figure 10.13 for an illustration of the constructions.

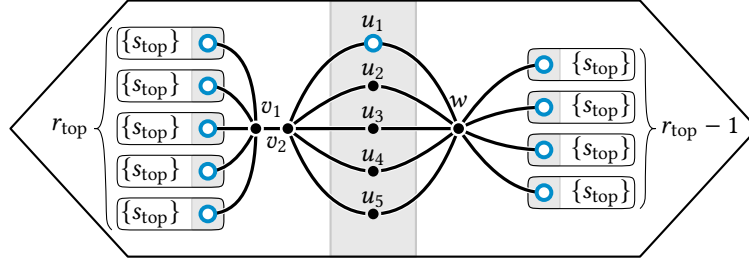
Lemma 10.7.2. Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite with $0 \notin \rho$. Then, for all $k \geq 1$, $\text{HW}_{=1}^{(k)}$ is realizable.

Proof. We distinguish three different cases for the definition of the realization (G, U) with portals $U = \{u_1, \dots, u_k\}$.

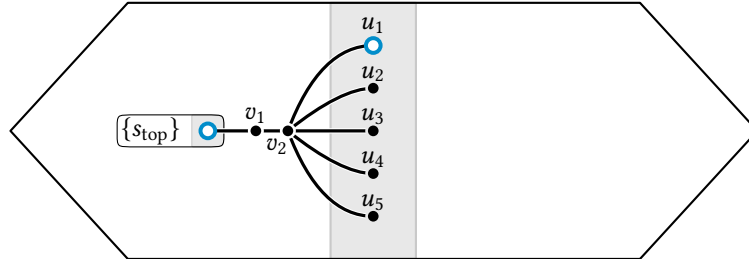
Case 1: There is an $r \in \rho$ with $r \geq 2$ and $r - 1 \notin \rho$. We create two additional vertices v and w , and make both adjacent to the portal u_i for all $i \in [1..k]$. Further, for all $j \in [1..r - 1]$,



(a) The construction in Case 1, that is, when there is an $r \in \rho$ with $r \geq 2$ and $r - 1 \notin \rho$.



(b) The construction in Case 2, that is, when $\rho = [1 \dots r]$ for some $r \geq 2$.



(c) The construction in Case 3, that is, when $\rho = \{1\}$.

Figure 10.13: The realization of $\text{HW}_{=1}^{(k)}$ from Lemma 10.7.2. We depict only the portal vertices of the providers and provide an exemplary solution for $k = 5$ in each case.

we create a copy $(H_{j,1}, u_{j,1})$ of the $\{\sigma_{s_{\text{top}}}\}$ -realizer from Lemma 10.7.1, and connect v to vertex $u_{j,1}$. Next, for all $j \in [1 \dots r_{\text{top}} - 1]$, we create a copy $(H_{j,2}, u_{j,2})$ of the $\{\sigma_{s_{\text{top}}}\}$ -realizer from Lemma 10.7.1 and connect w to vertex $u_{j,2}$. Consult Figure 10.13a for a visualization of this construction.

To see that (G, U) realizes $\text{HW}_{=1}^{(k)}$, first observe that by Lemma 10.7.1, any vertex $u_{\star,\star}$ is always selected and receives s_{top} selected neighbors inside the corresponding realizer. Hence, no neighbor of any vertex $u_{\star,\star}$ outside the corresponding realizer can be selected. In particular, the vertices v and w have to be unselected.

Since $r - 1 \notin \rho$, at least one portal vertex $u_1, \dots, u_k$ must be selected into the solution to satisfy the ρ -constraint of v . Further, as ρ is finite and w already has $r_{\text{top}} - 1$ selected neighbors, at most one of the k portal vertices $u_1, \dots, u_k$ can be selected. Thus, we must select exactly one portal vertex into the solution, completing the proof for this case.

Case 2: $\rho = [1 \dots r]$ for $r \geq 2$. We create three additional vertices v_1, v_2 , and w , and make v_2 and w adjacent to the portal u_i for all $i \in [1 \dots k]$. We also connect v_1 and v_2 . Further, for all $j \in [1 \dots r_{\text{top}}]$, we create a copy $(H_{j,1}, u_{j,1})$ of the $\{\sigma_{s_{\text{top}}}\}$ -realizer from Lemma 10.7.1, and connect v_1 to vertex $u_{j,1}$. Next, for all $j \in [1 \dots r_{\text{top}} - 1]$, we create a copy $(H_{j,2}, u_{j,2})$ of the $\{\sigma_{s_{\text{top}}}\}$ -realizer from Lemma 10.7.1, and connect w to vertex $u_{j,2}$. Consult Figure 10.13b for a visualization of this construction.

As in Case 1, the two vertices v_1 and w must be unselected. Now, observe that v_2 must also be unselected as v_1 already has r_{top} selected neighbors and $r_{\text{top}} + 1 \notin \rho$. As $0 \notin \rho$, at least one of the neighbors of v_2 (but not v_1) must be selected, that is, at least one of $u_1, \dots, u_k$ must be selected. Finally, observe that as in Case 1, the vertex w ensures that at most one of the vertices $u_1, \dots, u_k$ is selected. This completes the proof in this case.

Case 3: $\rho = \{1\}$. We use the same construction as in Case 2, but we remove the vertex w . Consult Figure 10.13c for a visualization of this simplified construction.

The constraint that exactly one portal is selected follows from the properties of v_2 , as it needs exactly one selected neighbor. This completes the proof of Lemma 10.7.2. $\square$

We lift the realization of $\text{HW}_{=1}$ from Lemma 10.7.2 to the following reduction where we replace each relation by its realization.

Main Lemma 10.7.3. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite with $0 \notin \rho$. For all constants d , there is a pathwidth-preserving reduction from $(\sigma, \rho)\text{-DOMSET}^{\text{HW}_{=1}}$ on instances of arity at most d to $(\sigma, \rho)\text{-DOMSET}$.*

Proof. Let G be an instance of $(\sigma, \rho)\text{-DOMSET}^{\text{HW}_{=1}}$ where all relations have arity at most d .

We use Lemma 10.7.2 to replace each $\text{HW}_{=1}$ -relation by its realization. It is easy to check that these realizations do not use relations, that is, they are graphs with portals. Let H be the resulting graph.

A given path decomposition of G is modified as follows. For each $\text{HW}_{=1}$ -relation R of G , there is a bag β in the path decomposition of G that contains R and the scope of R . We duplicate the bag β and denote by β_R the copy. We add all the vertices of the realization of R to β_R .

As the arity of each relation is bounded by a constant, and we add only vertices of one realization to a bag, we have $\text{pw}(H) \leq \text{pw}(G) + \mathcal{O}(1)$. Hence, the reduction is pathwidth-preserving. $\square$

Next, we prove the corresponding result for the counting version, which is significantly more complicated.

10.8 Counting Version: Realizing Relations

For the counting version, the realization of $\text{HW}_{=1}$ is much more complicated than for the decision version. Indeed, the proof of Main Lemma 10.8.1 is very technical and splits into a number of cases and sequences of reductions. Intuitively, in the counting version, it is not always plausible that relations can be realized directly by attaching some graph gadget. Instead, we heavily rely on interpolation techniques to isolate the number of selections we wish to determine.

For this reason, we also exclude the case when $\rho = \mathbb{N}$ and σ is cofinite, and refer to Section 10.9 for this special case. Formally, in the following, we show that if the counting problem $(\sigma, \rho)\text{-\#DOMSET}$ is not polynomial-time solvable (and excluding this special case), then it is sufficiently expressive to realize $\text{HW}_{=1}$, that is, we prove Main Lemma 10.8.1.

Main Lemma 10.8.1. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite or cofinite such that (σ, ρ) is non-trivial. Moreover, assume that σ is finite or $\rho \neq \mathbb{N}$.*

For all constants d , there is a pathwidth-preserving reduction from $(\sigma, \rho)\text{-\#DOMSET}^{\text{HW}_{=1}}$ on instances with arity at most d to $(\sigma, \rho)\text{-\#DOMSET}$.

To formally state the main steps to prove Main Lemma 10.8.1, we first introduce some notation which we use throughout this section.

Notation. Let Ω_0 be the set of all finite and cofinite subsets of $\mathbb{N}$, and let $\Omega := \Omega_0 \setminus \{\emptyset\}$.

Fix a pair of sets $(\sigma, \rho) \in \Omega^2$. For some non-negative integer k , let $\mathcal{P} \subseteq \Omega_0^2$ be a set of pairs $\{(\sigma^{(i)}, \rho^{(i)}) \in \Omega_0^2 \mid i \in [1..k]\}$. We now define a generalization of $(\sigma, \rho)\text{-\#DOMSET}$ that allows us to augment the problem with additional constraints specified by $\mathcal{P}$ in order to obtain more modeling power. For ease of notation, we define $(\sigma^{(0)}, \rho^{(0)}) := (\sigma, \rho)$.

The problem $(\sigma, \rho)\text{-\#DOMSET}^{\mathcal{P}}$ takes as input some graph G together with a mapping $\lambda: V(G) \rightarrow [0..k]$. Then, a λ -set of G is a subset $S \subseteq V(G)$ such that

- for each $v \in S$, we have $|N(v) \cap S| \in \sigma^{(\lambda(v))}$, and
- for each $v \notin S$, we have $|N(v) \cap S| \in \rho^{(\lambda(v))}$.

The problem $(\sigma, \rho)\text{-\#DOMSET}^{\mathcal{P}}$ then asks for the number of λ -sets of G .

If $\mathcal{P}$ is the empty set, then we also drop the superscript and simply write $(\sigma, \rho)\text{-\#DOMSET}$. In this case, it also suffices to take as input only the graph G and assume that λ is constant 0. A λ -set in this case is simply a (σ, ρ) -set. We also refer to the set S as a *solution* of $(\sigma, \rho)\text{-\#DOMSET}^{\mathcal{P}}$ (on input G, λ). For a less formal, but more convenient specification of λ , we say that a vertex v with $\lambda(v) = i$ is a $(\sigma^{(i)}, \rho^{(i)})$ -vertex.

Sometimes we restrict the problem $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$ to instances for which certain pairs from $\mathcal{P}$ are used only a constant number of times (w.r.t. the number of vertices in G). To this end, for some positive integer c , we say that a pair $(\sigma^{(i)}, \rho^{(i)}) \in \mathcal{P}$ is c -bounded if we restrict $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$ to instances G, λ with $|\lambda^{-1}(i)| \leq c$.

In order to define pairs in $\mathcal{P}$, the following notation is useful. For a set τ of non-negative integers (think of σ or ρ) and an integer s , we define $\tau - s := \{k - s \mid k \in \tau, k - s \geq 0\}$.

When working with $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$, it is helpful to generalize some definitions from Chapter 2. For a graph G with portals U and a mapping $\lambda: V(G) \rightarrow [0..k]$, the tuple $\mathcal{G} = (G, U, \lambda)$ is a *gadget* for $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$. Again, we drop λ if $\mathcal{P} = \emptyset$. Then, a *partial solution* of (G, U, λ) is a set $S \subseteq V(G)$ such that

- for each $v \in S \setminus U$, we have $|N(v) \cap S| \in \sigma^{(\lambda(v))}$, and
- for each $v \notin S \cup U$, we have $|N(v) \cap S| \in \rho^{(\lambda(v))}$.

Let $n = |V(G)|$ be the number of vertices in G . Recall that $\mathbb{A}_n := \{\sigma_0, \dots, \sigma_n, \rho_0, \dots, \rho_n\}$. Recall that a partial solution S *witnesses* the vector $x \in \mathbb{A}_n^U$ if, for each $u \in U$ and $z := |N(u) \cap S|$, we have

$$x[u] = \begin{cases} \sigma_z & \text{if } u \in S, \\ \rho_z & \text{otherwise.} \end{cases}$$

Then, for $x \in \mathbb{A}_n^U$, we denote by $\text{ext}_{\mathcal{G}}(x)$ the number of partial solutions of $\mathcal{G}$ that witness x . We also refer to this as the number of *extensions* of x (to the gadget $\mathcal{G}$).

Given a set of pairs $\mathcal{P}$ from Ω_0^2 and a pair $(\sigma', \rho') \in \Omega_0^2$, we use the shorthand $\mathcal{P} + (\sigma', \rho')$ for the set $\mathcal{P} \cup \{(\sigma', \rho')\}$.

Main Steps. As mentioned, in its entirety the proof of Main Lemma 10.8.1 is rather long and involved. This is why we try to break it down as much as possible. The main outline consists of two steps where we consider two cases for the second step. In the following, we outline the proof relying on these three intermediate result, and prove them afterward.

Consult Figure 10.14 for a more detailed overview of the reductions involved.

As a first common step, we prove in Lemma 10.8.5 that we can realize relations if we have access to unselected vertices where the set ρ is shifted. We defer the proof to Section 10.8.1.

Lemma 10.8.5. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$. Then, $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{HW}=1} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#\text{DOMSET}^{(\emptyset, \rho - r_{\text{top}} + 1)}$.*

For the second step, we distinguish two cases depending on ρ ; the first case considers $\rho \neq \mathbb{N}$ and the second case considers $\rho = \mathbb{N}$ which implies that σ is finite.

We cover the first case by Lemma 10.8.10 which we prove in Section 10.8.2.

Lemma 10.8.10. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . If $\rho \neq \mathbb{N}$, then $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P} + (\emptyset, \rho - r_{\text{top}} + 1)} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$.*

We cover the second case when $\rho = \mathbb{N}$ but σ is finite by Lemma 10.8.17, which we prove in Section 10.8.3.

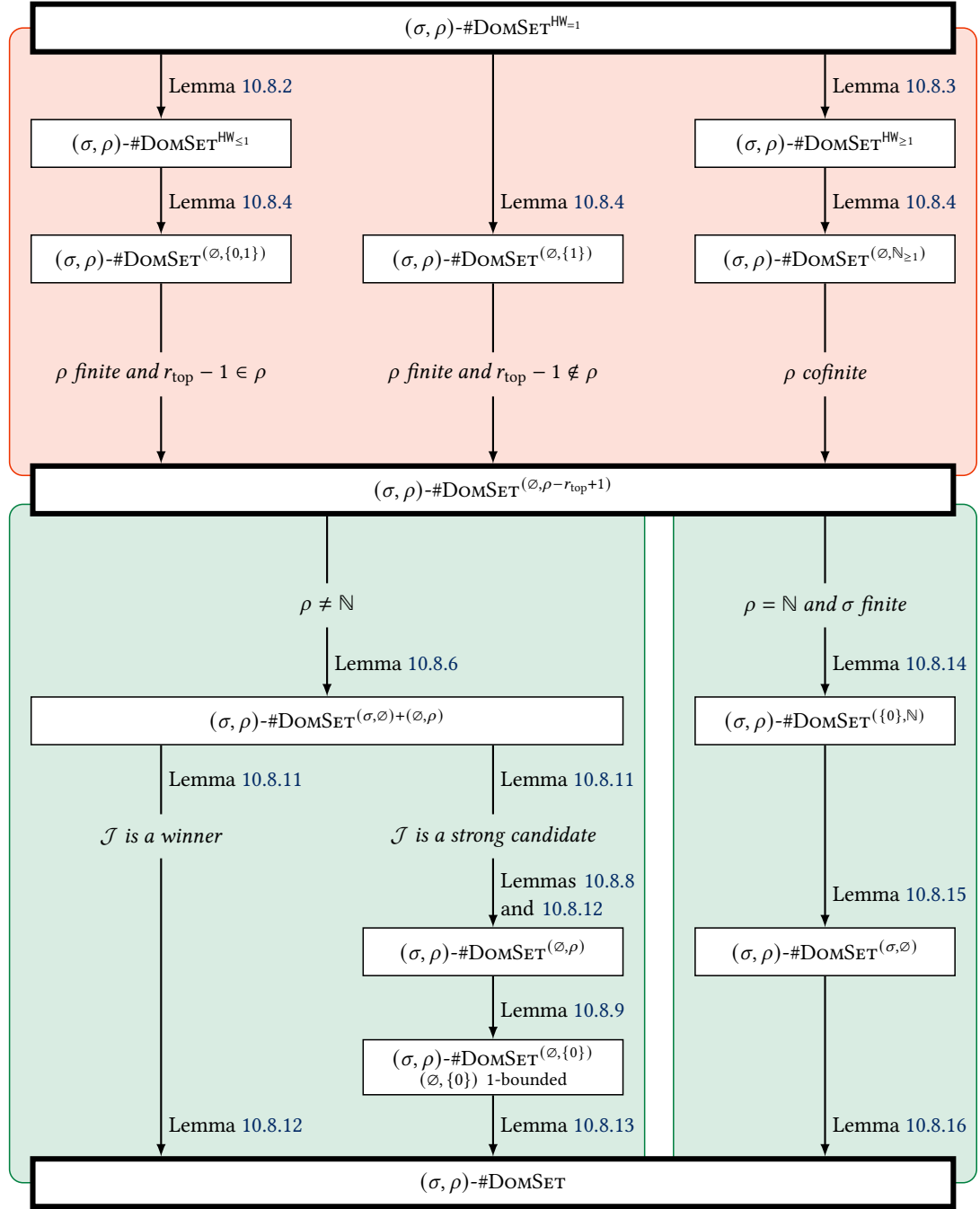


Figure 10.14: An overview of the reductions in Section 10.8 to realize $\text{HW}=1$ for $(\sigma, \rho)\text{-}\#\text{DOMSET}$. The sets σ and ρ are finite or cofinite sets where $\rho \neq \mathbb{N}$ or σ is finite. The reductions in the box on the top are given in Section 10.8.1 and prove Lemma 10.8.5. The reductions in the box on the left are given in Section 10.8.2 and prove Lemma 10.8.10, and the reductions in box on the right are given in Section 10.8.3 and prove Lemma 10.8.17.

Lemma 10.8.17. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . If $\rho = \mathbb{N}$ and σ is finite, then $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}+(\emptyset, \mathbb{N}_{\geq 1})} \leq_{pw} (\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}}$.*

With these three results, we have everything ready to prove the main result of this section in the straightforward way.

Main Lemma 10.8.1. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are finite or cofinite such that (σ, ρ) is non-trivial. Moreover, assume that σ is finite or $\rho \neq \mathbb{N}$.*

For all constants d , there is a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#DOMSET^{HW=1}$ on instances with arity at most d to $(\sigma, \rho)\text{-}\#DOMSET$.

Proof. We distinguish two cases.

- If $\rho \neq \mathbb{N}$, then the proof follows from Lemmas 10.8.5 and 10.8.10 with $\mathcal{P} = \emptyset$.
- If $\rho = \mathbb{N}$, then we know by assumption that σ is finite. Hence, the proof follows from Lemmas 10.8.5 and 10.8.17 with $\mathcal{P} = \emptyset$. $\square$

Before we dive into the proofs of Lemmas 10.8.5, 10.8.10 and 10.8.17, we first give a rough outline to point out how the proofs relate, or how they differ.

The key idea of Lemma 10.8.5 is to show that each of the following pairs can be used to model $HW=1$ constraints:

- $(\emptyset, \{1\})$,
- $(\emptyset, \{0, 1\})$,
- $(\emptyset, \mathbb{N}_{\geq 1})$,

The first of these pairs, for example, refers to vertices that are never selected since the first entry is $\emptyset$, and if they are unselected they have exactly one selected neighbor. Depending on the set ρ , we show that one such vertex corresponds to $(\emptyset, \rho - r_{\text{top}} + 1)$ -vertices.

The key idea of Lemma 10.8.10 is to force the selection status of (σ, ρ) -vertices. More precisely, if $\rho \neq \mathbb{N}$, we show that it suffices to force the selection status of (σ, ρ) -vertices, that is, if we can model $(\emptyset, \rho)$ -vertices as well as $(\sigma, \emptyset)$ -vertices, then we can model $(\emptyset, \rho - r_{\text{top}} + 1)$. Then, it remains to show that we can force the selection status of (σ, ρ) -vertices.

The key idea of Lemma 10.8.17 is related but much simpler. For the case when $\rho = \mathbb{N}$ and σ is finite, it suffices to force selected vertices, that is, to model $(\sigma, \emptyset)$ -vertices, in order to simulate $(\emptyset, \rho - r_{\text{top}} + 1)$ -vertices.

Organization. We first prove Lemma 10.8.5 in Section 10.8.1. Then, in Section 10.8.2 we prove Lemma 10.8.10. Finally, in Section 10.8.3 we prove Lemma 10.8.17.

10.8.1 Shifting the Set ρ

To prove the reduction from $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{HW}=1}$ to $(\sigma, \rho)\text{-}\#\text{DOMSET}^{(\emptyset, \rho - r_{\text{top}} + 1)}$, that is, to prove Lemma 10.8.5, we first realize $\text{HW}_{=1}$ -relations using $\text{HW}_{\geq 1}$ or $\text{HW}_{\leq 1}$. We define these relations as $\text{HW}_{\leq 1} := \text{HW}_{\in \{0,1\}}$ and $\text{HW}_{\geq 1} := \text{HW}_{\in \mathbb{N}_{\geq 1}}$ (see also Definition 2.1.6 for the formal definition of $\text{HW}_{\in S}$).

In the following we first show in Lemma 10.8.2 that we can model $\text{HW}_{=1}$ -relations using $\text{HW}_{\leq 1}$ -relations. Afterward in Lemma 10.8.3, we show that we can model $\text{HW}_{=1}$ -relations using $\text{HW}_{\geq 1}$ -relations.

Then, as a next step, which is formalized in Lemma 10.8.4, we use certain additional $(\emptyset, \rho')$ -pairs to replace the relations. We replace

- $\text{HW}_{=1}$ by $(\emptyset, \{1\})$ -vertices,
- $\text{HW}_{\leq 1}$ by $(\emptyset, \{0, 1\})$ -vertices, and
- $\text{HW}_{\geq 1}$ by $(\emptyset, \mathbb{N}_{\geq 1})$ -vertices.

Finally, we show that depending on the set ρ one of the cases corresponds to $(\emptyset, \rho - r_{\text{top}} + 1)$ -vertices which then concludes the proof.

Modifying the Relations. As the first step, we establish that $\text{HW}_{=1}$ -relations can also be modeled using $\text{HW}_{\leq 1}$ -relations. We use interpolation techniques and the auxiliary result from Lemma 10.2.1 to obtain the reduction. We restate Lemma 10.2.1 for convenience.

Lemma 10.2.1. *Fix two non-empty sets $\sigma, \rho \subseteq \mathbb{N}$. For all $s \in \sigma$ and $r \in \rho$ with $r \geq 1$, there is a $\{\sigma_s, \rho_r\}$ -provider $(G, \{u\})$ that has disjoint (σ, ρ) -sets X and Y with $X \cup Y = V(G)$, where $u \in X$ with $|N(u) \cap X| = s$ and $|N(u) \cap Y| = r$.*

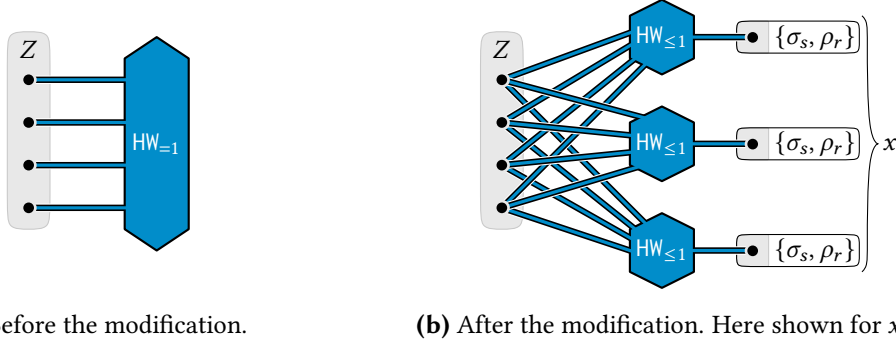
Note that the requirements of Lemma 10.2.1 are always fulfilled if (σ, ρ) is a non-trivial pair in Ω^2 . In this case, both σ and ρ are non-empty, and $\rho \neq \{0\}$.

Lemma 10.8.2. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$. Then, $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{HW}=1} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{HW}\leq 1}$.*

Proof. Let I be an instance of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{HW}=1}$. Let $\mathcal{C}$ be the set of relational constraints of I (all of which enforce $\text{HW}_{=1}$ -relations), and let $\mathcal{Z} = \{\text{scp}(C) \mid C \in \mathcal{C}\}$. We remove the relational constraints $\mathcal{C}$ from the instance I to form a new instance I' . I' can be cast as an instance of $(\sigma, \rho)\text{-}\#\text{DOMSET}$. Then, the number of solutions of I is identical to the number of those solutions of I' that select precisely one vertex from each $Z \in \mathcal{Z}$.

By Lemma 10.2.1, we can pick $s \in \sigma$ and $r \in \rho$ such that there is a $\{\sigma_s, \rho_r\}$ -provider J with portal p such that J has $\alpha \geq 1$ (σ, ρ) -sets that contain p , and it has $\beta \geq 1$ (σ, ρ) -sets that do not contain p . It is not clear whether $\alpha = \beta$ or not.

For a positive integer x , let I'_x be the instance of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{HW}\leq 1}$ obtained from I' by adding, for each set $Z \in \mathcal{Z}$, x copies $J_1^Z, \dots, J_x^Z$ of the graph J . Let $p_1^Z, \dots, p_x^Z$ be the corresponding copies of p . For each $i \in [1 \dots x]$, we add a relational constraint with scope



(a) Before the modification.

(b) After the modification. Here shown for $x = 3$.

Figure 10.15: The construction of instance I'_x from Lemma 10.8.2 to replace a $\text{HW}_{=1}$ relation between the vertices of a single set Z by $\text{HW}_{\leq 1}$ -relations.

$Z \cup \{p_i^Z\}$ that enforces a $\text{HW}_{\leq 1}$ -relation on this scope. Consult Figure 10.15 for a visualization of this construction (for a single set Z).

Note that a solution S' of I' for (σ, ρ) -#DOMSET can be extended only to a solution of I'_x for (σ, ρ) -#DOMSET $^{\text{HW}_{\leq 1}}$ if, in each set $Z \in \mathcal{Z}$, at most one vertex is selected (because of the attached $\text{HW}_{\leq 1}$ -relations). Let us say that solutions of I' with this property are *good*. Suppose in a good solution S' , a set $Z \in \mathcal{Z}$ is entirely unselected. Then, there are $f_0 := (\alpha + \beta)^x$ feasible extensions to its attached copies of J ($\alpha + \beta$ for each graph J_i^Z since p_i^Z can either be selected or not). If exactly 1 vertex of Z is selected in S' , then there are $f_1 := \beta^x$ extensions to the attached copies of J (as we can only use those selections of the J_i^Z 's for which p_i^Z is unselected).

Let a_i be the number of good solutions of I' in which precisely i of the sets in $\mathcal{Z}$ are entirely unselected. In the remaining $|\mathcal{Z}| - i$ sets from $\mathcal{Z}$, exactly one vertex is selected. Let $\#\text{Sol}(I'_x)$ denote the number of solutions of I'_x . Then,

$$\begin{aligned}
 \#\text{Sol}(I'_x) &= \sum_{i=0}^{|\mathcal{Z}|} a_i \cdot f_0^i \cdot f_1^{|\mathcal{Z}|-i} \\
 &= f_1^{|\mathcal{Z}|} \cdot \sum_{i=0}^{|\mathcal{Z}|} a_i \cdot \left(\frac{f_0}{f_1}\right)^i \\
 &= \beta^{x \cdot |\mathcal{Z}|} \cdot \sum_{i=0}^{|\mathcal{Z}|} a_i \cdot \left(\left(\frac{\alpha + \beta}{\beta}\right)^x\right)^i.
 \end{aligned}$$

Recall that $\beta \geq 1$, and therefore, $\#\text{Sol}(I'_x)/\beta^{x \cdot |\mathcal{Z}|}$ is a polynomial in $((\alpha + \beta)/\beta)^x$ of degree $|\mathcal{Z}|$. Since $(\alpha + \beta)/\beta > 1$, it suffices to choose $|\mathcal{Z}|$ different values of $x > 0$ for the interpolation. This way we can recover the coefficients a_i (see Fact 2.1.7, for example). In particular, we can recover a_0 which corresponds to the number of good solutions of I' in which none of the sets in $\mathcal{Z}$ are entirely unselected, that is, the solution in which all of these sets contain precisely one selected vertex. This is exactly the number of solutions of I which we want to compute.

Since the scope of a relational constraint in I is considered as a clique in the definition of the pathwidth of I , for $Z \in \mathcal{Z}$, we can add all vertices from $J_1^Z, \dots, J_x^Z$ one after another to a

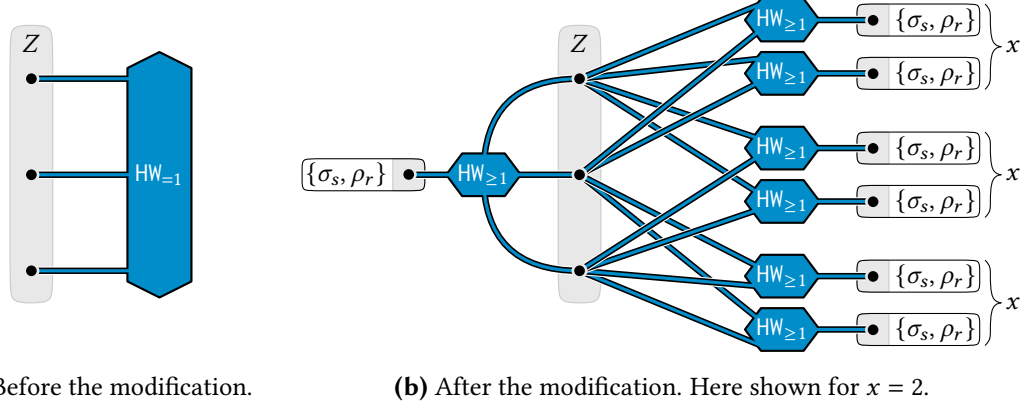


Figure 10.16: The construction of instance I'_x from Lemma 10.8.3 to replace a $\text{HW}_{=1}$ relation between the vertices of a single set Z by $\text{HW}_{\geq 1}$ -relations.

copy of the original bag containing the clique Z . Hence, the pathwidth of I'_x is at most that of I plus an additive constant (depending only on σ and ρ). $\square$

As a next result, we show that we can replace $\text{HW}_{=1}$ by $\text{HW}_{\geq 1}$.

Lemma 10.8.3. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$. Then, $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{HW}_{=1}} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{HW}_{\geq 1}}$.*

Proof. The proof is similar to that of Lemma 10.8.2, but uses a different gadget, which leads to different numbers of extensions that have to be considered. Let I be an instance of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{HW}_{=1}}$. We define $\mathcal{Z}$, I' , J , α , and β precisely as we did in the proof of Lemma 10.8.2.

When we define I'_x , we divert from the proof of Lemma 10.8.2. For a positive integer x , let I'_x be the instance of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{HW}_{\geq 1}}$ that we obtain from I' where we make each $Z \in \mathcal{Z}$ the scope of a $\text{HW}_{\geq 1}$ -relation. In addition, for each subset Z' of Z with $|Z'| = |Z| - 1$, we introduce a total of x copies $J_1^{(Z, Z')}, \dots, J_x^{(Z, Z')}$ of the graph J . For each corresponding copy $p_i^{(Z, Z')}$ of p we introduce a $\text{HW}_{\geq 1}$ -relation with scope $Z' \cup \{p_i^{(Z, Z')}\}$. Note that some set Z' may receive multiple such attachments for different supersets Z . We refer to Figure 10.16 for an illustration of the construction.

Note that a solution S' of I' for $(\sigma, \rho)\text{-}\#\text{DOMSET}$ can be extended only to a solution of I'_x for $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{HW}_{\geq 1}}$ if at least one vertex is selected in each set $Z \in \mathcal{Z}$ because of the $\text{HW}_{\geq 1}$ -relation attached to Z . Let us say that solutions of I' with this property are *good*. If, in a good solution S' of I' , some set Z' contains at least one selected vertex, then there are $f_1 := (\alpha + \beta)^x$ feasible extensions to the graphs $J_1^{(Z, Z')}, \dots, J_x^{(Z, Z')}$; $\alpha + \beta$ for each graph $J_i^{(Z, Z')}$ since the corresponding copy of p can either be selected or not. If no vertex of Z' is selected in S' , then there are only $f_0 := \alpha^x$ extensions; each copy of p in some gadget $J_i^{(Z, Z')}$ has to be selected to fulfill the corresponding $\text{HW}_{\geq 1}$ constraints. Continuing from this, we say that set $Z \in \mathcal{Z}$ is *undesired* if it contains at least two selected vertices from S' , and the set is *desired* if it contains precisely one selected vertex from S' .

For an undesired set $Z \in \mathcal{Z}$, there is at least one vertex selected in each of the $|Z|$ corresponding subsets Z' . This gives a total of $f_1^{|Z|}$ extensions to the graphs $J_{\star}^{(Z, \star)}$. In contrast, for a desired set Z , there is exactly one subset Z' that is entirely unselected. This gives a total of $f_1^{(|Z|-1)} \cdot f_0$ extensions.

For a set $\mathcal{Z}' \subseteq \mathcal{Z}$, denote by $a_{\mathcal{Z}'}$ the number of good solutions of I' for which the sets in $\mathcal{Z}'$ are undesired and the sets in $\mathcal{Z} \setminus \mathcal{Z}'$ are desired. Further, we denote by a_i the number of good solutions of I' in which precisely i of the sets in $\mathcal{Z}$ are undesired. Let $\#Sol(I'_x)$ denote the number of solutions of I'_x . Then,

$$\begin{aligned} \#Sol(I'_x) &= \sum_{\mathcal{Z}' \subseteq \mathcal{Z}} a_{\mathcal{Z}'} \cdot \left(\prod_{Z \in \mathcal{Z}'} f_1^{|Z|} \cdot \prod_{Z \notin \mathcal{Z}'} \left(f_1^{(|Z|-1)} \cdot f_0 \right) \right) \\ &= f_1^{\sum_{Z \in \mathcal{Z}} (|Z|-1)} \cdot \sum_{\mathcal{Z}' \subseteq \mathcal{Z}} a_{\mathcal{Z}'} \cdot f_1^{|\mathcal{Z}'|} \cdot f_0^{|\mathcal{Z} \setminus \mathcal{Z}'|} \\ &= \underbrace{f_1^{\sum_{Z \in \mathcal{Z}} (|Z|-1)} \cdot f_0^{|\mathcal{Z}|}}_{F=} \cdot \sum_{i=0}^{|\mathcal{Z}|} a_i \cdot \left(\frac{(\alpha + \beta)^x}{\alpha^x} \right)^i. \end{aligned}$$

Note that $\#Sol(I'_x)/F$ is a polynomial in $(\alpha + \beta)^x/\alpha^x$ of degree $|\mathcal{Z}|$. Since $\alpha, \beta \geq 1$, the values of $((\alpha + \beta)/\alpha)^x$ are defined and distinct for different $x > 0$. It suffices to choose $|\mathcal{Z}|$ different values of $x > 0$ for the interpolation. This way we can recover the coefficients a_i (see Fact 2.1.7, for example). In particular, we can recover a_0 , which corresponds to the number of good solutions of I' in which none of the sets in $\mathcal{Z}$ are undesired, which means that all sets in $\mathcal{Z}$ are desired, that is, in all sets in $\mathcal{Z}$ precisely one vertex is selected. This is exactly the number of solutions of I , which we want to compute.

As in the proof of Lemma 10.8.2, we can argue that the pathwidth of I'_x is at most that of I plus an additive term in $\mathcal{O}(1)$. $\square$

Removing the Relations. As a next step, we replace the relations by some appropriate vertices. We simply replace $\text{HW}_{=1}$ relation by an $(\emptyset, \{1\})$ -vertex, similarly for $\text{HW}_{\leq 1}$ and $(\emptyset, \{0, 1\})$ -vertices, and $\text{HW}_{\geq 1}$ and $(\emptyset, \mathbb{N}_{\geq 1})$ -vertices.

Lemma 10.8.4. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$. Then, we have each of the following three reductions:*

1. $(\sigma, \rho)\text{-}\#DOMSET^{\text{HW}_{=1}} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#DOMSET^{(\emptyset, \{1\})}$.
2. $(\sigma, \rho)\text{-}\#DOMSET^{\text{HW}_{\leq 1}} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#DOMSET^{(\emptyset, \{0, 1\})}$.
3. $(\sigma, \rho)\text{-}\#DOMSET^{\text{HW}_{\geq 1}} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#DOMSET^{(\emptyset, \mathbb{N}_{\geq 1})}$.

Proof. We focus on the first case only, as the other cases follow analogously. Let I be an instance of $(\sigma, \rho)\text{-}\#DOMSET^{\text{HW}_{=1}}$. We create an instance I' of $(\sigma, \rho)\text{-}\#DOMSET^{(\emptyset, \{1\})}$ as follows. For each relation $\text{HW}_{=1}$ with scope S , we completely connect the vertices in S to a new $(\emptyset, \{1\})$ -vertex.

It is straightforward to see that the solutions of I and I' are in a one-to-one-correspondence. Since each scope of a relation in I is considered as a clique in the definition of the pathwidth of I , the pathwidth of I' is at most that of I plus 1. $\square$

Now, we combine the previous reductions from Lemmas 10.8.2 and 10.8.3 with Lemma 10.8.4 to prove the main result.

Lemma 10.8.5. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$. Then, $(\sigma, \rho)\text{-}\#DOMSET^{\text{HW}=1} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#DOMSET^{(\emptyset, \rho - r_{\text{top}} + 1)}$.*

Proof. First, suppose that ρ is finite. If $r_{\text{top}} - 1 \notin \rho$, then $\rho - (r_{\text{top}} - 1)$ is equal to $\{1\}$, and the reduction follows from Case 1 of Lemma 10.8.4. Otherwise, we have that $r_{\text{top}} - 1 \in \rho$. In this case, $\rho - (r_{\text{top}} - 1)$ is equal to $\{0, 1\}$, and the reduction follows from Lemmas 10.8.2 and 10.8.4.

Second, suppose that ρ is cofinite. Then, r_{top} is the largest integer missing from ρ plus 1, and consequently $r_{\text{top}} - 1 \notin \rho$. Therefore, $\rho - (r_{\text{top}} - 1)$ is equal to $\mathbb{N}_{\geq 1}$, and the reduction follows from Lemmas 10.8.3 and 10.8.4.

Thus, we obtain $(\sigma, \rho)\text{-}\#DOMSET^{\text{HW}=1} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#DOMSET^{(\emptyset, \rho - (r_{\text{top}} - 1))}$ which concludes the proof. $\square$

10.8.2 The Case When $\rho \neq \mathbb{N}$

In this section, we reduce from $(\sigma, \rho)\text{-}\#DOMSET^{(\emptyset, \rho - r_{\text{top}} + 1)}$ to $(\sigma, \rho)\text{-}\#DOMSET$ when ρ is not equal to $\mathbb{N}$. Formally, we prove Lemma 10.8.10.

Lemma 10.8.10. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . If $\rho \neq \mathbb{N}$, then $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}+(\emptyset, \rho - r_{\text{top}} + 1)} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}}$.*

Instead of providing this reduction directly, we first reduce to the version with $(\sigma, \emptyset)$ -vertices and $(\emptyset, \rho)$ -vertices. Intuitively, these are regular (σ, ρ) -vertices that we force to be selected and vertices that we force to be unselected, respectively.

Lemma 10.8.6. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . Then, for all $0 \leq i \leq r_{\text{top}}$, $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}+(\emptyset, \rho - i)} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}+(\sigma, \emptyset) + (\emptyset, \rho)}$.*

Proof. We first construct a gadget $(H, \{u\})$ which simulates a $(\sigma', \emptyset)$ -vertex, for some $\sigma' \subseteq \mathbb{N}$ with $0 \in \sigma'$. The gadget H consists of a clique of $s_{\min} + 1$ $(\sigma, \emptyset)$ -vertices. Fix one vertex in this clique as the distinguished portal u . It is immediate that u behaves like a $(\sigma - s_{\min}, \emptyset)$ -vertex which is, by definition of $s_{\min}$, a $(\sigma', \emptyset)$ -vertex with $0 \in \sigma'$.

Now, we modify a given $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}+(\emptyset, \rho - i)}$ instance I as follows. We turn each $(\emptyset, \rho - i)$ -vertex v into a $(\emptyset, \rho)$ -vertex that is adjacent to the portals of i new copies of H . Since the portals of the freshly added gadgets are selected, the vertex v already has i selected neighbors, and thus, behaves like an $(\emptyset, \rho - i)$ -vertex with respect to the vertices of the original instance. Let I' be the resulting instance of $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}+(\sigma, \emptyset) + (\emptyset, \rho)}$.

It remains to argue that this modification preserves the pathwidth. For each $(\emptyset, \rho - i)$ -vertex v , there is a bag in the path decomposition of I containing v . We duplicate this bag i times for the path decomposition of I' , and add one copy of H to precisely one bag. As the size of H is bounded by a function in $s_{\min}$, the size of H is constant and the pathwidth of I' increases only by an additive constant. $\square$

Now, the main work is to show that selected and unselected vertices can always be modeled, that is, we replace the $(\sigma, \emptyset)$ -vertices and the $(\emptyset, \rho)$ -vertices by special gadgets. We show that

We might have $r_{\text{top}} = 0$ for $\rho = \mathbb{N}$.

this is possible if gadgets with certain extension counts exist to which we refer as *winners* and *strong candidates*.

Definition 10.8.7 (Candidates and Winners). *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some possibly empty set of pairs from Ω_0^2 . Let $\mathcal{G}$ be a gadget for $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}}$ with a single portal. We say that $\mathcal{G}$ is a candidate if it has the following properties:*

- (I) $\text{ext}_{\mathcal{G}}(\rho_0), \text{ext}_{\mathcal{G}}(\sigma_0) \geq 1$,
- (II) $\text{ext}_{\mathcal{G}}(\rho_0) \neq \text{ext}_{\mathcal{G}}(\sigma_0)$,
- (III) $\text{ext}_{\mathcal{G}}(\sigma_i) = \text{ext}_{\mathcal{G}}(\rho_i) = 0$ for all $i \geq 2$.

We say that a candidate $\mathcal{G}$ is a winner if additionally it holds that

- (IV) $\text{ext}_{\mathcal{G}}(\rho_1) = \text{ext}_{\mathcal{G}}(\sigma_1) = 0$.

We say that a candidate $\mathcal{G}$ is a strong candidate if (instead of Condition (IV)) it additionally holds that

- (V) $\text{ext}_{\mathcal{G}}(\rho_1) \geq 1$.
- (VI) $\text{ext}_{\mathcal{G}}(\rho_0) \neq \text{ext}_{\mathcal{G}}(\sigma_0) + \text{ext}_{\mathcal{G}}(\sigma_1)$.

As a first result, which we defer to Section 10.8.2.1, we prove that these gadgets exist. Unfortunately, we cannot choose which of the two gadgets we want to construct, but the choice depends on the sets σ and ρ .

Lemma 10.8.11. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$. If $\rho \neq \mathbb{N}$, then there is a gadget $\mathcal{G} = (G, \{p\})$ for $(\sigma, \rho)\text{-}\#DOMSET$ that is either a winner or a strong candidate.*

When Lemma 10.8.11 returns a winner, we can directly use the reduction from Lemma 10.8.12, which we prove in Section 10.8.2.2, and conclude the proof.

This is why the gadget is called a *winner*.

Lemma 10.8.12. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . If we are given a winner $\mathcal{J} = (J, \{v\})$ for $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}}$, then $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}+(\emptyset, \rho)+(\sigma, \emptyset)} \leq_{pw} (\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}}$.*

Otherwise, that is, if Lemma 10.8.11 returns a strong candidate, we first observe that we can simulate a winner when we additionally allow $(\emptyset, \rho)$ -vertices.

Lemma 10.8.8. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . If we are given a strong candidate $\mathcal{J}$ for $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}}$, then there is a winner $\mathcal{J}'$ for $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}+(\emptyset, \rho)}$.*

Proof. We replace all neighbors of the portal vertex in $\mathcal{J}$ by $(\emptyset, \rho)$ -vertices, and denote the resulting gadget by $\mathcal{J}'$.

We first observe that $\mathcal{J}'$ is a gadget for $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}+(\emptyset, \rho)}$ that inherits the properties $\text{ext}_{\mathcal{J}'}(\rho_0), \text{ext}_{\mathcal{J}'}(\sigma_0) \geq 1$, and $\text{ext}_{\mathcal{J}'}(\rho_0) \neq \text{ext}_{\mathcal{J}'}(\sigma_0)$ from $\mathcal{J}$ as the neighbors of the portal are not selected in these solutions.

But now, since all neighbors of the portal have to be unselected, we also have $\text{ext}_{\mathcal{J}'}(\rho_i) = \text{ext}_{\mathcal{J}'}(\sigma_i) = 0$ for all $i \geq 1$. We conclude that $\mathcal{J}'$ is a *winner*. $\square$

With Lemma 10.8.8, we can use Lemma 10.8.12, to remove the selected $(\sigma, \emptyset)$ -vertices. The $(\emptyset, \rho)$ -vertices are technically also removed by this reduction, but, unfortunately, the modified strong candidate introduces new such vertices. Hence, we need an additional result to remove $(\emptyset, \rho)$ -vertices if we are given a strong candidate. More precisely, we use the reduction from Lemma 10.8.9 which allows us to remove these vertices if we have access to a *single* unselected vertex that must not have any neighbors in any solution.

Lemma 10.8.9. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . Then, $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\emptyset, \rho)} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\emptyset, \{0\})}$, even if $(\emptyset, \{0\})$ is 1-bounded in the target problem.*

Proof. The construction is straightforward: we make all vertices that should be unselected (that should be $(\emptyset, \rho)$ -vertices) adjacent to a single $(\emptyset, \{0\})$ -vertex p .

The key observation is that vertex p forces its neighbors to be unselected, but since p itself is unselected, it does not alter the original solutions. Although, vertex v could be connected to many vertices, the reduction is pathwidth-preserving, as we only have one such vertex in total. $\square$

As a last non-trivial task, it remains to remove this special unselected vertex. We remove this vertex by the reduction from Lemma 10.8.13 which we prove in Section 10.8.2.3.

Lemma 10.8.13. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . If we are given a strong candidate $\mathcal{J}$ for $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$, then $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\emptyset, \{0\})} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$, where $(\emptyset, \{0\})$ is 1-bounded in the source problem.*

With Lemma 10.8.13 we have everything ready to prove the main result of this section in the straightforward way. We refer to Figure 10.14 for an illustration of the proof outline.

Lemma 10.8.10. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . If $\rho \neq \mathbb{N}$, then $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\emptyset, \rho-r_{\text{top}}+1)} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$.*

Proof. We first use Lemma 10.8.6 with $i = r_{\text{top}} - 1$ to obtain a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\emptyset, \rho-(r_{\text{top}}-1))}$ to $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\sigma, \emptyset)+(\emptyset, \rho)}$.

Then, denote by $\mathcal{J} = (J, \{v\})$ the gadget from Lemma 10.8.11. We know that $\mathcal{J}$ is either a winner or a strong candidate for $(\sigma, \rho)\text{-}\#\text{DOMSET}$. Note that $\mathcal{J}$ is also a winner or strong candidate for $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$ that does not use vertices from $\mathcal{P}$.

If $\mathcal{J}$ is a winner, then we reduce from $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\sigma, \emptyset)+(\emptyset, \rho)}$ to $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$ by Lemma 10.8.12 which concludes the proof in this case.

Therefore, we assume from now on that $\mathcal{J}$ is a strong candidate. We denote by $\mathcal{J}'$ the winner for $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\emptyset, \rho)}$ that we obtain from Lemma 10.8.8 when applied to $\mathcal{J}$. Now, we use Lemma 10.8.12 once more, to obtain a reduction from $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\sigma, \emptyset)+(\emptyset, \rho)}$ to $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\emptyset, \rho)}$. By Lemma 10.8.9, we can reduce to $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\emptyset, \{0\})}$ where $(\emptyset, \{0\})$ is 1-bounded. As the gadget $\mathcal{J}$ fulfills the requirements of Lemma 10.8.13, we obtain the final reduction to $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$. $\square$

In the remainder, we focus on the proof of Lemma 10.8.11 in Section 10.8.2.1, the proof of Lemma 10.8.12 in Section 10.8.2.2, and the proof of Lemma 10.8.13 in Section 10.8.2.3.

10.8.2.1 Constructing Strong Candidates and Winners

We start with the proof of Lemma 10.8.11. Recall the definitions of (strong) candidates and winners.

Definition 10.8.7 (Candidates and Winners). *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some possibly empty set of pairs from Ω_0^2 . Let $\mathcal{G}$ be a gadget for (σ, ρ) -#DOMSET $^{\mathcal{P}}$ with a single portal. We say that $\mathcal{G}$ is a candidate if it has the following properties:*

- (I) $\text{ext}_{\mathcal{G}}(\rho_0), \text{ext}_{\mathcal{G}}(\sigma_0) \geq 1$,
- (II) $\text{ext}_{\mathcal{G}}(\rho_0) \neq \text{ext}_{\mathcal{G}}(\sigma_0)$,
- (III) $\text{ext}_{\mathcal{G}}(\sigma_i) = \text{ext}_{\mathcal{G}}(\rho_i) = 0$ for all $i \geq 2$.

We say that a candidate $\mathcal{G}$ is a winner if additionally it holds that

- (IV) $\text{ext}_{\mathcal{G}}(\rho_1) = \text{ext}_{\mathcal{G}}(\sigma_1) = 0$.

We say that a candidate $\mathcal{G}$ is a strong candidate if (instead of Condition (IV)) it additionally holds that

- (V) $\text{ext}_{\mathcal{G}}(\rho_1) \geq 1$.
- (VI) $\text{ext}_{\mathcal{G}}(\rho_0) \neq \text{ext}_{\mathcal{G}}(\sigma_0) + \text{ext}_{\mathcal{G}}(\sigma_1)$.

In the following, we show the existence of such gadgets.

Lemma 10.8.11. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$. If $\rho \neq \mathbb{N}$, then there is a gadget $\mathcal{G} = (G, \{p\})$ for (σ, ρ) -#DOMSET that is either a winner or a strong candidate.*

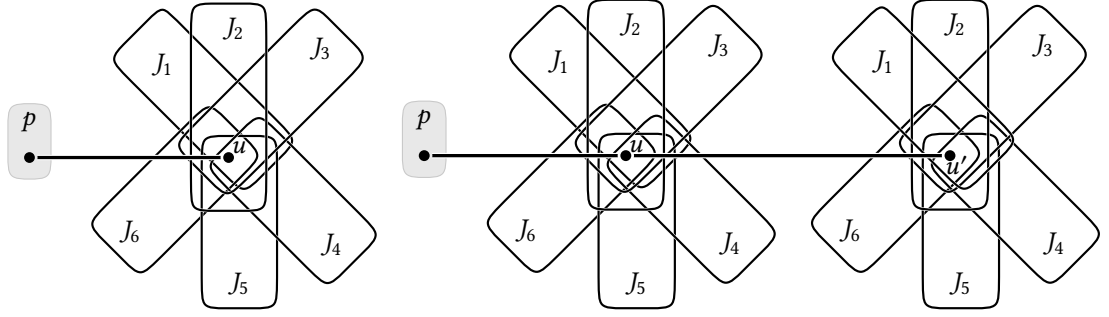
Proof. We start the construction with an auxiliary gadget.

Claim 10.8.11.1. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$. Then, there is a gadget $\mathcal{J} = (J, \{p\})$ for (σ, ρ) -#DOMSET such that $\text{ext}_{\mathcal{J}}(\rho_0), \text{ext}_{\mathcal{J}}(\rho_1) \geq 1$, but $\text{ext}_{\mathcal{J}}(\rho_i) = 0$ for each $i \geq 2$.*

Proof of Claim. Since $\rho \neq \{0\}$ by non-triviality, there is an $r \in \rho$ with $r \geq 1$. Hence, for some $s \in \sigma$, we can apply Lemma 10.2.1 to obtain a $\{\sigma_s, \rho_r\}$ -provider G with portal u such that G has two disjoint solutions S_0 and S_1 for which $u \notin S_0$ with $|N(u) \cap S_0| = r$, and $u \in S_1$ with $|N(u) \cap S_1| = s \in \sigma$. Observe that the size of G depends only on σ and ρ , and therefore, the size of all gadgets defined in this proof ultimately depend only on σ and ρ .

Then, we create a new vertex p and connect it to the vertex u of the gadget G . We define J as the resulting gadget, with p as portal.

Note that each partial solution of the gadget $(J, \{p\})$ in which p is unselected is a solution of (σ, ρ) -#DOMSET when restricted to G (u has a feasible number of selected neighbors in G). Since such solutions exist independently of the selection status of u (Lemma 10.2.1), both states ρ_0 and ρ_1 (of the portal p) can be extended by $(J, \{p\})$. Since p only has a single neighbor in J , there are no extensions for ρ_i if $i \geq 2$. $\triangleleft$



(a) The gadget $\mathcal{Z} = (Z, \{p\})$ with portal p for $x = 6$. (b) The modified gadget $\mathcal{Z}^* = (Z^*, \{p\})$ resulting from combining the gadget $\mathcal{Z} = (Z, \{p\})$ with a copy $\mathcal{Z}' = (Z', \{p'\})$ where $p' = u$.

Figure 10.17: The constructions of the gadgets from Lemma 10.8.11, where J denotes the gadget from Claim 10.8.11.1.

By Claim 10.8.11.1, the gadget $\mathcal{J}$ has $f_i := \text{ext}_{\mathcal{J}}(\rho_i) \geq 1$ if and only if $i \in \{0, 1\}$. For a yet to be determined value x , let $(J_1, \{u_1\}), \dots, (J_x, \{u_x\})$ be x copies of the gadget $\mathcal{J}$. From this we obtain a graph Z by identifying the vertices $u_1, \dots, u_x$ to a single vertex u , which is then adjacent to the portal p of $\mathcal{Z} = (Z, \{p\})$. See Figure 10.17a for an illustration.

Now, consider extensions of the states ρ_0 and σ_0 of $\mathcal{Z}$ depending on whether σ and ρ are finite or cofinite. Since in both cases the vertex u , that is, the unique neighbor of portal p in Z , is unselected, each of the attached copies of $\mathcal{J}$ is in state ρ_0 or ρ_1 . Then, we obtain a solution whenever u obtains a feasible number of selected neighbors (from p and its neighbors in the attached copies of $\mathcal{J}$).

If ρ is finite, we have

$$\begin{aligned} \text{ext}_{\mathcal{Z}}(\rho_0) &= \sum_{r \in \rho} \binom{x}{r} \cdot f_1^r \cdot f_0^{x-r} \\ &= f_0^x \cdot \sum_{r \in \rho} \binom{x}{r} \cdot \left(\frac{f_1}{f_0}\right)^r \end{aligned}$$

and analogously

$$\text{ext}_{\mathcal{Z}}(\sigma_0) = f_0^x \cdot \sum_{r \in \rho, r \geq 1} \binom{x}{r-1} \cdot \left(\frac{f_1}{f_0}\right)^{r-1}.$$

Now, if ρ is cofinite, we have

$$\begin{aligned} \text{ext}_{\mathcal{Z}}(\rho_0) &= (f_0 + f_1)^x - \sum_{r \notin \rho} \binom{x}{r} \cdot f_1^r \cdot f_0^{x-r} \\ &= (f_0 + f_1)^x - f_0^x \cdot \sum_{r \notin \rho, r \geq 0} \binom{x}{r} \cdot \left(\frac{f_1}{f_0}\right)^r \end{aligned}$$

and analogously

$$\text{ext}_{\mathcal{Z}}(\sigma_0) = (f_0 + f_1)^x - f_0^x \cdot \sum_{r \notin \rho, r \geq 1} \binom{x}{r-1} \cdot \left(\frac{f_1}{f_0}\right)^{r-1}.$$

Claim 10.8.11.2. *There is an integer x_0 such that, for all $x \geq x_0$, $\mathcal{Z}$ is a candidate.*

Proof of Claim. We have to show properties (I), (II), and (III) from Definition 10.8.7.

First, consider the case where ρ is finite. Then, $\text{ext}_{\mathcal{Z}}(\rho_0)/f_0^x$ is a polynomial in x of degree r_{top} , whereas $\text{ext}_{\mathcal{Z}}(\sigma_0)/f_0^x$ is a polynomial of degree $r_{\text{top}} - 1$. Since $r_{\text{top}} \geq 1$ by the fact that (σ, ρ) is non-trivial and ρ is finite, it follows that these polynomials are not constant 0. Hence, there is a value of x that depends only on ρ for which $\text{ext}_{\mathcal{Z}}(\rho_0) \neq \text{ext}_{\mathcal{Z}}(\sigma_0)$ and $\text{ext}_{\mathcal{Z}}(\rho_0), \text{ext}_{\mathcal{Z}}(\sigma_0) \geq 1$, as required.

Second, suppose that ρ is cofinite. Then, $\sum_{r \notin \rho} \binom{x}{r} \cdot (f_1/f_0)^r$ is a polynomial in x . Its degree is $r_{\text{top}} - 1$, which is the largest non-negative integer that is missing from ρ . Since $\rho \neq \mathbb{N}$, we have $r_{\text{top}} \geq 1$ and the polynomial is not constant 0 but possibly a constant if $r_{\text{top}} = 1$. Similarly, $\sum_{r \notin \rho, r \geq 1} \binom{x}{r-1} \cdot (f_1/f_0)^{r-1}$ is a polynomial in x with degree $r_{\text{top}} - 2$. In this case, the sum might be empty (and thus, equal to 0), but in any case the expressions for $\text{ext}_{\mathcal{Z}}(\rho_0)$ and $\text{ext}_{\mathcal{Z}}(\sigma_0)$ are positive and distinct for all sufficiently large x .

Finally, since p has only a single neighbor in $\mathcal{Z}$, there are no extensions in $\mathcal{Z}$ for states σ_i or ρ_i if $i \geq 2$. $\triangleleft$

Now, we are in one of three cases.

Case 1: $\text{ext}_{\mathcal{Z}}(\rho_1) = \text{ext}_{\mathcal{Z}}(\sigma_1) = 0$.

Case 2: $\text{ext}_{\mathcal{Z}}(\rho_1) = 0$ and $\text{ext}_{\mathcal{Z}}(\sigma_1) \geq 1$.

Case 3: $\text{ext}_{\mathcal{Z}}(\rho_1) \geq 1$.

In Case 1, the statement of Lemma 10.8.11 holds. Thus, the candidate $\mathcal{Z}$ is a winner, and we are done. In Case 2, $\mathcal{Z}$ is a candidate, but not a strong candidate. In this case, we define another gadget $\mathcal{Z}^*$ (a modification of $\mathcal{Z}$), and show that $\mathcal{Z}^*$ satisfies the constraint from Case 3. As a consequence, we have to deal only with Case 3. In this third case, the candidate $\mathcal{Z}$ has the property from Item (V), and it remains to show that $\text{ext}_{\mathcal{Z}}(\rho_0) \neq \text{ext}_{\mathcal{Z}}(\sigma_0) + \text{ext}_{\mathcal{Z}}(\sigma_1)$ in order for $\mathcal{Z}$ to be a strong candidate. We start by defining $\mathcal{Z}^*$, and subsequently show how to handle Case 3.

Recall that u is the unique neighbor of the portal p in $\mathcal{Z}$. Let $\mathcal{Z}' = (\mathcal{Z}', \{p'\})$ be another copy of $\mathcal{Z}$. Then, we attach, the gadget $\mathcal{Z}'$ to the vertex u in $\mathcal{Z}$, by identifying u and p' . This forms the new gadget $\mathcal{Z}^* = (\mathcal{Z}^*, \{p\})$. See Figure 10.17b for an illustration of this construction.

Claim 10.8.11.3. *Suppose that $\text{ext}_{\mathcal{Z}}(\rho_1) = 0$ and $\text{ext}_{\mathcal{Z}}(\sigma_1) \geq 1$. Then, $\mathcal{Z}^*$ is a candidate with $\text{ext}_{\mathcal{Z}^*}(\rho_1) \geq 1$. Moreover, $\text{ext}_{\mathcal{Z}^*}(\rho_0) = \text{ext}_{\mathcal{Z}}(\rho_0)^2$ and $\text{ext}_{\mathcal{Z}^*}(\sigma_0) = \text{ext}_{\mathcal{Z}}(\sigma_0) \cdot \text{ext}_{\mathcal{Z}}(\rho_0)$.*

Proof of Claim. As $\mathcal{Z}$ is a candidate with $\text{ext}_{\mathcal{Z}}(\rho_1) = 0$, we have $\text{ext}_{\mathcal{Z}}(\rho_i) = 0$ for $i \geq 1$. Note that $\text{ext}_{\mathcal{Z}^*}(\rho_0) = \text{ext}_{\mathcal{Z}}(\rho_0)^2$ because if u is unselected, then $\mathcal{Z}'$ has to be in state ρ_0 as $\mathcal{Z}'$ is a copy of $\mathcal{Z}$ and $\text{ext}_{\mathcal{Z}}(\rho_i) = 0$ for $i \geq 1$. This does not affect the number of selected

neighbors of u , so every partial solution of $\mathcal{Z}$ that witnesses ρ_0 can be extended by every such partial solution of the copy $\mathcal{Z}'$ to a partial solution of $\mathcal{Z}^*$ that witnesses ρ_0 . Analogously, $\text{ext}_{\mathcal{Z}^*}(\sigma_0) = \text{ext}_{\mathcal{Z}}(\sigma_0) \cdot \text{ext}_{\mathcal{Z}}(\rho_0)$.

Thus, we still have $\text{ext}_{\mathcal{Z}^*}(\rho_0) \neq \text{ext}_{\mathcal{Z}^*}(\sigma_0)$ and $\text{ext}_{\mathcal{Z}^*}(\rho_0), \text{ext}_{\mathcal{Z}^*}(\sigma_0) \geq 1$. Also, as before, portal p has only a single neighbor in $\mathcal{Z}^*$, and consequently, there are no extensions for states σ_i or ρ_i if $i \geq 2$.

We now verify that $\text{ext}_{\mathcal{Z}^*}(\rho_1) \geq 1$. Since $\text{ext}_{\mathcal{Z}}(\sigma_1) \geq 1$, there is a partial solution S of $\mathcal{Z}$ that witnesses σ_1 . Let S' be a copy of this set for the gadget $\mathcal{Z}'$. Then, $S^* := S' \cup S \setminus \{p\}$ is a partial solution of $\mathcal{Z}^*$ that witnesses ρ_1 . The key observation is that u has the same number of selected neighbors as in the partial solution S ; instead of p , vertex u now has a single selected neighbor in $\mathcal{Z}'$. $\blacktriangleleft$

In order to complete the proof of Lemma 10.8.11 for Cases 2 and 3, we set

$$\mathcal{G} := \begin{cases} \mathcal{Z}^* & \text{if } \text{ext}_{\mathcal{Z}}(\rho_1) = 0 \text{ and } \text{ext}_{\mathcal{Z}}(\sigma_1) \geq 1. \\ \mathcal{Z} & \text{if } \text{ext}_{\mathcal{Z}}(\rho_1) \geq 1. \end{cases}$$

As a consequence, we have that $\mathcal{G}$ is a candidate with the properties from Case 3, that is, $\text{ext}_{\mathcal{G}}(\rho_1) \geq 1$. It remains to show Property (VI), that is, show that $\text{ext}_{\mathcal{G}}(\rho_0) \neq \text{ext}_{\mathcal{G}}(\sigma_0) + \text{ext}_{\mathcal{G}}(\sigma_1)$, for $\mathcal{G}$ to be a strong candidate.

To this end, we consider two cases depending on whether ρ is cofinite or finite.

ρ Is Cofinite. In this case, we can directly obtain a strong candidate from $\mathcal{G}$ as follows.

Claim 10.8.11.4. *If ρ is cofinite, then there is an integer x_0 such that, for all $x \geq x_0$, $\mathcal{G}$ is a strong candidate.*

Proof of Claim. It remains to show that $\text{ext}_{\mathcal{G}}(\rho_0) \neq \text{ext}_{\mathcal{G}}(\sigma_0) + \text{ext}_{\mathcal{G}}(\sigma_1)$. For cofinite ρ , recall that, for the candidate $\mathcal{Z}$, we have that

$$\text{ext}_{\mathcal{Z}}(\rho_0) = (f_0 + f_1)^x - f_0^x \cdot \sum_{r \notin \rho, r \geq 0} \binom{x}{r} \cdot \left(\frac{f_1}{f_0}\right)^r$$

and

$$\text{ext}_{\mathcal{Z}}(\sigma_0) = (f_0 + f_1)^x - f_0^x \cdot \sum_{r \notin \rho, r \geq 1} \binom{x}{r-1} \cdot \left(\frac{f_1}{f_0}\right)^{r-1}.$$

For sufficiently large x , we have $\text{ext}_{\mathcal{Z}}(\rho_0) < \text{ext}_{\mathcal{Z}}(\sigma_0)$ and consequently, $\text{ext}_{\mathcal{Z}}(\rho_0) \neq \text{ext}_{\mathcal{Z}}(\sigma_0) + \text{ext}_{\mathcal{Z}}(\sigma_1)$. This concludes the proof if $\mathcal{G} = \mathcal{Z}$.

For the case when $\mathcal{G} = \mathcal{Z}^*$, Claim 10.8.11.3 yields that $\text{ext}_{\mathcal{G}}(\rho_0) = \text{ext}_{\mathcal{Z}}(\rho_0)^2$ and $\text{ext}_{\mathcal{G}}(\sigma_0) = \text{ext}_{\mathcal{Z}}(\sigma_0) \cdot \text{ext}_{\mathcal{Z}}(\rho_0)$. Combined with the previous observations, this implies $\text{ext}_{\mathcal{G}}(\rho_0) < \text{ext}_{\mathcal{G}}(\sigma_0)$. $\blacktriangleleft$

ρ Is Finite. Now, suppose that ρ is finite. We fix x in the definition of $\mathcal{G}$, that is, in the definition of $\mathcal{Z}$ or $\mathcal{Z}^*$, respectively, to be sufficiently large for Claim 10.8.11.2 to hold. We can assume that $\text{ext}_{\mathcal{G}}(\rho_0) = \text{ext}_{\mathcal{G}}(\sigma_0) + \text{ext}_{\mathcal{G}}(\sigma_1)$, as otherwise $\mathcal{G}$ is a strong candidate and we are done. We make one last modification and define $\mathcal{G}' = (G', \{q\})$ as follows.

For a yet to be determined value y , let $(G_1, \{p_1\}), \dots, (G_y, \{p_y\})$ be y copies of the gadget $\mathcal{G}$. We use the same trick as before. We obtain a graph G' by identifying the vertices $p_1, \dots, p_y$ to a single vertex p , which is then adjacent to the portal q of $\mathcal{G}' = (G', \{q\})$. In order to complete the proof of Lemma 10.8.11, we show the following claim.

Claim 10.8.11.5. *If ρ is finite and $\text{ext}_{\mathcal{G}}(\rho_0) = \text{ext}_{\mathcal{G}}(\sigma_0) + \text{ext}_{\mathcal{G}}(\sigma_1)$, then there is an integer y_0 such that, for all $y \geq y_0$, $\mathcal{G}'$ is a strong candidate.*

Proof of Claim. Using that $\text{ext}_{\mathcal{G}}(\rho_0), \text{ext}_{\mathcal{G}}(\rho_1) \geq 1$, the proof that $\mathcal{G}'$ is a candidate for sufficiently large y is analogous to the proof that $\mathcal{Z}$ is a candidate for sufficiently large x .

Since $\mathcal{G}$ is a candidate, we have $\text{ext}_{\mathcal{G}}(\sigma_0) \geq 1$ and $\text{ext}_{\mathcal{G}}(\rho_0) \neq \text{ext}_{\mathcal{G}}(\sigma_0)$. From the fact that $\text{ext}_{\mathcal{G}}(\rho_0) = \text{ext}_{\mathcal{G}}(\sigma_0) + \text{ext}_{\mathcal{G}}(\sigma_1)$, it then follows that $\text{ext}_{\mathcal{G}}(\sigma_1) \geq 1$. To shorten notation, we define

$$\begin{aligned} g_0 &:= \text{ext}_{\mathcal{G}}(\rho_0), & g_1 &:= \text{ext}_{\mathcal{G}}(\rho_1), \\ h_0 &:= \text{ext}_{\mathcal{G}}(\sigma_0), & \text{and } h_1 &:= \text{ext}_{\mathcal{G}}(\sigma_1). \end{aligned}$$

From the properties of $\mathcal{G}$, we have $g_0, g_1, h_0, h_1 \geq 1$, $g_0 \neq h_0$, and $g_0 = h_0 + h_1$.

Then, as before, using the fact that ρ is finite, we have

$$\begin{aligned} \text{ext}_{\mathcal{G}'}(\rho_0) &= g_0^y \cdot \sum_{r \in \rho} \binom{y}{r} \cdot \left(\frac{g_1}{g_0}\right)^r \quad \text{and} \\ \text{ext}_{\mathcal{G}'}(\sigma_0) &= g_0^y \cdot \sum_{r \in \rho, r \geq 1} \binom{y}{r-1} \cdot \left(\frac{g_1}{g_0}\right)^{r-1}. \end{aligned}$$

First, suppose that σ is finite, then we analogously have

$$\begin{aligned} \text{ext}_{\mathcal{G}'}(\rho_1) &= h_0^y \cdot \sum_{s \in \sigma} \binom{y}{s} \cdot \left(\frac{h_1}{h_0}\right)^s \quad \text{and} \\ \text{ext}_{\mathcal{G}'}(\sigma_1) &= h_0^y \cdot \sum_{s \in \sigma, s \geq 1} \binom{y}{s-1} \cdot \left(\frac{h_1}{h_0}\right)^{s-1}. \end{aligned}$$

We conclude that $\text{ext}_{\mathcal{G}'}(\rho_1) \geq 1$ as $h_0, h_1 \geq 1$ and the corresponding sum is not empty. Hence, $\mathcal{G}'$ has Property (V).

Note that there are three polynomials P_1, P_2 , and P_3 in y such that

$$\begin{aligned} \text{ext}_{\mathcal{G}'}(\rho_0) &= g_0^y \cdot P_1(y), \\ \text{ext}_{\mathcal{G}'}(\sigma_0) &= g_0^y \cdot P_2(y), \quad \text{and} \\ \text{ext}_{\mathcal{G}'}(\sigma_1) &= h_0^y \cdot P_3(y), \end{aligned}$$

where $P_1(y) > P_2(y) + 1$ (for sufficiently large y) since $\rho \neq \{0\}$ as (σ, ρ) is non-trivial.

Thus, from the fact that $g_0 = h_0 + h_1$ with $h_1 \geq 1$, and consequently, $g_0 > h_0$, for sufficiently large y we have

$$\frac{\text{ext}_{\mathcal{G}'}(\sigma_0) + \text{ext}_{\mathcal{G}'}(\sigma_1)}{\text{ext}_{\mathcal{G}'}(\rho_0)} = \frac{g_0^y \cdot P_2(y) + h_0^y \cdot P_3(y)}{g_0^y \cdot P_1(y)} = \frac{P_2(y)}{P_1(y)} + \frac{h_0^y \cdot P_3(y)}{g_0^y \cdot P_1(y)} < 1.$$

This shows that $\mathcal{G}'$ is a strong candidate if σ is finite.

It remains to consider the case when σ is cofinite. The expressions for $\text{ext}_{\mathcal{G}'}(\rho_0)$ and $\text{ext}_{\mathcal{G}'}(\sigma_0)$ are the same as in the previous case, but

$$\begin{aligned} \text{ext}_{\mathcal{G}'}(\rho_1) &= (h_0 + h_1)^y - h_0^y \cdot \sum_{s \notin \sigma} \binom{y}{s} \cdot \left(\frac{h_1}{h_0}\right)^s \quad \text{and} \\ \text{ext}_{\mathcal{G}'}(\sigma_1) &= (h_0 + h_1)^y - h_0^y \cdot \sum_{s \notin \sigma, s \geq 1} \binom{y}{s-1} \cdot \left(\frac{h_1}{h_0}\right)^{s-1}. \end{aligned}$$

Clearly, $\text{ext}_{\mathcal{G}'}(\rho_1) \geq 1$ since we can choose y to be sufficiently large. Moreover, $\text{ext}_{\mathcal{G}'}(\sigma_1) \leq (h_0 + h_1)^y = g_0^y$ by assumption. Hence, for sufficiently large y , we have

$$\frac{\text{ext}_{\mathcal{G}'}(\sigma_0) + \text{ext}_{\mathcal{G}'}(\sigma_1)}{\text{ext}_{\mathcal{G}'}(\rho_0)} \leq \frac{g_0^y \cdot P_2(y) + g_0^y}{g_0^y \cdot P_1(y)} = \frac{P_2(y) + 1}{P_1(y)} < 1.$$

This shows that $\mathcal{G}'$ is a strong candidate if σ is cofinite. ◀

This concludes the construction of the gadget which is either a winner or a strong candidate. □

10.8.2.2 Winners Force the Selection Status

In the following we show how to use the winners from Lemma 10.8.11 to force the selection status of vertices. As we heavily rely on the properties of the winner, we restate the definition here for convenience.

Definition 10.8.7 (Candidates and Winners). *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some possibly empty set of pairs from Ω_0^2 . Let $\mathcal{G}$ be a gadget for $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}}$ with a single portal. We say that $\mathcal{G}$ is a candidate if it has the following properties:*

- (I) $\text{ext}_{\mathcal{G}}(\rho_0), \text{ext}_{\mathcal{G}}(\sigma_0) \geq 1$,
- (II) $\text{ext}_{\mathcal{G}}(\rho_0) \neq \text{ext}_{\mathcal{G}}(\sigma_0)$,
- (III) $\text{ext}_{\mathcal{G}}(\sigma_i) = \text{ext}_{\mathcal{G}}(\rho_i) = 0$ for all $i \geq 2$.

We say that a candidate $\mathcal{G}$ is a winner if additionally it holds that

- (IV) $\text{ext}_{\mathcal{G}}(\rho_1) = \text{ext}_{\mathcal{G}}(\sigma_1) = 0$.

We say that a candidate $\mathcal{G}$ is a strong candidate if (instead of Condition (IV)) it additionally holds that

$$(V) \text{ ext}_{\mathcal{G}}(\rho_1) \geq 1.$$

$$(VI) \text{ ext}_{\mathcal{G}}(\rho_0) \neq \text{ext}_{\mathcal{G}}(\sigma_0) + \text{ext}_{\mathcal{G}}(\sigma_1).$$

Now, the main work is to show that selected and unselected vertices can always be modeled, that is, prove Lemma 10.8.12. We show that this is the case whenever a “winner” exists.

Lemma 10.8.12. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . If we are given a winner $\mathcal{J} = (J, \{v\})$ for (σ, ρ) -#DOMSET $^{\mathcal{P}}$, then (σ, ρ) -#DOMSET $^{\mathcal{P}+(\emptyset, \rho)+(\sigma, \emptyset)} \leq_{\text{pw}} (\sigma, \rho)$ -#DOMSET $^{\mathcal{P}}$.*

Proof. Let $\mathcal{J} = (J, \{v\})$ be the winner for (σ, ρ) -#DOMSET $^{\mathcal{P}}$.

The proof is split into two steps where we first show the reduction from (σ, ρ) -#DOMSET $^{\mathcal{P}+(\emptyset, \rho)}$ to (σ, ρ) -#DOMSET $^{\mathcal{P}}$, that is, how to force vertices to be unselected. For the second step, we show the reduction from (σ, ρ) -#DOMSET $^{\mathcal{P}+(\sigma, \emptyset)+(\emptyset, \rho)}$ to (σ, ρ) -#DOMSET $^{\mathcal{P}+(\emptyset, \rho)}$, that is, how to force vertices to be selected given that one can already force unselected vertices.

Claim 10.8.12.1. (σ, ρ) -#DOMSET $^{\mathcal{P}+(\emptyset, \rho)} \leq_{\text{pw}} (\sigma, \rho)$ -#DOMSET $^{\mathcal{P}}$.

Proof of Claim. As $\mathcal{J}$ is a winner, then the only states of $\mathcal{J}$ with non-zero extensions are ρ_0 and σ_0 . In particular, the portal v never receives any selected neighbors within $\mathcal{J}$.

Let $I = (G, \lambda)$ be an instance of (σ, ρ) -#DOMSET $^{\mathcal{P}+(\emptyset, \rho)}$, and let $U = \{u_1, \dots, u_k\}$ be the set of $(\emptyset, \rho)$ -vertices in G . We obtain the instance $I' = (G', \lambda')$ from I by replacing the $(\emptyset, \rho)$ -vertices in U by (σ, ρ) -vertices.

For a positive integer x , we define an instance $I_x = (G_x, \lambda_x)$ of (σ, ρ) -#DOMSET $^{\mathcal{P}}$ based on I' by attaching x new copies of the gadget $\mathcal{J}$ to each vertex $u \in U$, where u is identified with the portal of each attached copy. The function λ_x is identical to λ' and the vertices of the attached copies of $\mathcal{J}$ stay as defined in $\mathcal{J}$.

Let $\#Sol(I_x)$ denote the number of solutions of (σ, ρ) -#DOMSET $^{\mathcal{P}}$ on input I_x . Further, let a_i be the number of solutions of (σ, ρ) -#DOMSET $^{\mathcal{P}}$ on instance I' for which precisely i of the vertices in U are selected. Then, our goal is to compute a_0 since this is precisely the number of solutions on instance I .

We recover a_0 from $\#Sol(I_x)$ using interpolation. We observe that

$$\begin{aligned} \#Sol(I_x) &= \sum_{i=0}^{|U|} a_i \cdot \text{ext}_{\mathcal{J}}(\sigma_0)^{x \cdot i} \cdot \text{ext}_{\mathcal{J}}(\rho_0)^{x \cdot (|U| - i)} \\ &= \text{ext}_{\mathcal{J}}(\rho_0)^{x \cdot |U|} \cdot \sum_{i=0}^{|U|} a_i \cdot \left(\frac{\text{ext}_{\mathcal{J}}(\sigma_0)}{\text{ext}_{\mathcal{J}}(\rho_0)} \right)^{x \cdot i}. \end{aligned}$$

Hence, $\#Sol(I_x) / \text{ext}_{\mathcal{J}}(\rho_0)^{x \cdot |U|}$ is a polynomial of degree $|U|$ with coefficients $a_0, \dots, a_{|U|}$ and indeterminate $(\text{ext}_{\mathcal{J}}(\sigma_0) / \text{ext}_{\mathcal{J}}(\rho_0))^x$. Since $\mathcal{J}$ is a winner, we have $\text{ext}_{\mathcal{J}}(\rho_0), \text{ext}_{\mathcal{J}}(\sigma_0) \geq 1$, and $\text{ext}_{\mathcal{J}}(\rho_0) \neq \text{ext}_{\mathcal{J}}(\sigma_0)$. Thus, we can recover the coefficients using polynomial interpolation with $|U|$ distinct values of x , which gives $|U|$ distinct values of the indeterminate. In particular, we obtain a_0 as required. $\triangleleft$

Note that despite the identical approach, we have to apply the interpolation twice, with different sets U .

For the next claim, we use exactly the same approach as for the previous one.

Claim 10.8.12.2. $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}+(\sigma, \emptyset)+(\emptyset, \rho)} \leq_{pw} (\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}+(\emptyset, \rho)}$.

Proof of Claim. Observe that the coefficient $a_{|U|}$ in the proof of the previous claim corresponds to the number of solutions for which all vertices in U are selected. This is precisely what we need in order to model the vertices in U as $(\sigma, \emptyset)$ -vertices. Hence, with the same proof, we obtain the reduction. $\triangleleft$

The proof now follows from the combination of Claims 10.8.12.1 and 10.8.12.2. $\square$

10.8.2.3 Forcing a Single Unselected Vertex with No Selected Neighbors

In the following we show how to force a single vertex to be unselected and to have no selected neighbors in a solution. In contrast to the results in Section 10.8.2.2, we now use a strong candidate as requirement. We restate the definition once more.

Definition 10.8.7 (Candidates and Winners). *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some possibly empty set of pairs from Ω_0^2 . Let $\mathcal{G}$ be a gadget for $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}}$ with a single portal. We say that $\mathcal{G}$ is a candidate if it has the following properties:*

- (I) $\text{ext}_{\mathcal{G}}(\rho_0), \text{ext}_{\mathcal{G}}(\sigma_0) \geq 1$,
- (II) $\text{ext}_{\mathcal{G}}(\rho_0) \neq \text{ext}_{\mathcal{G}}(\sigma_0)$,
- (III) $\text{ext}_{\mathcal{G}}(\sigma_i) = \text{ext}_{\mathcal{G}}(\rho_i) = 0$ for all $i \geq 2$.

We say that a candidate $\mathcal{G}$ is a winner if additionally it holds that

- (IV) $\text{ext}_{\mathcal{G}}(\rho_1) = \text{ext}_{\mathcal{G}}(\sigma_1) = 0$.

We say that a candidate $\mathcal{G}$ is a strong candidate if (instead of Condition (IV)) it additionally holds that

- (V) $\text{ext}_{\mathcal{G}}(\rho_1) \geq 1$.
- (VI) $\text{ext}_{\mathcal{G}}(\rho_0) \neq \text{ext}_{\mathcal{G}}(\sigma_0) + \text{ext}_{\mathcal{G}}(\sigma_1)$.

Now we prove Lemma 10.8.13.

Lemma 10.8.13. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . If we are given a strong candidate $\mathcal{J}$ for $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}}$, then $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}+(\emptyset, \{0\})} \leq_{pw} (\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}}$, where $(\emptyset, \{0\})$ is 1-bounded in the source problem.*

Proof. Let $I = (G, \lambda)$ be an instance of $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}+(\emptyset, \{0\})}$, and let $n = |V(G)|$. If G contains no $(\emptyset, \{0\})$ -vertex, then we can directly call the $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}}$ -oracle on I . So, suppose that p is the unique $(\emptyset, \{0\})$ -vertex in G .

Note that $\mathcal{G} = (G, \{p\}, \lambda)$ can be interpreted as a gadget for $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}}$ with portal p . Then, our goal is to compute $\text{ext}_{\mathcal{G}}(\rho_0)$, as this corresponds to the number of partial solutions

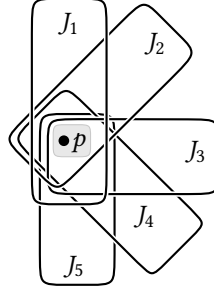


Figure 10.18: The construction of the gadget $\mathcal{J}^{(x)}$ for $x = 5$ from Lemma 10.8.13. This gadget is used to replace a single $(\emptyset, \{0\})$ -vertex p .

of $\mathcal{G}$ for which p is unselected and has no selected neighbors in $\mathcal{G}$. These are precisely the solutions of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\emptyset, \{0\})}$ on input I .

The proof outline is as follows. For an integer x , we use x copies of the strong candidate $\mathcal{J}$ to define a gadget $\mathcal{J}^{(x)} := (J^{(x)}, \{p\})$. The main idea is to combine the gadgets $\mathcal{G}$ and $\mathcal{J}^{(x)}$ into an instance I_x of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$. Then, using the oracle for $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$, we can compute the number of solutions for I_x . By our design of $\mathcal{J}^{(x)}$, then we are able to use polynomial interpolation over x to recover the number of extensions for $\mathcal{G}$ from these values. In particular, we can recover the value of $\text{ext}_{\mathcal{G}}(\rho_0)$ which corresponds to the number of solutions for I as outlined above.

Definition of I_x . For a positive integer x , we define an instance $I_x = (G_x, \lambda_x)$ of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$ as follows. Intuitively, in I_x , the vertex p is replaced by a (σ, ρ) -vertex to which we attach x copies of the strong candidate $\mathcal{J}$, where p acts as portal for each of them. Note that since $\mathcal{J}$ is a gadget for $(\sigma, \rho)\text{-}\#\text{DOMSET}$, it is also a gadget for $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$ that happens to use only (σ, ρ) -vertices.

More formally, for each $i \in [1..x]$, let $(J_i, \{v_i\})$ be a copy of $\mathcal{J}$. The graph G_x is obtained from G by identifying all vertices in $\{p, v_1, \dots, v_x\}$ to a single vertex. Then, λ_x is identical to λ on the vertices in $V(G) \setminus \{p\}$, and maps the remaining vertices to 0, that is, the vertices in $\bigcup_{i=1}^x V(J_i)$ are (σ, ρ) -vertices.

With this definition, we define $J^{(x)}$ as the graph induced by the union of the x copies of $\mathcal{J}$ that are attached to p in G_x . As mentioned above, $\mathcal{J}^{(x)} := (J^{(x)}, \{p\})$ can be cast as a gadget for $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$. See Figure 10.18 for an illustration of the construction.

We first analyze the pathwidth of I_x . Since all copies of $\mathcal{J}$ are disjoint (except for the portal p), we duplicate, for each such copy of $\mathcal{J}$, one bag from the original path decomposition of G , and add the vertices of the corresponding copy of $\mathcal{J}$ to it. If we create a fresh copy of the bag for each copy of $\mathcal{J}$, then the pathwidth increases by at most the size of the strong candidate $\mathcal{J}$. As the size of a strong candidate depends only on σ and ρ , it is bounded by a constant, and thus, the reduction is pathwidth-preserving.

Here, we drop the λ -term since λ_x is constant 0 on $J^{(x)}$.

Properties of the Gadgets and Graphs. To analyze the constructions and gadgets from above, the usual definitions of r_{top} and s_{top} are not convenient. Instead, we use r^* as the

maximum element of ρ if ρ is finite, and the maximum missing integer from ρ if ρ is cofinite. Analogously, we define s^* for σ . In the following, we always assume that $x \geq \max\{s^*, r^*\}$.

Let $\#Sol(I_x)$ denote the number of solutions of (σ, ρ) -#DOMSET ^{P} on input I_x . When we write A_x for the number of solutions in I_x when p is not selected, that is,

$$A_x = \sum_{i+j \in \rho} \text{ext}_{\mathcal{G}}(\rho_i) \cdot \text{ext}_{\mathcal{J}^{(x)}}(\rho_j), \quad (10.8.1)$$

and when we write B_x for the number of solutions in I_x when p is selected, that is,

$$B_x = \sum_{i+j \in \sigma} \text{ext}_{\mathcal{G}}(\sigma_i) \cdot \text{ext}_{\mathcal{J}^{(x)}}(\sigma_j), \quad (10.8.2)$$

then we have

$$\#Sol(I_x) = A_x + B_x. \quad (10.8.3)$$

Analysis of A_x . We first focus on the more detailed analysis of A_x , which corresponds to the case when p is not selected. The results for B_x then follow analogously. By the definition of $\mathcal{J}^{(x)}$, we get that $\text{ext}_{\mathcal{J}^{(x)}}(\rho_r) = 0$ for all $r > x$. By the definition of a strong candidate, we have $\text{ext}_{\mathcal{J}}(\rho_j) \geq 1$ if and only if $j \in \{0, 1\}$. Thus, in order for p to have state ρ_r in $\mathcal{J}^{(x)}$, there are precisely r copies of $\mathcal{J}$ in which p has state ρ_1 , and in the remaining $x - r$ copies it has state ρ_0 . Hence, for $r \in [0 \dots x]$, we have

$$\text{ext}_{\mathcal{J}^{(x)}}(\rho_r) = \binom{x}{r} \cdot \text{ext}_{\mathcal{J}}(\rho_1)^r \cdot \text{ext}_{\mathcal{J}}(\rho_0)^{x-r}.$$

For ease of notation, we rewrite this expression as

$$\text{ext}_{\mathcal{J}^{(x)}}(\rho_r) = \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \cdot a_r(x) \quad (10.8.4)$$

where

$$a_r(x) := \binom{x}{r} \cdot \text{ext}_{\mathcal{J}}(\rho_1)^r \cdot \text{ext}_{\mathcal{J}}(\rho_0)^{r^*-r}. \quad (10.8.5)$$

Observe that $a_r(x)$ is a polynomial of degree r in x since $\text{ext}_{\mathcal{J}}(\rho_0)$ and $\text{ext}_{\mathcal{J}}(\rho_1)$ are positive integers by our assumptions about the gadget $\mathcal{J}$. Moreover, if $r \leq r^* \leq x$, then $a_r(x)$ is a positive integer.

In the following, it is convenient to consider a state of p in $\mathcal{G}$, and then analyze the number of corresponding extensions to $\mathcal{J}^{(x)}$ that give a feasible selection for I_x overall. Essentially, given that p has precisely i selected neighbors in $\mathcal{G}$, the extensions to $\mathcal{J}^{(x)}$ have to complement this state of p in $\mathcal{G}$ such that p has a feasible number of neighbors overall.

We define $f_i(x)$ as the number of extensions of $\mathcal{J}^{(x)}$ to a solution of I_x , given that p is unselected and already has i selected neighbors in $\mathcal{G}$, that is, vertex p has state ρ_i in $\mathcal{G}$. Formally, with

$$\alpha := \text{ext}_{\mathcal{J}}(\rho_1) + \text{ext}_{\mathcal{J}}(\rho_0) \quad (10.8.6)$$

and the observation that

$$\alpha^x = (\text{ext}_{\mathcal{J}}(\rho_1) + \text{ext}_{\mathcal{J}}(\rho_0))^x = \sum_{r=0}^x \text{ext}_{\mathcal{J}(x)}(\rho_r),$$

we define

$$f_i(x) := \begin{cases} \sum_{i+r \in \rho} \text{ext}_{\mathcal{J}(x)}(\rho_r) & \text{if } \rho \text{ is finite,} \\ \alpha^x - \sum_{i+r \notin \rho} \text{ext}_{\mathcal{J}(x)}(\rho_r) & \text{if } \rho \text{ is cofinite.} \end{cases}$$

With Equation (10.8.4), we have

$$f_i(x) = \begin{cases} \sum_{i+r \in \rho} \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \cdot a_r(x) & \text{if } \rho \text{ is finite,} \\ \alpha^x - \sum_{i+r \notin \rho} \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \cdot a_r(x) & \text{if } \rho \text{ is cofinite,} \end{cases}$$

which we can simplify to

$$f_i(x) = \begin{cases} \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \cdot \sum_{i+r \in \rho} a_r(x) & \text{if } \rho \text{ is finite} \\ \alpha^x - \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \cdot \sum_{i+r \notin \rho} a_r(x) & \text{if } \rho \text{ is cofinite.} \end{cases} \quad (10.8.7)$$

Now we revisit the definition of A_x and combine the previous observations. Denote by δ the degree of p in G . From the above arguments and the definition of $f_i(x)$, we get

$$A_x = \sum_{i+j \in \rho} \text{ext}_{\mathcal{G}}(\rho_i) \cdot \text{ext}_{\mathcal{J}(x)}(\rho_j) = \sum_{i=0}^{\delta} \text{ext}_{\mathcal{G}}(\rho_i) \cdot f_i(x). \quad (10.8.8)$$

Next, we replace $f_i(x)$ by Equation (10.8.7). If ρ is finite, then we have

$$A_x = \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \cdot \underbrace{\sum_{i=0}^{\delta} \text{ext}_{\mathcal{G}}(\rho_i) \cdot \sum_{i+r \in \rho} a_r(x)}_{a(x) :=} \quad (10.8.9)$$

and if ρ is cofinite, then we have

$$A_x = \alpha^x \cdot \sum_{i=0}^{\delta} \text{ext}_{\mathcal{G}}(\rho_i) - \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \cdot \underbrace{\sum_{i=0}^{\delta} \text{ext}_{\mathcal{G}}(\rho_i) \cdot \sum_{i+r \notin \rho} a_r(x)}_{a(x) :=}. \quad (10.8.10)$$

In Equation (10.8.9) together with Equation (10.8.10), we define an expression $a(x)$ depending on whether ρ is finite or cofinite. We later use the crucial fact that $a(x)$ is a polynomial in x because, for each r , $a_r(x)$ is a polynomial in x .

Analysis of B_x . For the case when the vertex p is selected, we get the following results analogously to the previous observations. Similarly to before, we have $\text{ext}_{\mathcal{J}(x)}(\sigma_j) = 0$ for all $j > x$. Moreover, for all $s \in [0..x]$, we have

$$\text{ext}_{\mathcal{J}(x)}(\sigma_s) = \text{ext}_{\mathcal{J}}(\sigma_0)^{x-s^*} \cdot b_s(x) \quad (10.8.11)$$

where

$$b_s(x) := \binom{x}{s} \cdot \text{ext}_{\mathcal{J}}(\sigma_1)^s \cdot \text{ext}_{\mathcal{J}}(\sigma_0)^{s^*-s}. \quad (10.8.12)$$

We observe that $b_s(x)$ is a polynomial in x . However, in contrast to $a_r(x)$, the value $b_s(x)$ is not necessarily a positive integer as the definition of the strong candidate $\mathcal{J}$ does not guarantee that $\text{ext}_{\mathcal{J}}(\sigma_1)$ is non-zero.

Analogously to the previous case, we set

$$\beta := \text{ext}_{\mathcal{J}}(\sigma_1) + \text{ext}_{\mathcal{J}}(\sigma_0), \quad (10.8.13)$$

If σ is finite, then we can rewrite B_x as

$$B_x = \text{ext}_{\mathcal{J}}(\sigma_0)^{x-s^*} \cdot \underbrace{\sum_{i=0}^{\delta} \text{ext}_{\mathcal{G}}(\sigma_i) \cdot \sum_{i+s \in \sigma} b_s(x)}_{b(x) :=} \quad (10.8.14)$$

and if σ is cofinite, then we can write

$$B_x = \beta^x \cdot \sum_{i=0}^{\delta} \text{ext}_{\mathcal{G}}(\sigma_i) - \text{ext}_{\mathcal{J}}(\sigma_0)^{x-s^*} \cdot \underbrace{\sum_{i=0}^{\delta} \text{ext}_{\mathcal{G}}(\sigma_i) \cdot \sum_{i+s \notin \sigma} b_s(x)}_{b(x) :=} \quad (10.8.15)$$

Again, $b(x)$ is a polynomial in x because each $b_s(x)$ is a polynomial in x .

Polynomial Interpolation. At this point, let us recall that our goal is to compute $\text{ext}_{\mathcal{G}}(\rho_0)$ which equals the number of solutions of (σ, ρ) -#DOMSET $^{\mathcal{P}+(\emptyset, \{0\})}$ on input I . We aim to use polynomial interpolation to recover A_x as a polynomial in x with $\text{ext}_{\mathcal{G}}(\rho_0)$ as a coefficient. Given the different expressions for A_x and B_x depending on whether σ and ρ are finite or cofinite, we split the proof into two cases at this point. We start with the substantially easier case where both sets are finite.

Both σ and ρ Are Finite. In order to do polynomial interpolation on A_x , we first show how to isolate the term A_x from the value of $\#\text{Sol}(I_x)$, which we can compute by an oracle call.

Claim 10.8.13.1. *For sufficiently large $x \in \mathcal{O}(n)$ and given $\#\text{Sol}(I_x)$, the term A_x can be computed in time polynomial in n .*

Proof of Claim. From the definition of a candidate, we know that $\text{ext}_{\mathcal{J}}(\sigma_0)$ and $\text{ext}_{\mathcal{J}}(\rho_0)$ are distinct positive values. We show how to recover A_x from $\#\text{Sol}(I_x)$ if $\text{ext}_{\mathcal{J}}(\rho_0) > \text{ext}_{\mathcal{J}}(\sigma_0)$. In the case when $\text{ext}_{\mathcal{J}}(\rho_0) < \text{ext}_{\mathcal{J}}(\sigma_0)$, the term B_x can be recovered analogously, and then $A_x = \#\text{Sol}(I_x) - B_x$.

From the combination of Equation (10.8.3) with Equations (10.8.9) and (10.8.14), we obtain

$$\begin{aligned} \#\text{Sol}(I_x) &= A_x + B_x \\ &= \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \cdot \sum_{i=0}^{\delta} \text{ext}_{\mathcal{G}}(\rho_i) \cdot \sum_{i+r \in \rho} a_r(x) \\ &\quad + \text{ext}_{\mathcal{J}}(\sigma_0)^{x-s^*} \cdot \sum_{i=0}^{\delta} \text{ext}_{\mathcal{G}}(\sigma_i) \cdot \sum_{i+s \in \sigma} b_s(x). \end{aligned}$$

Note that the terms $b_s(x)$ are polynomials in x , and that each of the terms $\text{ext}_{\mathcal{G}}(\sigma_i)$ is upper bounded by 2^n . Hence, the outer sum of B_x can be bounded by $q(x) \cdot 2^n$, for some polynomial $q(x)$ in x . Thus, from the fact that $\text{ext}_{\mathcal{J}}(\rho_0) > \text{ext}_{\mathcal{J}}(\sigma_0)$, we can choose $x \in \mathcal{O}(n)$ such that

$$\frac{B_x}{\text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*}} \leq \frac{\text{ext}_{\mathcal{J}}(\sigma_0)^{x-s^*} \cdot q(x) \cdot 2^n}{\text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*}} = \left(\frac{\text{ext}_{\mathcal{J}}(\sigma_0)}{\text{ext}_{\mathcal{J}}(\rho_0)} \right)^x \cdot \frac{\text{ext}_{\mathcal{J}}(\rho_0)^{r^*} \cdot q(x) \cdot 2^n}{\text{ext}_{\mathcal{J}}(\sigma_0)^{s^*}} < 1$$

since $\text{ext}_{\mathcal{J}}(\rho_0) > \text{ext}_{\mathcal{J}}(\sigma_0)$. By the definition of A_x from Equation (10.8.9), we have that $A_x / \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*}$ is an integer.

Therefore, $\lfloor \#\text{Sol}(I_x) / \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \rfloor = A_x / \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*}$, and thus, we can recover A_x as $A_x = \lfloor \#\text{Sol}(I_x) / \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \rfloor \cdot \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*}$. $\triangleleft$

Recall that our ultimate goal is to compute $\text{ext}_{\mathcal{G}}(\rho_0)$ with calls to a (σ, ρ) - $\#\text{DOMSET}^{\mathcal{P}}$ -oracle. By Equation (10.8.9) together with Equation (10.8.5), we have

$$p(x) := \frac{A_x}{\text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*}} = \sum_{i=0}^k \text{ext}_{\mathcal{G}}(\rho_i) \cdot \left(\sum_{i+r \in \rho} \binom{x}{r} \cdot \text{ext}_{\mathcal{J}}(\rho_1)^r \cdot \text{ext}_{\mathcal{J}}(\rho_0)^{r^*-r} \right).$$

Note that $P(x)$ is a polynomial in x whose highest degree term is $\text{ext}_{\mathcal{G}}(\rho_0) \text{ext}_{\mathcal{J}}(\rho_1)^{r^*} \cdot x^{r^*}$. We use polynomial interpolation, to recover the coefficients of $P(x)$ by evaluating the polynomial for $r^* + 1$ distinct values of x . Indeed, by Claim 10.8.13.1 we can compute A_x with a (σ, ρ) - $\#\text{DOMSET}^{\mathcal{P}}$ -oracle call as long as x is sufficiently large in $\mathcal{O}(n)$; and we can compute $\text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*}$ as $\text{ext}_{\mathcal{J}}(\rho_0)$ is a positive integer that does not depend on n . This can be done in time polynomial in x , that is, in time polynomial in n as $x \in \mathcal{O}(n)$.

As the gadget $\mathcal{J}$ is a strong candidate, we know that $\text{ext}_{\mathcal{J}}(\rho_1)$ is non-zero. Hence, we can recover $\text{ext}_{\mathcal{G}}(\rho_0)$ from the coefficient of the highest degree term x^{r^*} .

We summarize that, for finite σ and ρ , we can solve (σ, ρ) - $\#\text{DOMSET}^{\mathcal{P}+(\emptyset, \{0\})}$ on instance I by r^* oracle calls to (σ, ρ) - $\#\text{DOMSET}^{\mathcal{P}}$ on instances of the form I_x .

One of σ or ρ Is Cofinite. In the cofinite case, we encounter an additional obstacle when we want to compute $\text{ext}_{\mathcal{G}}(\rho_0)$ from A_x by polynomial interpolation. If ρ is cofinite, then A_x

contains the unwanted exponential expression α^x by Equation (10.8.10). Instead, we consider a different sum ψ which allows us to eliminate this exponential term. We define

$$\begin{aligned}\psi(x) &:= \#Sol(I_{x+2}) - (\alpha + \beta) \cdot \#Sol(I_{x+1}) + \alpha \cdot \beta \cdot \#Sol(I_x) \\ &= \underbrace{A_{x+2} - (\alpha + \beta) \cdot A_{x+1} + \alpha \cdot \beta \cdot A_x}_{A'_x :=} + \underbrace{B_{x+2} - (\alpha + \beta) \cdot B_{x+1} + \alpha \cdot \beta \cdot B_x}_{B'_x :=}\end{aligned}\quad (10.8.16)$$

where $\alpha = \text{ext}_{\mathcal{J}}(\rho_1) + \text{ext}_{\mathcal{J}}(\rho_0)$ and $\beta = \text{ext}_{\mathcal{J}}(\sigma_1) + \text{ext}_{\mathcal{J}}(\sigma_0)$ as defined in Equations (10.8.6) and (10.8.13).

So, we aim to recover $\text{ext}_{\mathcal{G}}(\rho_0)$ from A'_x , rather than from A_x directly. Before we recover the value A'_x , we define a polynomial $a'(x)$, to shorten notation. Formally, we set

$$a'(x) := \text{ext}_{\mathcal{J}}(\rho_0)^2 \cdot a(x+2) - (\alpha + \beta) \cdot \text{ext}_{\mathcal{J}}(\rho_0) \cdot a(x+1) + a(x) \quad (10.8.17)$$

If ρ is finite, then Equation (10.8.9) and the definition of A'_x yield

$$\begin{aligned}A'_x &= A_{x+2} - (\alpha + \beta) \cdot A_{x+1} + \alpha \cdot \beta \cdot A_x \\ &= \text{ext}_{\mathcal{J}}(\rho_0)^{x+2-r^*} \cdot a(x+2) \\ &\quad - (\alpha + \beta) \cdot \text{ext}_{\mathcal{J}}(\rho_0)^{x+1-r^*} \cdot a(x+1) \\ &\quad + \alpha \cdot \beta \cdot \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \cdot a(x) \\ &= \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \cdot a'(x).\end{aligned}$$

Analogously, if ρ is cofinite, then Equation (10.8.10) and the definition of A'_x yield

$$\begin{aligned}A'_x &= \left(\alpha^{x+2} \cdot \sum_{i=0}^{\delta} \text{ext}_{\mathcal{G}}(\rho_i) - \text{ext}_{\mathcal{J}}(\rho_0)^{x+2-r^*} \cdot a(x+2) \right) \\ &\quad - (\alpha + \beta) \cdot \left(\alpha^{x+1} \cdot \sum_{i=0}^{\delta} \text{ext}_{\mathcal{G}}(\rho_i) - \text{ext}_{\mathcal{J}}(\rho_0)^{x+1-r^*} \cdot a(x+1) \right) \\ &\quad + \alpha \cdot \beta \cdot \left(\alpha^x \cdot \sum_{i=0}^{\delta} \text{ext}_{\mathcal{G}}(\rho_i) - \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \cdot a(x) \right) \\ &= (\alpha^{x+2} - (\alpha + \beta) \cdot \alpha^{x+1} + \alpha \cdot \beta \cdot \alpha^x) \cdot \left(\sum_{i=0}^{\delta} \text{ext}_{\mathcal{G}}(\rho_i) \right) - \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \cdot a'(x) \\ &= -\text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \cdot a'(x).\end{aligned}$$

We conclude that $A'_x = \pm \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \cdot a'(x)$.

The following claim establishes that we can isolate the term A'_x from the value of $\psi(x)$, which we can compute using oracle calls.

Claim 10.8.13.2. *For sufficiently large $x \in \mathcal{O}(n)$, given the three values $\#Sol(I_{x+2})$, $\#Sol(I_{x+1})$, and $\#Sol(I_x)$, we can compute the term A'_x in time polynomial in n .*

Proof of Claim. By the fact that $\mathcal{J}$ is a candidate, we know that $\text{ext}_{\mathcal{J}}(\sigma_0)$ and $\text{ext}_{\mathcal{J}}(\rho_0)$ are distinct positive values. We show how to recover A'_x from $\psi(x)$ if $\text{ext}_{\mathcal{J}}(\rho_0) > \text{ext}_{\mathcal{J}}(\sigma_0)$. Otherwise, that is, when $\text{ext}_{\mathcal{J}}(\rho_0) < \text{ext}_{\mathcal{J}}(\sigma_0)$, we can recover the term B'_x analogously, and set $A'_x = \psi(x) - B'_x$.

Recall that $A'(x) = \pm \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \cdot a'(x)$, where the sign depends on whether ρ is finite or cofinite. Analogously to Equation (10.8.17), we define

$$b'(x) := \text{ext}_{\mathcal{J}}(\sigma_0)^2 \cdot b(x+2) - (\alpha + \beta) \cdot \text{ext}_{\mathcal{J}}(\sigma_0) \cdot b(x+1) + \alpha \cdot \beta \cdot b(x).$$

With this definition, we have $B'(x) = \pm \text{ext}_{\mathcal{J}}(\sigma_0)^{x-s^*} \cdot b'(x)$ where the sign depends on whether σ is finite or cofinite.

Then, we recall that $b(x)$ is a polynomial in x , and that $\text{ext}_{\mathcal{G}}(\sigma_0)$, which contributes to the coefficients of $b(x)$, is upper bounded by 2^n . Thus, we can choose $x \in \mathcal{O}(n)$ such that $B'_x / \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} < 1$, where we use the crucial fact that $\text{ext}_{\mathcal{J}}(\rho_0) > \text{ext}_{\mathcal{J}}(\sigma_0)$. Then, we observe that $a'(x) = A'_x / \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*}$ is an integer as $a_r(x)$ is an integer for $r \leq r^*$, and consequently, $a(x)$ is an integer (see Equations (10.8.5) and (10.8.10) for their definitions).

Therefore, we have

$$\left\lfloor \frac{\psi(x)}{\text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*}} \right\rfloor = \frac{A'_x}{\text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*}} + \left\lfloor \frac{B'_x}{\text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*}} \right\rfloor = \pm a'(x),$$

and we can compute $A'_x = \lfloor \psi(x) / \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} \rfloor \cdot \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*}$, that is, recover A'_x . $\triangleleft$

Recall that our ultimate goal is to compute $\text{ext}_{\mathcal{G}}(\rho_0)$ using calls to a (σ, ρ) -#DOMSET^P-oracle. As noted previously, the terms $a(x)$ are polynomials in x . Therefore, it is straightforward to verify that $P(x) := A'_x / \text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*} = \pm a'(x)$ is also a polynomial in x (whether ρ is finite or cofinite). By Claim 10.8.13.2 we can compute A'_x and thus, also the value $\pm a'(x)$. As a last step, we compute $\text{ext}_{\mathcal{G}}(\rho_0)$ using standard interpolation.

Claim 10.8.13.3. *Given the coefficients of $P(x)$, we can compute the value $\text{ext}_{\mathcal{G}}(\rho_0)$.*

Proof of Claim. We are interested in the highest degree monomial of $P(x)$. By the definition of $a'(x)$ from Equation (10.8.17), we have

$$P(x) = \pm \text{ext}_{\mathcal{J}}(\rho_0)^2 \cdot a(x+2) \mp (\alpha + \beta) \text{ext}_{\mathcal{J}}(\rho_0) \cdot a(x+1) \pm a(x). \quad (10.8.18)$$

Recall that $a_r(x)$ is a polynomial of degree r . By the definition of $a(x)$ in Equation (10.8.9), we observe that the highest degree monomial of $a(x)$ is $\text{ext}_{\mathcal{G}}(\rho_0) \cdot \text{ext}_{\mathcal{J}}(\rho_1)^{r^*} \cdot x^{r^*}$, where r^* has a different meaning depending on whether ρ is finite or cofinite.

Therefore, by Equation (10.8.18), the highest degree monomial of $p(x)$ is determined by the highest degree monomial of

$$\text{ext}_{\mathcal{G}}(\rho_0) \cdot \text{ext}_{\mathcal{J}}(\rho_1)^{r^*} \cdot \left(\text{ext}_{\mathcal{J}}(\rho_0)^2 \cdot (x+2)^{r^*} - (\alpha + \beta) \cdot \text{ext}_{\mathcal{J}}(\rho_0) (x+1)^{r^*} + \alpha \cdot \beta \cdot x^{r^*} \right),$$

which is

$$\text{ext}_{\mathcal{G}}(\rho_0) \cdot \text{ext}_{\mathcal{J}}(\rho_1)^{r^*} \cdot \left(\text{ext}_{\mathcal{J}}(\rho_0)^2 - (\alpha + \beta) \cdot \text{ext}_{\mathcal{J}}(\rho_0) + \alpha \cdot \beta \right) \cdot x^{r^*}.$$

Hence, the coefficient of x^{r^*} in $P(x)$ is $c = \text{ext}_{\mathcal{G}}(\rho_0) \cdot \text{ext}_{\mathcal{J}}(\rho_1)^{r^*} \cdot c'$ where

$$c' := (\text{ext}_{\mathcal{J}}(\rho_0) - \alpha) \cdot (\text{ext}_{\mathcal{J}}(\rho_0) - \beta).$$

Recall that $\mathcal{J}$ is a strong candidate, and therefore, we have

- $\text{ext}_{\mathcal{J}}(\rho_1) \geq 1$, and consequently
- $\text{ext}_{\mathcal{J}}(\rho_0) \neq \text{ext}_{\mathcal{J}}(\rho_0) + \text{ext}_{\mathcal{J}}(\rho_1) = \alpha$, and also
- $\text{ext}_{\mathcal{J}}(\rho_0) \neq \text{ext}_{\mathcal{J}}(\sigma_0) + \text{ext}_{\mathcal{J}}(\sigma_1) = \beta$ by Property (VI).

This proves that $c' \neq 0$ and $\text{ext}_{\mathcal{J}}(\rho_1)^{r^*} \neq 0$. Consequently, once we recovered the coefficient c , we can compute the value $\text{ext}_{\mathcal{G}}(\rho_0)$ as well, which concludes the proof. $\blacktriangleleft$

Finally, we evaluate $P(x)$ for $r^* + 1$ distinct and sufficiently large values of $x \in \mathcal{O}(n)$ based on the computation of A'_x according to Claim 10.8.13.2 with three (σ, ρ) -#DOMSET $^{\mathcal{P}}$ -oracle calls. Then, we use polynomial interpolation to recover the coefficients of $P(x)$ and, by Claim 10.8.13.3, we efficiently compute $\text{ext}_{\mathcal{J}}(\rho_0)^{x-r^*}$ as $\text{ext}_{\mathcal{J}}(\rho_0)$ does not depend on n .

So, computing the coefficients can be done in time polynomial in $x \in \mathcal{O}(n)$, and from the coefficients, one can compute $\text{ext}_{\mathcal{G}}(\rho_0)$ according to Claim 10.8.13.3.

In summary, we can solve (σ, ρ) -#DOMSET $^{\mathcal{P}+(\emptyset, \{0\})}$ on instance I with at most $3(r^* + 1)$ (that is, three per A'_x) oracle calls to (σ, ρ) -#DOMSET $^{\mathcal{P}}$ on instances of the form I_x . This concludes the proof of Lemma 10.8.13. $\square$

10.8.3 The Case When σ Is Finite and $\rho = \mathbb{N}$

The goal of this section is to prove Lemma 10.8.17, which formally states a reduction from (σ, ρ) -#DOMSET $^{\mathcal{P}+(\emptyset, \mathbb{N}_{\geq 1})}$ to (σ, ρ) -#DOMSET $^{\mathcal{P}}$ if ρ does not enforce any constraints, that is, when $\rho = \mathbb{N}$, but the set σ is finite. We give this reduction in three steps.

- We start with the reduction from (σ, ρ) -#DOMSET $^{\mathcal{P}+(\emptyset, \mathbb{N}_{\geq 1})}$ to (σ, ρ) -#DOMSET $^{\mathcal{P}+(\{0\}, \mathbb{N})}$ in Lemma 10.8.14.
- Then, we reduce to (σ, ρ) -#DOMSET $^{\mathcal{P}+(\sigma, \emptyset)}$ in Lemma 10.8.15.
- Finally, we give the reduction to (σ, ρ) -#DOMSET $^{\mathcal{P}}$ in Lemma 10.8.16.

As a last result, we combine Lemmas 10.8.14 to 10.8.16 to prove Lemma 10.8.17.

Step 1: From $(\emptyset, \mathbb{N}_{\geq 1})$ to $(\{0\}, \mathbb{N})$. We start with the first step of this chain.

Lemma 10.8.14. Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . If $\rho = \mathbb{N}$, then (σ, ρ) -#DOMSET $^{\mathcal{P}+(\emptyset, \mathbb{N}_{\geq 1})} \leq_{\text{pw}} (\sigma, \rho)$ -#DOMSET $^{\mathcal{P}+(\{0\}, \mathbb{N})}$.

Proof. Let $I = (V, E, \lambda)$ be an instance of (σ, ρ) -#DOMSET $^{\mathcal{P}+(\emptyset, \mathbb{N}_{\geq 1})}$ and denote by U the $(\emptyset, \mathbb{N}_{\geq 1})$ -vertices. We first observe that we can assume that these vertices form an independent set as they must be unselected in any solution. Furthermore, we write $\mathcal{Z} = \{N(u) \mid u \in U\}$.

We write $I' = (V, E, \lambda')$ for the instance of (σ, ρ) -#DOMSET $^{\mathcal{P}+(\{0\}, \mathbb{N})}$ that is obtained from I where we make all vertices in U (σ, ρ) -vertices. Then, the number of solutions of I is identical to the number of those solutions of I' that select at least one vertex from each $Z \in \mathcal{Z}$.

For a positive integer x , we modify the instance I' and attach, for each set $Z \in \mathcal{Z}$, a total of x $(\{0\}, \mathbb{N})$ -vertices $v_1^Z, \dots, v_x^Z$, each of which is completely connected to Z . We denote the resulting instance of (σ, ρ) -#DOMSET $^{\mathcal{P}+(\{0\}, \mathbb{N})}$ by I'_x .

Let S be some solution for I' . In order for S to be extended to a solution of I'_x , we need that all vertices in V that are *not* in some $Z \in \mathcal{Z}$ already have a feasible number of selected neighbors, that is, their σ -constraints and ρ -constraints are satisfied. We say that subsets S with this property are *good*.

Suppose that in some good set S , a set $Z \in \mathcal{Z}$ is entirely unselected. Then, there are 2^x feasible extensions to the x attached vertices $v_1^Z, \dots, v_x^Z$; each of them can be selected or not. For this, we crucially exploit that $\rho = \mathbb{N}$. On the contrary, if at least one vertex of Z is selected, then all vertices $v_1^Z, \dots, v_x^Z$ have to be unselected in every feasible extension of S to a solution of I'_x because the vertices $v_\star^Z$ are $(\{0\}, \mathbb{N})$ -vertices.

Let a_i denote the number of good solutions for I' in which precisely i of the sets in $\mathcal{Z}$ are entirely unselected. We observe that a_0 is the number of good selections S for I' such that no set in $\mathcal{Z}$ is entirely unselected, that is, each set $Z \in \mathcal{Z}$ contains at least one vertex from S . Hence, all attached vertices $v_\star^Z$ have to be unselected, which implies that all vertices in Z obtain a feasible number of selected neighbors already from S . Therefore, the sets S that contribute to a_0 are precisely the (σ, ρ) -sets of I with the additional property that each set Z in $\mathcal{Z}$ contains at least one selected vertex. This shows that a_0 is precisely the number of solutions of I .

When we write $\#Sol(I'_x)$ for the number of solutions of I'_x , we have

$$\#Sol(I'_x) = \sum_{i=0}^{|\mathcal{Z}|} a_i \cdot 2^{x \cdot i}.$$

Assume that instance I contains n vertices. Then, for each i , we have $a_i \leq 2^n$. Hence, when we set $x := n + 1$, we have $a_0 < 2^x$. Thus, we can recover a_0 by a *single* (σ, ρ) -#DOMSET $^{\mathcal{P}+(\{0\}, \mathbb{N})}$ -oracle call as $a_0 = \#Sol(I'_x) \bmod 2^x$.

Observe that the size of the new instance might increase quadratically, as, for each $(\emptyset, \mathbb{N}_{\geq 1})$ -vertex (and there might be $\mathcal{O}(n)$ of them), we add $\mathcal{O}(n)$ vertices to the instance.

It remains to argue that the pathwidth does not change too much. For each set Z , there is a bag β containing Z . We duplicate this bag β a total of $x \in \mathcal{O}(n)$ times, and each vertex v_i^Z is added to exactly one of these copies. Hence, the pathwidth of I'_x is at most that of I plus an additive constant. $\square$

Step 2: From $(\{0\}, \mathbb{N})$ to $(\sigma, \emptyset)$. For the second step in this chain of reductions, we require that σ is finite. We add a simple gadget to the $(\{0\}, \mathbb{N})$ -vertices, which we want to remove, to transform the output of Lemma 10.8.14 into an instance of (σ, ρ) -#DOMSET $^{\mathcal{P}+(\sigma, \emptyset)}$, that is, there are vertices which are forced to be selected.

Lemma 10.8.15. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . If $\rho = \mathbb{N}$ and σ is finite, then (σ, ρ) -#DOMSET $^{\mathcal{P}+(\{0\}, \mathbb{N})} \leq_{\text{pw}} (\sigma, \rho)$ -#DOMSET $^{\mathcal{P}+(\sigma, \emptyset)}$.*

Proof. Observe that the case $\sigma = \{0\}$ is trivial, as then $(\{0\}, \mathbb{N})$ -vertices are precisely (σ, ρ) -vertices. Moreover, we do not use $(\sigma, \emptyset)$ -vertices.

Note that s_{top} is also used for the ρ -state.

Hence, we assume that $\sigma \neq \{0\}$ which especially implies that $s_{\text{top}} > 0$. We first construct a parsimonious $\{\sigma_{s_{\text{top}}}, \rho_{s_{\text{top}}}\}$ -provider $\mathcal{J}$ for (σ, ρ) -#DOMSET $^{\mathcal{P}+(\sigma, \emptyset)}$. We refer to Definitions 8.1.10 and 10.6.2 for the formal definition.

Claim 10.8.15.1. *There is a parsimonious $\{\sigma_{s_{\text{top}}}, \rho_{s_{\text{top}}}\}$ -provider $\mathcal{J} = (J, \{u\})$ for (σ, ρ) -#DOMSET $^{\mathcal{P}+(\sigma, \emptyset)}$.*

Proof of Claim. We distinguish two cases depending on whether $s_{\text{top}} - 1 \in \sigma$ or not.

$s_{\text{top}} - 1 \notin \sigma$: The graph J consists of a clique C with s_{top} $(\sigma, \emptyset)$ -vertices. Additionally, we create two (σ, ρ) -vertices u and v and make both adjacent to all vertices of the clique C . We choose vertex u as the portal of the provider.

Note that there are precisely two partial solutions for this gadget. The vertices of C are $(\sigma, \emptyset)$ -vertices, and therefore, they are always selected. If the portal u is selected, then the vertices in C cannot have another selected neighbor as σ is finite, and therefore, vertex v must be unselected which is fine as $\rho = \mathbb{N}$. If the portal u is unselected, then the vertices in C have $s_{\text{top}} - 1$ selected neighbors within C . Since $s_{\text{top}} - 1 \notin \sigma$, the vertex v must be selected which is fine as v now also has s_{top} selected neighbors. In either case, portal u has s_{top} selected neighbors within J .

$s_{\text{top}} - 1 \in \sigma$: We first construct a parsimonious $\{\sigma_1, \rho_1\}$ -provider. We start with a clique C with s_{top} $(\sigma, \emptyset)$ -vertices. We add one (σ, ρ) -vertex u , make it adjacent to precisely one vertex of the clique, and define u as the portal. We denote the resulting gadget by $\mathcal{J}' = (J', \{u'\})$.

Note that all vertices in C have degree $s_{\text{top}} - 1$ in C . Hence, selecting all vertices of C gives a partial solution independently of the selection status of u as $s_{\text{top}} - 1, s_{\text{top}} \in \sigma$ and $\rho = \mathbb{N}$. In either case, the portal u obtains one selected neighbor from J' and thus, $\mathcal{J}'$ is a parsimonious $\{\sigma_1, \rho_1\}$ -provider.

Now, we create s_{top} copies of $\mathcal{J}'$ and identify all copies of the portal u' with the portal u of the resulting gadget $\mathcal{J} = (J, \{u\})$. As the provider $\mathcal{J}'$ is parsimonious, there are precisely two solutions and in each the portal u receives s_{top} selected neighbors. $\triangleleft$

With the parsimonious $\{\sigma_{s_{\text{top}}}, \rho_{s_{\text{top}}}\}$ -realizer $\mathcal{J}$ from Claim 10.8.15.1, it is straightforward to define the reduction from (σ, ρ) -#DOMSET $^{\mathcal{P}+(\{0\}, \mathbb{N})}$ to (σ, ρ) -#DOMSET $^{\mathcal{P}+(\sigma, \emptyset)}$.

Consider an instance $I = (G, \lambda)$ of (σ, ρ) -#DOMSET $^{\mathcal{P}+(\{0\}, \mathbb{N})}$, and denote by U the set of $(\{0\}, \mathbb{N})$ -vertices of I . We define an instance I' of (σ, ρ) -#DOMSET $^{\mathcal{P}+(\sigma, \emptyset)}$ based on I where we transform each vertex u in U into a (σ, ρ) -vertex, and make u the portal of a new copy of the parsimonious $\{\sigma_{s_{\text{top}}}, \rho_{s_{\text{top}}}\}$ -realizer $\mathcal{J}$.

In each solution, vertex u obtains s_{top} selected neighbors from its copy of $\mathcal{J}$. Hence, we can treat vertex u as a $(\sigma - s_{\text{top}}, \rho - s_{\text{top}})$ -vertex, which is precisely a $(\{0\}, \mathbb{N})$ -vertex as σ is finite and $\rho = \mathbb{N}$. Thus, the number of solutions of instance I for (σ, ρ) -#DOMSET $^{\mathcal{P}+(\{0\}, \mathbb{N})}$ is equal to the number of solutions of I' for (σ, ρ) -#DOMSET $^{\mathcal{P}+(\sigma, \emptyset)}$.

It remains to bound the pathwidth of the new instance. For each vertex u in U , there is a bag β in the path decomposition of G that contains u . We create a copy of this bag for the path decomposition of I' , and add the vertices of the copy of $\mathcal{J}$ that is attached to u to this copy of the bag. As we can create a separate copy of the original bag for each vertex in U , the pathwidth increases by at most the size of $\mathcal{J}$, which depends only on σ and ρ . Hence, the pathwidth increases at most by a constant. $\square$

Step 3: Removing $(\sigma, \emptyset)$. It remains to remove the $(\sigma, \emptyset)$ -vertices which are introduced by the previous reduction.

Lemma 10.8.16. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . If $\rho = \mathbb{N}$ and σ is finite, then $(\sigma, \rho)\text{-}\#DOMSET^{(\sigma, \emptyset)} \leq_{pw} (\sigma, \rho)\text{-}\#DOMSET$.*

Proof. We first construct and analyze an auxiliary gadget which we use in the reduction. First assume that $s_{\min} \neq 1$. Let $\mathcal{J}^{(x)} = (J^{(x)}, \{p\})$ be a gadget for $(\sigma, \rho)\text{-}\#DOMSET$ that consists of p together with x cliques with $s_{\min} + 1$ vertices. The portal p is adjacent to precisely one vertex from each clique.

We analyze the number of extensions for this gadget assuming the portal p already has i neighbors from the outside. If the portal p is not selected, then each of the cliques in $\mathcal{J}^{(x)}$ can be only entirely selected or entirely unselected as selecting only a few vertices would give the selected vertices less than $s_{\min}$ selected neighbors. As p is unselected, each combination of entirely selected or unselected cliques is feasible since $\rho = \mathbb{N}$. This implies that the gadget $\mathcal{J}^{(x)}$ has 2^x feasible extensions if vertex p is not selected.

If the portal p is selected, the situation is more complicated unless $s_{\min} = 0$. If at least two vertices of the clique are unselected, then no vertex can have $s_{\min}$ selected neighbors, and there is no solution. If precisely one vertex is unselected, say q_0 , and all other vertices are selected, then the neighbor of the portal p in the clique, say q_1 , has $s_{\min}$ neighbors. However, some other selected vertex of the clique, that is, a vertex different from q_0 and q_1 , which exists as $s_{\min} + 1 \geq 3$, has less than s_{top} neighbors and there is no solution as well. Thus, each clique must be entirely selected or unselected.

We denote by $f_i(x)$ the number of feasible extensions to $\mathcal{J}^{(x)}$, assuming that p already has i neighbors. Formally, we have

$$f_i(x) := \begin{cases} \sum_{\substack{s \in [0 \dots x]: \\ i+s \in \sigma}} \binom{x}{s} & \text{if } s_{\min} + 1 \in \sigma, \\ 1 & \text{if } s_{\min} + 1 \notin \sigma \text{ and } i \in \sigma, \\ 0 & \text{if } s_{\min} + 1 \notin \sigma \text{ and } i \notin \sigma. \end{cases}$$

Since σ is finite, $f_i(x)$ is a polynomial of degree at most s_{top} . Also, observe that the constant of $f_i(x)$ is 1 if $i \in \sigma$, and 0 otherwise.

Before we proceed with the proof, we analyze the gadget for the case when $s_{\min} = 1$. If the portal p is unselected, then the situation remains unchanged. If the portal p is selected, then we can partially select the cliques, that is, the vertex of degree 1 might not be selected as the

neighbor of p has portal p as selected neighbor. With the interpretation of $f_i(x)$ as above, we have

$$f_i(x) := \begin{cases} \sum_{\substack{s \in [0 \dots x] \\ i+s \in \sigma}} \binom{x}{s} \cdot 2^s & \text{if } 2 \in \sigma, \\ \sum_{\substack{s \in [0 \dots x] \\ i+s \in \sigma}} \binom{x}{s} & \text{if } 2 \notin \sigma. \end{cases}$$

For the case when $2 \in \sigma$, we observe that the term 2^s can be bounded by $2^{s_{\text{top}}}$ as σ is finite. Hence, as for the general case, $f_i(x)$ is a polynomial in x of degree at most s_{top} , and the constant of $f_i(x)$ is 1 if $i \in \sigma$ and 0 otherwise.

Now, consider an instance $I = (G, \lambda)$ of (σ, ρ) -#DOMSET $^{\mathcal{P}+(\sigma, \emptyset)}$, and denote by U the set of $(\sigma, \emptyset)$ -vertices of I . For a positive integer x , we define an instance I_x of (σ, ρ) -#DOMSET $^{\mathcal{P}}$ where, for each vertex $u \in U$, we make the vertex u a (σ, ρ) -vertex, create a copy of $\mathcal{J}^{(x)}$, and identify the respective copy of the portal vertex with u .

We first argue why this modification does not change the pathwidth too much. Recall that each vertex u in U is adjacent to one vertex of x $s_{\min} + 1$ -cliques. Since each clique is of constant size, we can duplicate an arbitrary bag that contains u x times, and add, to each copy of that bag, the vertices of precisely one clique. As a consequence, the pathwidth increases by at most a constant.

Let $\#Sol(I_x)$ be the number of solutions of (σ, ρ) -#DOMSET for the instance I_x . We define $p(x) := \#Sol(I_x) \bmod 2^x$ and claim that $p(x)$ is a polynomial whose degree depends on σ .

Claim 10.8.16.1. *It holds that $p(x)$ is a polynomial in x with maximum degree $s_{\text{top}} \cdot |U|$.*

Proof of Claim. Observe that $\#Sol(I_x)$ is a sum over all possible solutions for I multiplied by the number of corresponding extensions to the copies of $\mathcal{J}^{(x)}$.

Consider a solution S of I_x . If there is one vertex in $U \setminus S$, that is, a former $(\sigma, \emptyset)$ -vertex that is not selected, then there are 2^x extensions to the attached copy of $\mathcal{J}^{(x)}$. Hence, this solution does not contribute to $p(x)$.

In contrast, if all vertices from U are selected, that is, if $U \subseteq S$, then, for each vertex $u \in U$, and for the number i of selected neighbors of u in $S \cap V(G)$, there are $f_i(x)$ extensions to the copy of $\mathcal{J}^{(x)}$ that is attached to u . By the above argument, $f_i(x)$ is a polynomial in x with maximum degree s_{top} . Hence, $p(x)$ has degree at most $|U| \cdot s_{\text{top}}$. $\triangleleft$

As the next step, we prove that the constant term of $p(x)$ is the number of those solutions of I where the vertices in U are selected, which is precisely what we want.

Claim 10.8.16.2. *The constant term of $p(x)$ is equal to the number of solutions of I , that is, $p(0) = \#Sol(I)$.*

Proof of Claim. Recall that $f_i(x)$ is a polynomial of degree at most s_{top} , and that the constant of $f_i(x)$ is 1 if $i \in \sigma$, and 0 otherwise.

By the arguments from Claim 10.8.16.1, it suffices to analyze the solutions of I_x for which all vertices in U are selected. Thus, consider a selection $S_G \supseteq U$ of vertices from $V(G)$. This selection S_G contributes 1 to the constant term of $p(x)$ if and only if, for each vertex in U , the

corresponding number i of selected neighbors in $S_G \setminus U$ is in σ , and otherwise it contributes 0 to the constant. So, the set S_G contributes 1 if and only if S_G is a solution of I in which all vertices of U are selected.

We conclude that the constant of $p(x)$ is equal to the number of solutions of (σ, ρ) -#DOMSET ^{$\mathcal{P}+(\sigma, \emptyset)$} on instance I ; which is precisely what we want to compute. $\triangleleft$

As the degree of $p(x)$ in x is bounded by $s_{\text{top}} \cdot |U|$ by Claim 10.8.16.1, we can recover $p(x)$ by polynomial interpolation using at most $s_{\text{top}} \cdot |U| + 1$ different sufficiently large values of x and the corresponding oracle calls to (σ, ρ) -#DOMSET on instance I_x . Then, by Claim 10.8.16.2, the constant term of $p(x)$ gives the number of solutions of I . $\square$

Now, we have everything ready to prove the main result of this section which we restate here for convenience.

Lemma 10.8.17. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . If $\rho = \mathbb{N}$ and σ is finite, then (σ, ρ) -#DOMSET ^{$\mathcal{P}+(\emptyset, \mathbb{N}_{\geq 1})$} $\leq_{\text{pw}}$ (σ, ρ) -#DOMSET ^{$\mathcal{P}$} .*

Proof. The proof follows from the direct combination of Lemmas 10.8.14 and 10.8.16 with Lemma 10.8.15 as the combined reduction is pathwidth-preserving by the transitivity of pathwidth-preserving reductions from Observation 2.2.5. $\square$

10.9 Special Case: σ Is Cofinite and $\rho = \mathbb{N}$

In the following, we provide the missing realization of relations for the counting version when σ is cofinite and $\rho = \mathbb{N}$. In contrast to the previous results, we start from the variant of (σ, ρ) -#DOMSET where we allow arbitrary relations, that is, we start from (σ, ρ) -#DOMSET^{Rel}. Formally, we prove Main Lemma 10.9.1 in the following.

Main Lemma 10.9.1. *Fix two non-empty set $\sigma, \rho \subseteq \mathbb{N}$ that are non-trivial where σ is cofinite and $\rho = \mathbb{N}$. For all constants d , there is a pathwidth-preserving reduction from (σ, ρ) -#DOMSET^{Rel} on instances with arity at most d to (σ, ρ) -#DOMSET.*

Before we move to the formal proofs, we provide some intuition why this case is so special and needs to be considered separately.

Intuition for the Difficulty. Suppose that $\sigma = \mathbb{N}_{\geq 1}$ as this turns out to be the key case. Think of some $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertex v , that is, a normal (σ, ρ) -vertex, which is subject to some relation R . To model this relation, we need to distinguish between those solutions in which vertex v is selected and those in which it is not selected. That is, we need to design a gadget that has a different number of extensions depending on the selection state of v . Since we have $\rho = \mathbb{N}$, every *unselected* neighbor of v in a hypothetical realization behaves the same independently of the selection status of v . Thus, if v has only unselected neighbors, then they are not affected by the selection status of v . Therefore, we can only distinguish whether v is selected from v not being selected if v has at least one *selected* neighbor.

Now assume that v has at least one selected neighbor. Then, vertex v has a feasible number of selected neighbors as $\sigma = \mathbb{N}_{\geq 1}$ and $\rho = \mathbb{N}$. Hence, the state of v is independent of its

While we could also start from (σ, ρ) -#DOMSET^{HW=1}, the following reductions introduce arbitrary relations and thus, we do not need the results from Section 10.6.

selection status or how many additional selected neighbors it might obtain outside the gadget. Thus, vertex v is entirely unaffected by the selection status of its remaining neighbors and acts essentially as a $(\mathbb{N}, \mathbb{N})$ -vertex.

Vertex v now behaves the same no matter what the selection state is and how many selected neighbors it has. Thus, it is hard to imagine how we could recover the situation where v behaves like a $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertex while having a handle on its selection status. Intuitively we have two opposing goals. (1) We want to observe the selection status of v in the sense that we want to model the relation R by attaching some gadgets and using interpolation techniques. (2) Simultaneously we want to recover the situation where v behaves like a $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertex with respect to the original graph.

To handle this situation, we take a different view on the concept of “ (σ, ρ) -sets with relations”. In particular, we diverge from the previous notion of (σ, ρ) -sets with relations as defined in Definition 10.1.1. More precisely, we treat all vertices that are subject to some relation now as $(\mathbb{N}, \mathbb{N})$ -vertices, that is, they do not impose any constraints on the solutions other than that they have to satisfy the relations.

Generally speaking, the concept with $(\mathbb{N}, \mathbb{N})$ -vertices seems particularly suited for the setting where both σ and ρ are cofinite. Indeed, once some gadget provides $t_{\text{top}} = \max\{s_{\text{top}}, r_{\text{top}}\}$ selected neighbors to a vertex, this vertex “is always satisfied” in the sense that it acts as a $(\mathbb{N}, \mathbb{N})$ -vertex with respect to the rest of the graph outside the gadget. However, in the context of finite σ or ρ , it is much harder to imagine how to model the concept of being “always satisfied”, so it seems less useful there and the notion from Definition 10.1.1 is suited much better.

Before we state the intermediate results for the proof of Main Lemma 10.9.1, we first formalize the adjusted definition of relations.

Notation and Problem Definitions. Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $G = (V, E, \mathcal{C})$ be a graph with relations as defined in Definition 10.1.1. To define the new concept of (σ, ρ) -sets with relations, let U be the set of vertices that are in the scope of at least one relation.

We say that a selection $S \subseteq V(G)$ is a $(\sigma, \rho)^{\ddagger}$ -set of G if

- for each $v \in S \setminus U$, it holds that $|N(v) \cap S| \in \sigma$,
- for each $v \in V(G) \setminus (S \cup U)$, it holds that $|N(v) \cap S| \in \rho$, and
- for each relational constraint $C \in \mathcal{C}$, it holds that $S \cap \text{scp}(C) \in R(C)$.

Intuitively, the vertices in $V(G) \setminus U$ are (σ, ρ) -vertices, but the vertices in U act as $(\mathbb{N}, \mathbb{N})$ -vertices. Hence, they are not required to fulfill any σ -constraints or ρ -constraints. These constraints are always satisfied as any number of selected neighbors is feasible.

Similar to Section 10.8, we work with a generalization of $(\sigma, \rho)^{\ddagger}$ -sets to the setting where we can use a set of additional pairs $\mathcal{P} = \{(\sigma^{(i)}, \rho^{(i)}) \in \Omega_0^2 \mid i \in [1 \dots k]\}$ for some non-negative integer k . As before, we set $(\sigma^{(0)}, \rho^{(0)}) := (\sigma, \rho)$ for ease of notation. Then, given a mapping $\lambda: V(G) \rightarrow [0 \dots k]$, we say that a subset $S \subseteq V(G)$ is a $\lambda^{\ddagger}$ -set of G if

- for each vertex $v \in S \setminus U$, it holds that $|N(v) \cap S| \in \sigma^{(\lambda(v))}$,

- for each vertex $v \in V(G) \setminus (S \cup U)$, it holds that $|N(v) \cap S| \in \rho^{(\lambda(v))}$, and
- for each relational constraint $C \in \mathcal{C}$, it holds that $S \cap \text{scp}(C) \in R(C)$.

Then, $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}, \text{Rel}^\ddagger}$ is the problem that takes as input some graph with relations G and a function $\lambda: V(G) \rightarrow [0..k]$, and asks for the number of $\lambda^\ddagger$ -sets of G . As before, if $\mathcal{P}$ is empty, then we drop the superscript $\mathcal{P}$ and the function λ as part of the input to obtain the problem $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{Rel}^\ddagger}$ that takes as input a graph with relations G and asks for the number of $(\sigma, \rho)^\ddagger$ -sets of G .

With these definitions, we can move to the formal proof.

Outline of the Proof. As mentioned, the realization of the relations deviates from the general case in Section 10.8. However, the basic outline is still similar to the approach in the general case. We refer to Figure 10.19 for a visualization of the chain of reductions.

As a first result, we transition to the new notion of $(\sigma, \rho)^\ddagger$ -sets. Instead of providing a direct reduction, we use intermediate steps based on weighted relations and weighted vertices similar to the approach in Section 10.5. Formally, we summarize the first step by the reduction in Lemma 10.9.5. We defer the proof to Section 10.9.1.

Lemma 10.9.5. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$ where σ is cofinite and $\rho = \mathbb{N}$. For all constants d , there is a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{Rel}}$ on instances with arity at most d to $(\sigma, \rho)\text{-}\#\text{DOMSET}^{(\mathbb{N}_{\geq 1}, \mathbb{N}), \text{Rel}^\ddagger}$ with arity at most d .*

As a second step, we show that we can replace arbitrary relations by $\text{HW}_{=1}$ -relations even in this new context. Although this result uses the same approach as for the general case in Section 10.6, we provide a new proof because of the different notion of relations. Formally, we prove the reduction from Lemma 10.9.6 in Section 10.9.2.

Lemma 10.9.6. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$ where σ is cofinite and $\rho = \mathbb{N}$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . For all constants d , there is a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}, \text{Rel}^\ddagger}$ with arity at most d to $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}, \text{HW}_{=1}^\ddagger}$.*

Technically, an instance of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{HW}_{=1}^\ddagger}$ contains only $\text{HW}_{=1}$ -relations. However, as stated in Lemma 10.9.6, we frequently refer to them as $\text{HW}_{=1}^\ddagger$ -relations to highlight that these relations are interpreted differently in the context of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{Rel}^\ddagger}$.

As the last step, we prove that we can realize $\text{HW}_{=1}$ -relations. We use ideas from the general case in Section 10.8 for this. Formally, we prove the reduction from Lemma 10.9.11 in Section 10.9.3.

Lemma 10.9.11. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$ where σ is cofinite and $\rho = \mathbb{N}$. For all constants d , there is a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#\text{DOMSET}^{(\mathbb{N}_{\geq 1}, \mathbb{N}), \text{HW}_{=1}^\ddagger}$ with arity at most d to $(\sigma, \rho)\text{-}\#\text{DOMSET}$.*

Now, we have everything ready, and can combine Lemma 10.9.5 with Lemmas 10.9.6 and 10.9.11 to obtain the reduction from $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\text{Rel}}$ to $(\sigma, \rho)\text{-}\#\text{DOMSET}$ when $\rho = \mathbb{N}$ and σ is cofinite.

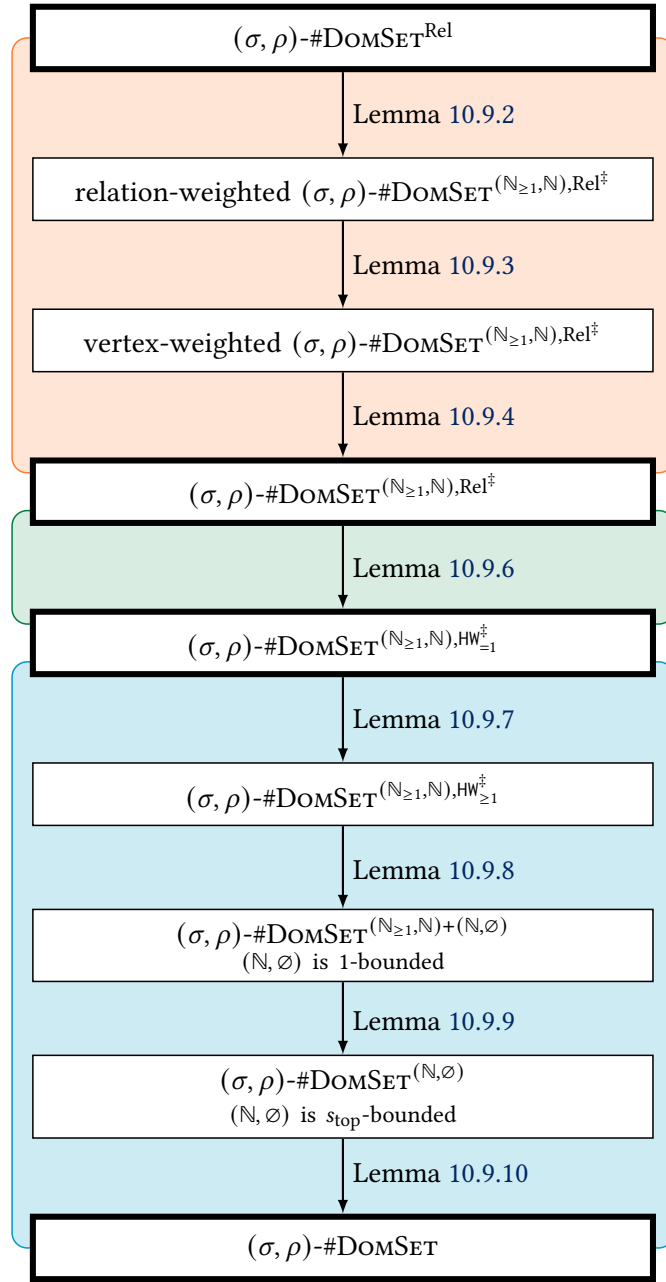


Figure 10.19: An overview of the reductions in Section 10.9 to realize $\text{HW}_{=1}$ for $(\sigma, \rho)\text{-}\#\text{DOMSET}$ when $\rho = \mathbb{N}$ and σ is cofinite, that is, for the proof of Main Lemma 10.9.1.

The reductions in the green box on the top are given in Section 10.9.1 and prove Lemma 10.9.5. The reduction in the orange box in the middle is given in Section 10.9.2 and proves Lemma 10.9.6. The reductions in blue box on the bottom are given in Section 10.9.3 and prove Lemma 10.9.11.

Main Lemma 10.9.1. *Fix two non-empty sets $\sigma, \rho \subseteq \mathbb{N}$ that are non-trivial where σ is cofinite and $\rho = \mathbb{N}$. For all constants d , there is a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#DOMSET^{Rel}$ on instances with arity at most d to $(\sigma, \rho)\text{-}\#DOMSET$.*

Proof. The proof chains the reductions from Lemmas 10.9.5, 10.9.6 and 10.9.11.

By Lemma 10.9.5 we obtain a pa-reduction from $(\sigma, \rho)\text{-}\#DOMSET^{Rel}$ on instances of arity at most d to $(\sigma, \rho)\text{-}\#DOMSET^{(\mathbb{N}_{\geq 1}, \mathbb{N}), Rel^{\ddagger}}$. Since these reductions are arity-preserving, it suffices to solve $(\sigma, \rho)\text{-}\#DOMSET^{(\mathbb{N}_{\geq 1}, \mathbb{N}), Rel^{\ddagger}}$ on instances of arity at most d' where d' is d plus some constant.

From Lemma 10.9.6 with $\mathcal{P} = \{(\mathbb{N}_{\geq 1}, \mathbb{N})\}$ we obtain a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#DOMSET^{(\mathbb{N}_{\geq 1}, \mathbb{N}), Rel^{\ddagger}}$ on instances of arity at most d' to $(\sigma, \rho)\text{-}\#DOMSET^{(\mathbb{N}_{\geq 1}, \mathbb{N}), HW_{=1}^{\ddagger}}$.

By Lemma 10.9.11, again with $\mathcal{P} = \{(\mathbb{N}_{\geq 1}, \mathbb{N})\}$, we obtain a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#DOMSET^{(\mathbb{N}_{\geq 1}, \mathbb{N}), HW_{=1}^{\ddagger}}$ to $(\sigma, \rho)\text{-}\#DOMSET$, which finishes the proof. $\square$

Organization. In the following, we first prove Lemma 10.9.5 in Section 10.9.1. Then, we prove Lemma 10.9.6 in Section 10.9.2. Finally, we prove Lemma 10.9.11 in Section 10.9.3.

10.9.1 Adjusting to the New Definition of Relations

As a first intermediate result, we establish the new interpretation of the relations based on additional $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertices. Formally, we prove Lemma 10.9.5.

Lemma 10.9.5. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$ where σ is cofinite and $\rho = \mathbb{N}$. For all constants d , there is a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#DOMSET^{Rel}$ on instances with arity at most d to $(\sigma, \rho)\text{-}\#DOMSET^{(\mathbb{N}_{\geq 1}, \mathbb{N}), Rel^{\ddagger}}$ with arity at most d .*

We split the proof into three steps that involve weighted versions of the problem at hand. We postpone the formal definition and provide the outline first.

1. First, we reduce from $(\sigma, \rho)\text{-}\#DOMSET^{Rel}$ to a relation-weighted version of $(\sigma, \rho)\text{-}\#DOMSET^{(\mathbb{N}_{\geq 1}, \mathbb{N}), Rel^{\ddagger}}$. This is made formal in Lemma 10.9.2 and exploits that we are dealing with the counting version of the problem.
2. We use weighted vertices to remove the weights from the relations. We formalize this result in Lemma 10.9.3.
3. We use interpolation techniques to remove the weighted vertices. We formalize this reduction in Lemma 10.9.4.

We remark that the second and third step follow analogously to the results from Section 10.5.

Before we start with the proofs, we define the weighted versions of the problem formally.

Weighted Versions. Analogously to the definitions in Section 10.5, for a non-negative integer q , we introduce the problem q -relation-weighted (σ, ρ) -#DOMSET $^{\mathcal{P}, \text{Rel}^\ddagger}$ that takes as input a graph with *weighted* relations $G = (V, E, \mathcal{C})$ such that there are at most q different weights, that is, $|\{f_C(U) \mid C \in \mathcal{C}, U \subseteq \text{scp}(C)\}| \leq q$. Each solution $S \subseteq V$ of (σ, ρ) -#DOMSET $^{\mathcal{P}, \text{Rel}^\ddagger}$ on input G obtains a weight $f(S) := \prod_{C \in \mathcal{C}} f_C(S \cap \text{scp}(C))$. The task is then to compute the sum of the values $f(S)$ summed over all $\lambda^\ddagger$ -sets S of G .

Similarly, for q -vertex-weighted (σ, ρ) -#DOMSET $^{\mathcal{P}, \text{Rel}^\ddagger}$, the input is a graph with relations G (that is, there are no relation weights) together with a weight function $\text{wt}: V(G) \rightarrow \mathbb{Q}$ such that $|\{\text{wt}(v) \mid v \in V\}| \leq q$, for some non-negative integer q . The task is to compute the sum of the values $g(S) := \prod_{v \in X} \text{wt}(v)$ summed over all $\lambda^\ddagger$ -sets S of G .

Now, we are ready to prove the first result of this section.

Lemma 10.9.2. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$ where σ is cofinite and $\rho = \mathbb{N}$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . There is a positive integer q depending only on σ such that there is a pa-reduction from (σ, ρ) -#DOMSET $^{\text{Rel}}$ to q -relation-weighted (σ, ρ) -#DOMSET $^{(\mathbb{N}_{\geq 1}, \mathbb{N}), \text{Rel}^\ddagger}$.*

Proof. Let $G = (V, E, \mathcal{C})$ denote an instance of (σ, ρ) -#DOMSET $^{\text{Rel}}$. Let $\mathcal{Z} = \{\text{scp}(C) \mid C \in \mathcal{C}\}$, and let U be the union of the sets in $\mathcal{Z}$. We could cast G as an instance of (σ, ρ) -#DOMSET $^{\text{Rel}^\ddagger}$. However, in solutions of (σ, ρ) -#DOMSET $^{\text{Rel}^\ddagger}$ on input G , all vertices in U behave like $(\mathbb{N}, \mathbb{N})$ -vertices, whereas in solutions of (σ, ρ) -#DOMSET $^{\text{Rel}}$ on input G , they behave like (σ, ρ) -vertices.

Idea. As a first step, consider the modified instance G' which is obtained from G by introducing, for each $Z \in \mathcal{Z}$ and each $z \in Z$, a vertex z_Z . Let $U' = \{z_Z \mid Z \in \mathcal{Z}, z \in Z\}$ and let $Z' = \{z_Z \mid z \in Z\}$. We know that, for each $Z \in \mathcal{Z}$, there is a relational constraint in G with scope Z . In G' , the scope of this constraint is Z' rather than Z and the corresponding relation uses $z_Z \in Z'$ whenever it previously used $z \in Z$. The vertices of G' are $V \cup U'$.

Observe that the solutions of (σ, ρ) -#DOMSET $^{\text{Rel}}$ on instance G are in a one-to-one correspondence with those solutions of (σ, ρ) -#DOMSET $^{\text{Rel}^\ddagger}$ on G' in which, for each $Z \in \mathcal{Z}$, a vertex $z \in Z$ is selected if and only if $z_Z \in Z'$ is selected. Intuitively, this means that the vertices of Z behave as (σ, ρ) -vertices, but also mirror the selection status of the vertices in Z' , which means that they satisfy the corresponding relation. We now introduce a gadget $\mathcal{J}$, which we use to ensure this property. See Figure 10.20 for an illustration.

Definition of an Auxiliary Gadget. The gadget $\mathcal{J} = (J, \{u, u'\}, \lambda_{\mathcal{J}})$ consists of a graph J with weighted relations and two portals u and u' . The vertex set of J contains u and u' , and additionally six new vertices a, b, c and a', b', c' . We connect the vertices as follows:

- portal u is adjacent to vertices c and b ,
- vertex c is adjacent to vertex a ,
- analogously, portal u' is adjacent to vertices c' and b' , and
- vertex c' is adjacent to vertex a' .

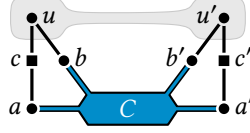


Figure 10.20: The construction of the gadget $\mathcal{J}$ with portals u and u' from Lemma 10.9.2. The vertices c and c' are $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertices and all other vertices are (σ, ρ) -vertices. The relation C is such that $R(C) = \{\emptyset, a, b, a', b', aa', ab', ba', bb'\}$.

Later, for each $Z \in \mathcal{Z}$ and $z \in Z$, we create a copy of the gadget $\mathcal{J}$ and identify u with z and u' with z_Z .

All of these vertices are (σ, ρ) -vertices, except the two vertices c and c' , which are $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertices.

Additionally, J contains a new weighted relation C with scope $\{a, b, a', b'\}$. It is more convenient to define this weighted relation as a relation $R \subseteq 2^{\{a, b, a', b'\}}$ together with a weight function $f_C: 2^{\{a, b, a', b'\}} \rightarrow \mathbb{Q}$ that takes non-zero values only on elements from R . By this notation, the corresponding relation R is the set of all subsets of $\{a, b, a', b'\}$ that contain at most one vertex of a or b , and at most one vertex of a' or b' . We show how to choose the weight function f_C in Claim 10.9.2.1.

We distinguish three different “states” of the vertex u with respect to $\mathcal{J}$.

- State σ_0 means that u is selected but has no selected neighbor in J .
- State $\sigma_{\geq 1}$ means that u is selected and has (at least) one selected neighbor in J .
- State ρ means that u is unselected with any number of selected neighbors in J .

We distinguish the same three states for the vertex u' .

We define $\mathbb{T} = \{\sigma_0, \sigma_{\geq 1}, \rho\}$ in the following. Then, given two states $\tau, \tau' \in \mathbb{T}$, we say that an extension of (τ, τ') to $\mathcal{J}$ is a feasible selection of vertices from J such that u is in state τ and u' is in state τ' . More precisely, a set $S \subseteq \{u, u', a, \dots, c'\}$ is a feasible extension of (τ, τ') if

- the vertices u and u' have states τ and τ' with regard to S , respectively,
- when $c \in S$, then c has at least one neighbor in S (it is a $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertex), and similarly for c' , and
- the set S contains at most one vertex of a and b , and at most one vertex of a' and b' , that is, the set S respects the relation R .

Let $\mathcal{S}(\tau, \tau')$ be the set of extensions of (τ, τ') to $\mathcal{J}$. Then, $\text{ext}_{\mathcal{J}}(\tau, \tau') := \sum_{S \in \mathcal{S}(\tau, \tau')} f_C(S \cap \text{scp}(C))$ is the sum of extensions of (τ, τ') to $\mathcal{J}$ weighted according to f_C . To conclude the construction of the gadget $\mathcal{J}$, we need to define the weights for the relation R , that is, we need to define f_C .

Claim 10.9.2.1. *We can choose the weights $f_C: \text{scp}(C) \rightarrow \mathbb{Q}$ such that $\text{ext}_{\mathcal{J}}(\tau, \tau') = 1$ if $(\tau, \tau') \in \{(\sigma_0, \sigma_0), (\rho, \rho)\}$, and otherwise $\text{ext}_{\mathcal{J}}(\tau, \tau') = 0$.*

Proof of Claim. For some feasible selection $S \subseteq \{a, b, c, a', b', c'\}$ of vertices of J , by the definition of R , the set S cannot contain both a and b . More formally, $S \cap \{a, b\}$ can be in three possible “states”, that is, it can be $\emptyset$, $\{a\}$, or $\{b\}$. An analogous statement holds for the set $\{a', b'\}$.

We define $\Phi = \{\emptyset, \{a\}, \{b\}\}$ and $\Phi' = \{\emptyset, \{a'\}, \{b'\}\}$. For $\tau \in \mathbb{T}$ and $\phi \in \Phi$, we define $m_{\tau, \phi}$ as the number of possible ways to select c that can be extended to a feasible selection S of J in which u is in state τ and $S \cap \{a, b\} = \phi$. For $\tau' \in \mathbb{T}$ and $\phi' \in \Phi'$, we define $m_{\tau', \phi'}$ analogously.

By checking for which selection status the constraint of the $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertex c can be satisfied, we obtain

$$M := (m_{\tau, \phi})_{\tau \in \mathbb{T}, \phi \in \Phi} = \begin{pmatrix} & \emptyset & \{a\} & \{b\} \\ \sigma_0 & 1 & 1 & 0 \\ \sigma_{\geq 1} & 1 & 1 & 2 \\ \rho & 1 & 2 & 1 \end{pmatrix}$$

By the symmetric structure of J , it follows that $M' := (m_{\tau', \phi'})_{\tau' \in \mathbb{T}, \phi' \in \Phi} = M$ if we use the same order of rows and columns for M' .

For $\tau, \tau' \in \mathbb{T}$, $\phi \in \Phi$, and $\phi' \in \Phi'$, we observe that $m_{\tau, \phi} \cdot m_{\tau', \phi'}$ denotes the number of partial solutions $S \in \mathcal{S}(\tau, \tau')$ with $S \cap \{a, a', b, b'\} = \phi \cup \phi'$. Then, we have

$$\text{ext}_{\mathcal{J}}(\tau, \tau') = \sum_{\phi \in \Phi, \phi' \in \Phi'} m_{\tau, \phi} \cdot m_{\tau', \phi'} \cdot f_C(\phi \cup \phi').$$

Now, we define the orderings $\sigma_0 < \sigma_{\geq 1} < \rho$ and $\emptyset < \{a\}, \{a'\} < \{b\}, \{b'\}$, which agrees with the definition of M as above. When we extend this to the corresponding lexicographical ordering of the tuples $(\tau, \tau') \in \mathbb{T}^2$ and $(\phi, \phi') \in \Phi \times \Phi'$, we have

$$(\text{ext}_{\mathcal{J}}(\tau, \tau'))_{(\tau, \tau') \in \mathbb{T}^2} = (M \otimes M) \cdot (f_C(\phi \cup \phi'))_{(\phi, \phi') \in \Phi \times \Phi'},$$

where $\otimes$ denotes the Kronecker product of matrices. Now, observe that M is invertible, and consequently, $M \otimes M$ is invertible. So, for every vector $\vec{v}$, we can set

$$(f_C(\phi \cup \phi'))_{(\phi, \phi') \in \Phi \times \Phi'} = (M \otimes M)^{-1} \cdot \vec{v}. \quad (10.9.1)$$

Especially, when we set $\vec{v} \in \{0, 1\}^{\mathbb{T}^2}$ with $\vec{v}[(\tau, \tau')] := \text{ext}_{\mathcal{J}}(\tau, \tau')$ for all $(\tau, \tau') \in \mathbb{T}^2$, the function f_C satisfies the constraints from the claim. As f_C can be efficiently computed, this concludes the proof. $\blacktriangleleft$

Now, we can use the gadget $\mathcal{J}$ to construct the new instance.

Constructing the Target Instance. As mentioned above, we use the gadget $\mathcal{J}$ to ensure that all pairs z and z_Z have the same selection status in G' . Given G' , we define an instance $I^* = (G^*, \lambda)$ of q -relation-weighted (σ, ρ) -#DOMSET $^{\frac{1}{2}}_{\{(\mathbb{N}_{\geq 1}, \mathbb{N})\}, \text{Rel}^{\frac{1}{2}}}$, where G^* is a graph with weighted relations, and λ is the mapping that determines which vertices are (σ, ρ) -vertices, and which are $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertices. We choose q later.

For each $Z \in \mathcal{Z}$ and $z \in Z$, we create a copy $J_{Z,z}$ of the gadget $\mathcal{J} = (J, \{u, u'\})$ from above and identify u with z and u' with z_Z .

Then, we obtain the graph with relations G^* from I^* as the union of G' and $\{J_{Z,z} \mid Z \in \mathcal{Z}, z \in Z\}$. We choose λ such that it coincides with $\lambda_{\mathcal{J}}$ on each copy of $\mathcal{J}$, and such that all vertices of G' (this includes the vertices in U) are (σ, ρ) -vertices, that is, only the copies of c and c' from $\mathcal{J}$ are $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertices. Note that the scope of the relations of G' contains precisely the vertices from Z' whenever the original relation observes the vertices in Z , for some $Z \in \mathcal{Z}$.

Correctness. A *configuration* is a function γ that maps each pair (Z, z) with $Z \in \mathcal{Z}$ and $z \in Z$ to a pair of states $(\gamma_1(Z, z), \gamma_2(Z, z)) \in \mathbb{T}^2$. We say that a solution of q -relation-weighted (σ, ρ) -#DOMSET $_{\{(\mathbb{N}_{\geq 1}, \mathbb{N})\}, \text{Rel}^{\ddagger}}$ on input I^* is a γ -selection if, for each pair (Z, z) , the vertex z is in state $\gamma_1(Z, z)$ with respect to $J_{Z,z}$, and z_Z is in state $\gamma_2(Z, z)$ with respect to $J_{Z,z}$.

Recall that each solution S of I^* has a weight $f(S) = \prod_{C \in \mathcal{C}} f_C(S \cap \text{scp}(C))$, where the product is over all relational constraints of I^* . When we write Γ for the set of all configurations, then, for each $\gamma \in \Gamma$, we denote by $\#\text{Sol}_{\gamma}(I^*)$ the sum of weights $f(S)$ over all γ -selections of I^* . Hence, the number of solutions for (σ, ρ) -#DOMSET $_{\{(\mathbb{N}_{\geq 1}, \mathbb{N})\}, \text{Rel}^{\ddagger}}$ on input I^* satisfies

$$\#\text{Sol}(I^*) = \sum_{\gamma \in \Gamma} \#\text{Sol}_{\gamma}(I^*).$$

As a next step, we extend the notion of γ -selections of the graph G' . A selection of vertices from G' is *compatible* with a configuration γ if it can be extended to a γ -selection of I^* . We denote by $\#\text{Sol}_{\gamma}(G')$ the number of selections of G' that are compatible with γ .

We relate $\#\text{Sol}_{\gamma}(I^*)$ and $\#\text{Sol}_{\gamma}(G')$ as follows.

Claim 10.9.2.2. *We have*

$$\#\text{Sol}_{\gamma}(I^*) = \#\text{Sol}_{\gamma}(G') \cdot \prod_{Z \in \mathcal{Z}, z \in Z} \text{ext}_{\mathcal{J}}(\gamma_1(Z, z), \gamma_2(Z, z)).$$

Proof of Claim. Let S' be some fixed set of vertices from G' that is compatible with γ . We analyze in how many ways we can extend S' to a γ -selection of I^* .

For each $Z \in \mathcal{Z}$ and for each $z \in Z$, we can choose any extension to $J_{Z,z}$ in which z has state $\gamma_1(Z, z)$ and z_Z has state $\gamma_2(Z, z)$. Note that these extensions are independent of the selected vertices in other copies of J . Here, we use the fact that we defined the extensions to $\mathcal{J}$ in a way that precisely captures the constraints that are imposed by a $\lambda^{\ddagger}$ -set. In particular, we do *not* require the copies of a, a', b , and b' to act as (σ, ρ) -vertices since they are in the scope of the relation R_v . Instead, these vertices act as $(\mathbb{N}, \mathbb{N})$ -vertices.

Hence, the set S' contributes by a value of $\prod_{Z \in \mathcal{Z}, z \in Z} \text{ext}_{\mathcal{J}}(\gamma_1(Z, z), \gamma_2(Z, z))$ to the weighted sum $\#\text{Sol}_{\gamma}(I^*)$. Note that this term is actually independent of S' . $\triangleleft$

From Claim 10.9.2.1 we know that $\#\text{Sol}_{\gamma}(I^*)$ is non-zero if and only if, for each $Z \in \mathcal{Z}$ and $z \in Z$, the pair $(\gamma_1(Z, z), \gamma_2(Z, z))$ is one of (σ_0, σ_0) or (ρ, ρ) . Moreover, in this case,

Here, we can use the *number* of solutions as the relations in G' are unweighted, that is, the accepted inputs have weight 1.

we have $\#Sol_Y(I^*) = \#Sol_Y(G')$. We say that configurations with this property are *good*. Consequently, the output on input I^* is

$$\sum_{Y \in \Gamma} \#Sol_Y(I^*) = \sum_{\text{good } Y \in \Gamma} \#Sol_Y(G').$$

Note that the solutions that contribute to $\sum_{\text{good } Y \in \Gamma} \#Sol_Y(G')$ are in a one-to-one correspondence to the solutions of the original problem (σ, ρ) - $\#DOMSET^{\text{Rel}}$ on input G . To see this, we observe that, for both allowed pairs of states (σ_0, σ_0) or (ρ, ρ) , the corresponding vertices z and z_Z have the same selection status. Moreover, if vertex z is selected, then it does not obtain additional selected neighbors from $J_{Z,z}$, and therefore it has some $s \in \sigma$ selected neighbors in G . If vertex z is unselected, then it may obtain additional selected neighbors from $J_{Z,z}$. However, this does not matter as $\rho = \mathbb{N}$ and therefore, vertex z still has $r \in \rho$ selected neighbors in G .

So, the solutions that contribute to $\sum_{\text{good } Y \in \Gamma} \#Sol_Y(G')$ are precisely those (σ, ρ) -sets of the original graph G in which each vertex $z \in Z$ is selected if and only if $z_Z \in Z'$ is selected. We have observed earlier that these are precisely the solutions of (σ, ρ) - $\#DOMSET^{\text{Rel}}$ on input G .

Analyzing the Reduction. Finally, it remains to show that the reduction is a pa-reduction. First, note that all weights are either 0, 1, or defined by Claim 10.9.2.1, in which case f_C has at most $2^4 = 16$ weights because of the size of the scope. Hence, there is some constant upper bound q on the number of different weights.

Second, the reduction is arity-preserving as all relations in I^* are the copy of some relation from G , or otherwise they are contained in $\mathcal{J}$, in which case they have arity 4.

Third, the reduction is pathwidth-preserving. To see this, recall that in the path decomposition associated with G , for each set $Z \in \mathcal{Z}$, there is some bag that contains all vertices of Z . For each $z \in Z$, we can add a copy of this bag that also contains the vertices of the copy $J_{Z,z}$ of $\mathcal{J}$, that is, a constant number of additional vertices. $\square$

In the next two steps, we show how to simulate the weights of the relations. Instead of explicitly mentioning the $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertices introduced in Lemma 10.9.2, we state the result for arbitrary sets of pairs $\mathcal{P}$.

As a first step, we replace the weighted relations by weighted vertices. Afterward, we remove the weighted vertices and thus, the weights from the instance. The proofs are analogous to the results presented in Section 10.5. To make this fact easily verifiable, we give the slightly modified constructions.

Lemma 10.9.3. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . For all constants q , there is a pa-reduction from q -relation-weighted (σ, ρ) - $\#DOMSET^{\mathcal{P}, \text{Rel}}$ to q -vertex-weighted (σ, ρ) - $\#DOMSET^{\mathcal{P}, \text{Rel}}$.*

Proof. We follow the ideas behind the proof of Lemma 10.5.6. The only difference to that construction is that here we do not need the $\{\sigma_s, \rho_r\}$ -providers that were used in Lemma 10.5.6. The purpose of these providers was to ensure that the newly introduced vertices that are not adjacent to other vertices, that is, $v_1, \dots, v_q$ in the following, receive a feasible number of neighbors depending on their selection status. We can easily omit these providers as these

By the definition of the function from Equation (10.9.1) there are at most 9 weights.

vertices are in the scope of the relation R and, by the definition of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}, \text{Rel}^\sharp}$, such vertices can now be treated like $(\mathbb{N}, \mathbb{N})$ -vertices. Hence, the previous proof goes through without any additional changes.

For completeness, we give the modified construction. For a fixed q , let $w_1, \dots, w_{q'}$ be the $q' \leq q$ different weights used by the relations of a q -relation-weighted $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}, \text{Rel}^\sharp}$ instance G . Formally, we have $\{w_1, \dots, w_{q'}\} = \{f_C(U) \mid C \in \mathcal{C}, U \subseteq \text{scp}(C)\}$. For ease of notation, we assume without loss of generality that $q' = q$ in the following. We apply the following modification for all constraints of G to form an instance G' of q -vertex-weighted $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}, \text{Rel}^\sharp}$.

For each $C \in \mathcal{C}$, we introduce q new vertices $v_1, \dots, v_q$ which form an independent set. The weight function wt assigns weight 1 to all vertices of the original instance, and for each $i \in [1..q]$ it assigns $\text{wt}(v_i) = w_i$. We replace the weighted relational constraint C by an unweighted relational constraint C' with $\text{scp}(C') := \text{scp}(C) \cup \{v_1, \dots, v_q\}$. The new constraint C' behaves, that is, accepts or rejects, in exactly the same way as C on the scope of C , but whenever C accepts with weight w_i for some $i \in [1..q]$, the new constraint C' also forces the vertex v_i to be selected whereas all the vertices v_j with $i \neq j$ are forced to be unselected.

The reduction increases pathwidth by at most q . Indeed, for each relation, there is a bag containing its scope, and we can copy this bag and add the additional vertices $v_1, \dots, v_q$ to this bag. Moreover, the arity of the instance increases by at most q . $\square$

As a next step, we remove the vertex weights from the vertices to obtain an unweighted instance of the problem. We follow the proof of Lemma 10.5.7 for this.

Lemma 10.9.4. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . For all constants q , there is a pa-reduction from q -vertex-weighted $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}, \text{Rel}^\sharp}$ to unweighted $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}, \text{Rel}^\sharp}$.*

Proof. As for the previous proof, the only difference to the proof from Lemma 10.5.7 is that the $\{\sigma_s, \rho_r\}$ -providers are omitted from the construction. As mentioned before, this does not incur any issues in this setting as the portals of these providers are always in the scope of a relation. As a consequence, they behave as $(\mathbb{N}, \mathbb{N})$ -vertices.

We use known interpolation techniques to remove the vertex weights. Let G be the graph with relations, and let $\text{wt}: V(G) \rightarrow \mathbb{Q}$ be the vertex weight function of a given q -vertex-weighted $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}, \text{Rel}^\sharp}$ instance I . For some $q' \leq q$, let $w_1, \dots, w_{q'}$ be the distinct values of the vertex weights, that is, $\{w_1, \dots, w_{q'}\} = \{\text{wt}(v) \mid v \in V(G)\}$. For ease of notation, we assume that $q' = q$.

We replace each weight w_i by a variable x_i and treat the unknown weighted sum of solutions of I as a polynomial P in the q variables $x_1, \dots, x_q$. Observe that there are at most n vertices, and hence, the maximum degree is at most n for each variable. Moreover, if we can realize all combinations of $n + 1$ different weights for each variable x_i , then we can use Fact 2.1.7 to recover the coefficients of P in time $\mathcal{O}((n + 1)^{3q})$, which is polynomial in n as q is a constant. Then, we can output $P(w_1, \dots, w_q)$ to recover the original solution.

It remains to realize $n + 1$ different weights for each variable. For this, it suffices to realize weights of the form 2^ℓ .

Let v be the vertex for which we want to realize weight 2^ℓ . For this, we introduce ℓ new vertices $v_1, \dots, v_\ell$. Moreover, we add a relational constraint C_j with scope $\{v, v_j\}$ that ensures that if v is unselected, then the new vertex v_j must also be unselected, but if vertex v is selected, then the new vertex v_j can be either selected or unselected. Hence, whenever vertex v is selected, this construction contributes a factor of 2^ℓ to the solution, whereas if v is unselected, then it contributes only a factor of 1.

Observe that the arity of the resulting graph increases by at most 2 as the new relational constraints C_j have arity 2 and no other relations are introduced.

It is straightforward to see that this modification does not change the pathwidth too much. Indeed, for each v , we can pick an arbitrary bag containing v , duplicate this bag for each $j \in [1 \dots \ell]$, and then add v_j to this copy of the bag. The claim then directly follows. $\square$

We summarize the reductions from Lemmas 10.9.2 to 10.9.4 by Lemma 10.9.5.

Lemma 10.9.5. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$ where σ is cofinite and $\rho = \mathbb{N}$. For all constants d , there is a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#DOMSET^{Rel}$ on instances with arity at most d to $(\sigma, \rho)\text{-}\#DOMSET^{(\mathbb{N}_{\geq 1}, \mathbb{N}), Rel^\ddagger}$ with arity at most d .*

Proof. By chaining Lemma 10.9.2 together with Lemmas 10.9.3 and 10.9.4, where we set $\mathcal{P} = \{(\mathbb{N}_{\geq 1}, \mathbb{N})\}$ in the latter two results, we obtain a pa-reduction from $(\sigma, \rho)\text{-}\#DOMSET^{Rel}$ to $(\sigma, \rho)\text{-}\#DOMSET^{(\mathbb{N}_{\geq 1}, \mathbb{N}), Rel^\ddagger}$. Since these reductions are arity-preserving, in order to solve $(\sigma, \rho)\text{-}\#DOMSET^{Rel}$ on instances of arity at most d , it suffices to solve $(\sigma, \rho)\text{-}\#DOMSET^{(\mathbb{N}_{\geq 1}, \mathbb{N}), Rel^\ddagger}$ on instances of arity at most d' , where d' is d plus some constant. $\square$

As a next result, we show that we can replace arbitrary relations by a single type of relations.

10.9.2 Reducing to One Type of Relations

In the following, we show how to replace arbitrary relations by gadgets involving only $HW_{=1}^\ddagger$ -relations. The proof is very similar to the results from Section 10.6.

Recall that we write $HW_{=1}^\ddagger$ for the $HW_{=1}$ -relation to emphasize that the relation is interpreted differently in the context of $(\sigma, \rho)\text{-}\#DOMSET^{Rel^\ddagger}$. The definition of the relation itself remains unchanged. We extend this notion to other relations as well. In the following, we especially use $EQ^\ddagger$ for the EQ-relations.

Lemma 10.9.6. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$ where σ is cofinite and $\rho = \mathbb{N}$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . For all constants d , there is a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}, Rel^\ddagger}$ with arity at most d to $(\sigma, \rho)\text{-}\#DOMSET^{\mathcal{P}, HW_{=1}^\ddagger}$.*

Proof. The proof consists of two parts. First, we replace arbitrary relations by gadgets which use only $EQ^\ddagger$ and $HW_{=1}^\ddagger$ as relations. In the second part, we then also remove the $EQ^\ddagger$ -relations, and replace them by gadgets using only $HW_{=1}^\ddagger$ -relations. The first part is analogous to the proofs of Lemmas 10.6.3 and 10.6.4, and the second part is based on the proofs of Lemmas 10.6.5 and 10.6.6.

Removing Arbitrary Relations. The only difference to the earlier results from Section 10.6 is the fact that we omit the $\{\sigma_s, \rho_r\}$ -providers from the construction. For completeness, we give the construction of the gadget which replaces each arbitrary relation in the following.

Given a relation $R = \{r_1, \dots, r_{|R|}\}$, where $r_i \subseteq [1 \dots d']$ for $d' \leq d$, we define a graph with portals $H_R = (G, U)$ with $U = \{u_1, \dots, u_{d'}\}$. For each set r_i , we add $|[1 \dots d'] \setminus r_i|$ independent vertices $s_{i,j}$ with $j \in [1 \dots d'] \setminus r_i$, and we also add an extra vertex t_i to G . Next, we introduce an $\text{EQ}^\ddagger$ -relation of arity $|[1 \dots d'] \setminus r_i| + 1$ that observes the vertices $s_{i,\star}$ and t_i . Additionally, we add a $|R|$ -ary $\text{HW}_{=1}^\ddagger$ -relation that observes the vertices $t_\star$. Finally, for each $j \in [1 \dots d]$, we introduce a $\text{HW}_{=1}^\ddagger$ -relation that observes u_j and all vertices $s_{\star,j}$.

For a given (σ, ρ) - $\# \text{DOMSET}^{\mathcal{P}, \text{Rel}^\ddagger}$ instance H , we replace all relations R that are neither $\text{EQ}^\ddagger$ -nor $\text{HW}_{=1}^\ddagger$ -relations by the corresponding graph H_R , where we identify the portals of H_R with the respective vertices in the scope of R .

Let H' be the new instance of (σ, ρ) - $\# \text{DOMSET}^{\mathcal{P}, \text{EQ}^\ddagger, \text{HW}_{=1}^\ddagger}$. It follows from the proofs of Lemmas 10.6.3 and 10.6.4 that the number of solutions of H and H' are equal. Moreover, as the arity in the source problem is constant, the modification does not change the pathwidth by more than an additive constant (see Lemma 10.6.4 for the details).

Removing Equality Relations. Let H be an instance of (σ, ρ) - $\# \text{DOMSET}^{\mathcal{P}, \text{EQ}^\ddagger, \text{HW}_{=1}^\ddagger}$. We replace each relation EQ_k by the graph H_k , which is defined as follows, to obtain a (σ, ρ) - $\# \text{DOMSET}^{\mathcal{P}, \text{HW}_{=1}^\ddagger}$ instance H' .

The graph with portals $H_k = (G_k, U_k)$, where we have $U_k = \{u_1, \dots, u_k\}$, contains a single vertex v and, for each $i \in [1 \dots k]$, a $\text{HW}_{=1}^\ddagger$ -relation between the vertices v and u_i .

It directly follows from the construction and the proofs of Lemmas 10.6.5 and 10.6.6 that the number of solutions for H and H' are equal, the pathwidth increases only by an additive constant, and the arity increases by at most 2.

When we first apply the first part of the proof and then the second part, the statement follows. $\square$

As a next step, we show how to remove the $\text{HW}_{=1}^\ddagger$ -relations.

10.9.3 Realizing Relations

In the following, we show that we can replace the remaining $\text{HW}_{=1}^\ddagger$ -relations by (σ, ρ) -vertices. Formally, we prove Lemma 10.9.11.

Lemma 10.9.11. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$ where σ is cofinite and $\rho = \mathbb{N}$. For all constants d , there is a pathwidth-preserving reduction from (σ, ρ) - $\# \text{DOMSET}^{(\mathbb{N}_{\geq 1}, \mathbb{N}), \text{HW}_{=1}^\ddagger}$ with arity at most d to (σ, ρ) - $\# \text{DOMSET}$.*

We split the proof into four steps. We refer to Figure 10.19 for an overview.

1. In a first step, which is analogous to Lemma 10.8.3, we replace the $\text{HW}_{=1}^\ddagger$ -relations by $\text{HW}_{\geq 1}^\ddagger$ -relations. We formalize this in Lemma 10.9.7.

2. Next, we remove the $\text{HW}_{\geq 1}^\dagger$ -relations and make use of additional $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertices together with a single $(\mathbb{N}, \emptyset)$ -vertex. We provide this reduction in Lemma 10.9.8.
3. In Lemma 10.9.9, we replace the $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertices by a constant number of $(\mathbb{N}, \emptyset)$ -vertices.
4. Finally, in Lemma 10.9.10, we model $(\mathbb{N}, \emptyset)$ -vertices based on (σ, ρ) -vertices only. Here, we crucially require that there are not too many $(\mathbb{N}, \emptyset)$ -vertices, that is, only a constant number of them.

Afterward, we combine the reductions and prove Lemma 10.9.11.

Step 1: Replacing $\text{HW}_{=1}^\dagger$. As a first step, we show how to replace the $\text{HW}_{=1}^\dagger$ -relations by $\text{HW}_{\geq 1}^\dagger$ -relations.

Lemma 10.9.7. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$ where σ is cofinite and $\rho = \mathbb{N}$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . Then, $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}, \text{HW}_{=1}^\dagger} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}, \text{HW}_{\geq 1}^\dagger}$.*

Proof. The proof is almost identical to that of Lemma 10.8.3. Assume we are given an instance $I = (G, \mathcal{C}, \lambda)$ of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}, \text{HW}_{=1}^\dagger}$. Let $\mathcal{Z} = \{\text{scp}(C) \mid C \in \mathcal{C}\}$, and define U as the union of the sets in $\mathcal{Z}$.

For a positive integer x , let I_x be the instance of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}, \text{HW}_{\geq 1}^\dagger}$ obtained from G where we attach, for each $Z \in \mathcal{Z}$, a relational constraint with relation $\text{HW}_{\geq 1}^\dagger$ and scope Z . In addition, for each subset Z' of Z with $|Z'| = |Z| - 1$, we introduce a total of x (σ, ρ) -vertices $u_1^{(Z, Z')}, \dots, u_x^{(Z, Z')}$. For each $i \in [1 \dots x]$, we introduce a relation $\text{HW}_{\geq 1}^\dagger$ with scope $Z' \cup \{u_i^{(Z, Z')}\}$.

As in previous proofs, a selection of vertices from G is *good* if all vertices in $V(G) \setminus U$ have a feasible number of selected neighbors. If, in a good selection S , some Z' has at least one selected vertex, then there are 2^x feasible extensions to the vertices $u_1^{(Z, Z')}, \dots, u_x^{(Z, Z')}$. If no vertex of Z' is selected, then there is only a single such extension; all vertices $u_1^{(Z, Z')}, \dots, u_x^{(Z, Z')}$ are selected. Moreover, we say that if a set $Z \in \mathcal{Z}$ contains at least two selected vertices in S , then it is *undesired*. Otherwise, that is, if it contains precisely one selected vertex, then it is *desired*.

Observe that each $Z \in \mathcal{Z}$ has to contain at least one selected vertex, that is, Z is either desired or undesired. For an undesired set $Z \in \mathcal{Z}$, there is at least one vertex selected in each of the $|Z|$ corresponding subsets Z' ; this gives a total of $2^{x \cdot |Z|}$ extensions. In contrast, for a desired set Z , there is exactly one subset Z' that is entirely unselected; this gives a total of $2^{x \cdot (|Z| - 1)}$ extensions.

For $\mathcal{Z}' \subseteq \mathcal{Z}$, denote by $a_{\mathcal{Z}'}$ the number of good selections in which precisely the sets in $\mathcal{Z}'$ are undesired and the sets in $\mathcal{Z} \setminus \mathcal{Z}'$ are desired. Similarly, denote by a_i the number of good selections in which precisely i of the sets in $\mathcal{Z}$ are undesired. When we write $\#\text{Sol}(I_x)$ for the

number of solutions of I_x , we have

$$\begin{aligned} \# \text{Sol}(I_x) &= \sum_{Z' \subseteq Z} a_{Z'} \cdot \left(\prod_{Z \in Z'} 2^{x \cdot |Z|} \cdot \prod_{Z \notin Z'} 2^{x \cdot (|Z|-1)} \right) \\ &= 2^{x \cdot \sum_{Z \in \mathcal{Z}} (|Z|-1)} \cdot \sum_{Z' \subseteq Z} a_{Z'} \cdot 2^{x \cdot |Z'|} \\ &= F \cdot \sum_{i=0}^{|Z|} a_i \cdot 2^{x \cdot i} \end{aligned}$$

where we set

$$F := 2^{x \cdot \sum_{Z \in \mathcal{Z}} (|Z|-1)}.$$

Note that $\# \text{Sol}(I_x)/F$ is a polynomial in 2^x of degree $|Z|$. Hence, by choosing $|Z|$ different values of x , we can use polynomial interpolation to recover the coefficients a_i . In particular, a_0 is the number of good selections of vertices from G in which no set in $\mathcal{Z}$ is undesired, that is, every set $Z \in \mathcal{Z}$ is desired, that is, every set $Z \in \mathcal{Z}$ contains exactly one selected vertex. We conclude that a_0 is precisely the number of solutions of instance I of (σ, ρ) - $\# \text{DOMSET}^{\mathcal{P}, \text{HW}_{\geq 1}^\ddagger}$ which we want to compute.

Recall that the scope $Z \in \mathcal{Z}$ of each relation of I is considered to be a clique in the definition of the pathwidth of I . Using this we can add, for each $Z' \subseteq Z$ with $|Z'| = |Z| - 1$, the vertices $v_1^{(Z, Z')}, \dots, v_x^{(Z, Z')}$ one after another to a new copy of the original bag containing the clique Z . Hence, the pathwidth of I_x is at most that of I plus an additive constant. $\square$

Step 2: Replacing $\text{HW}_{\geq 1}^\ddagger$. As the next step, we remove the relations from the instance. For this reduction, we use $(\mathbb{N}, \emptyset)$ -vertices, that is, vertices which are forced to be selected, as well as $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertices. Note that the reduction from Lemma 10.9.2 introduces $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertices as well. However, we explicitly add these vertices to the result, to emphasize that they are needed in the reduction.

Lemma 10.9.8. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$ where σ is cofinite and $\rho = \mathbb{N}$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . For every constant $c \geq 0$, (σ, ρ) - $\# \text{DOMSET}^{\mathcal{P}, \text{HW}_{\geq 1}^\ddagger} \leq_{\text{pw}} (\sigma, \rho)$ - $\# \text{DOMSET}^{\mathcal{P} + (\mathbb{N}_{\geq 1}, \mathbb{N}) + (\mathbb{N}, \emptyset)}$, where $(\mathbb{N}, \emptyset)$ is c -bounded in the source problem, and $(\mathbb{N}, \emptyset)$ is $c + 1$ -bounded in the target problem.*

Proof. The proof uses a simple interpolation. Let $I = (G, \lambda)$ with $G = (V, E, C)$ be an instance of (σ, ρ) - $\# \text{DOMSET}^{\mathcal{P}, \text{HW}_{\geq 1}^\ddagger}$. We set $\mathcal{Z} = \{\text{scp}(C) \mid C \in C\}$, and let U denote the union of sets in $\mathcal{Z}$. As all relational constraints use $\text{HW}_{\geq 1}$ -relations, the $\lambda^\ddagger$ -sets of G are precisely the selections of vertices from V for which (1) every vertex in $V \setminus U$ has a feasible number of selected neighbors according to the constraints from λ , and (2) at least one vertex from each set $Z \in \mathcal{Z}$ is selected.

For a positive integer x , we define the instance $I_x = (G', \lambda')$ of (σ, ρ) - $\# \text{DOMSET}^{\mathcal{P} + (\mathbb{N}_{\geq 1}, \mathbb{N}) + (\mathbb{N}, \emptyset)}$ based on the graph (V, E) , that is, the graph G without relational constraints, as follows. We add a single $(\mathbb{N}, \emptyset)$ -vertex w . Then, for each set $Z \in \mathcal{Z}$, we add a total of x new $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertices

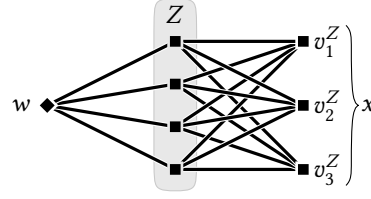


Figure 10.21: The construction of the instance I_x from Lemma 10.9.8 to simulate $\text{HW}_{\geq 1}^{\dagger}$. Here shown for $x = 3$ and a single set Z that was subject to a relation $\text{HW}_{\geq 1}^{\dagger}$. The vertex w is a $(\mathbb{N}, \emptyset)$ -vertex and the other vertices are $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertices.

$v_1^Z, \dots, v_x^Z$. We connect the vertices in Z completely to all vertices $v_{\star}^Z$ and to the vertex w . Moreover, we define λ' such that each vertex in U is a $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertex. Additionally, on the vertices in $V \setminus U$, the function λ' is identical to λ . We refer to Figure 10.21 for an illustration of this construction.

A selection of vertices from V is *good* if all vertices in $V \setminus U$ have a feasible number of selected neighbors according to their constraints defined by λ .

Now, let us consider the number of extensions of a good selection S to solutions of I_x . Observe that the $(\mathbb{N}, \emptyset)$ -vertex w has to be selected in every solution of I_x . This implies that, for each $Z \in \mathcal{Z}$, all vertices in Z —and they are $(\mathbb{N}_{\geq 1}, \mathbb{N})$ -vertices—have a feasible number of selected neighbors independently of how many neighbors they have in S .

Consider a set $Z \in \mathcal{Z}$ and assume that Z contains at least one selected vertex in a good selection. Then, there are 2^x extensions to the vertices $v_1^Z, \dots, v_x^Z$; each vertex can be selected to not. If no vertex in Z is selected, then there is only a single extension; all vertices $v_{\star}^Z$ are unselected.

For $i \in [0 \dots |\mathcal{Z}|]$, we denote by a_i the number of good selections in which exactly i of the sets in $\mathcal{Z}$ contain at least one selected vertex. Then, the number of solutions $\#\text{Sol}(I_x)$ of I_x satisfies

$$\#\text{Sol}(I_x) = \sum_{i=0}^{|\mathcal{Z}|} a_i \cdot 2^{x \cdot i}.$$

We observe that this is a polynomial in 2^x of degree $|\mathcal{Z}|$. Hence, we can choose $|\mathcal{Z}|$ different values of x , and use polynomial interpolation to recover the coefficients a_i . In particular, $a_{|\mathcal{Z}|}$ is the number of good selections in which every set $Z \in \mathcal{Z}$ contains at least one selected vertex, which is precisely the number of solutions of instance I of (σ, ρ) - $\text{DOMSET}^{\mathcal{P}, \text{HW}_{\geq 1}^{\dagger}}$ which we want to compute.

Since each set $Z \in \mathcal{Z}$ is considered to be a clique in the definition of the pathwidth of I , we can add all vertices $v_1^Z, \dots, v_x^Z$ one after another to a new copy of the original bag containing the clique Z . Moreover, we can add the vertex w to all bags of the decomposition. Hence, the pathwidth of I_x is at most that of I plus an additive constant. $\square$

Step 3: Shifting the Sets. Recall that, for a positive integer s and a set τ of integers, we use the notation $\tau - s$ to denote the set $\{t - s \mid t \in \tau, t - s \geq 0\}$. In the next step, we use s $(\mathbb{N}, \emptyset)$ -vertices to “shift” some pair (σ', ρ') down to $(\sigma' - s, \rho' - s)$.

While the shifting works for any pair (σ', ρ') , we apply this result to $(\sigma', \rho') = (\sigma, \rho)$ with $s = s_{\text{top}} - 1$ to model the pair $(\sigma - s, \rho - s) = (\mathbb{N}_{\geq 1}, \mathbb{N})$ where we use that $\rho = \mathbb{N}$.

Lemma 10.9.9. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$ where σ is cofinite and $\rho = \mathbb{N}$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 . Additionally, fix another pair of sets $(\sigma', \rho') \in \Omega^2$.*

For all constants $c, s \geq 0$, we have that $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\mathbb{N}, \emptyset)+(\sigma'-s, \rho'-s)} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\mathbb{N}, \emptyset)+(\sigma', \rho')}$, where $(\mathbb{N}, \emptyset)$ is c -bounded in the source problem, and $(\mathbb{N}, \emptyset)$ is $c + s$ -bounded in the target problem.

Proof. Let I be an instance of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\mathbb{N}, \emptyset)+(\sigma'-s, \rho'-s)}$ with at most c $(\mathbb{N}, \emptyset)$ -vertices. Let U be the set of $(\sigma' - s, \rho' - s)$ -vertices.

We define an instance $I' = (G', \lambda')$ of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\mathbb{N}, \emptyset)+(\sigma', \rho')}$ based on I , where we add s $(\mathbb{N}, \emptyset)$ -vertices $v_1, \dots, v_s$. Additionally, we make each vertex v_i adjacent to all vertices in U . We define λ' such that all vertices in U are (σ', ρ') -vertices in I' .

Observe that the new instance I' contains a total of at most $c + s$ $(\mathbb{N}, \emptyset)$ -vertices, that is, there are c such vertices from I and s new vertices. Note that the solutions of I' are in a one-to-one correspondence to the solutions of I for $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\mathbb{N}, \emptyset)+(\sigma'-s, \rho'-s)}$ as the vertices $v_1, \dots, v_s$ are always selected, and therefore, each vertex in U obtains precisely s additional selected neighbors.

Observe that we can add all vertices $v_1, \dots, v_s$ to every bag of a given path decomposition of I . Thus, the pathwidth increases by at most the constant s . $\square$

Step 4: Forcing Vertices to be Selected. In the final step in the chain of reductions, we remove the $(\mathbb{N}, \emptyset)$ -vertices from the instance, that is, the vertices that are forced to be selected.

We provide a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#\text{DOMSET}^{(\mathbb{N}, \emptyset)}$ where $(\mathbb{N}, \emptyset)$ is $(c + 1)$ -bounded to $(\sigma, \rho)\text{-}\#\text{DOMSET}^{(\mathbb{N}, \emptyset)}$ where $(\mathbb{N}, \emptyset)$ is c -bounded. That is, the pathwidth-preserving reduction removes a single $(\mathbb{N}, \emptyset)$ -vertex from the given $(\sigma, \rho)\text{-}\#\text{DOMSET}^{(\mathbb{N}, \emptyset)}$ instance. When we apply this reduction c times, we can remove all $(\mathbb{N}, \emptyset)$ -vertices. As c is a constant, this chain of reductions has constant length and is still pathwidth-preserving in total.

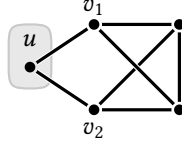
Lemma 10.9.10. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \Omega^2$ where σ is cofinite and $\rho = \mathbb{N}$, and let $\mathcal{P}$ be some set of pairs from Ω_0^2 .*

For every constant $c \geq 0$, $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\mathbb{N}, \emptyset)} \leq_{\text{pw}} (\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$, where $(\mathbb{N}, \emptyset)$ is $(c + 1)$ -bounded in the source problem, and $(\mathbb{N}, \emptyset)$ is c -bounded in the target problem.

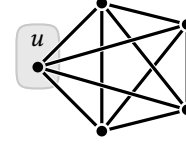
Proof. Let $I = (G, \lambda)$ be an instance of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}+(\mathbb{N}, \emptyset)}$, and let u be some $(\mathbb{N}, \emptyset)$ -vertex of I . If no such vertex exists, then we are done as we can simply cast I as an instance of $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$. For ease of notation, we assume that $c = 0$ and thus, we reduce to $(\sigma, \rho)\text{-}\#\text{DOMSET}^{\mathcal{P}}$.

We consider two cases depending on σ .

Case 1: $s_{\min} > 0$. In the first case, suppose that $s_{\min} > 0$. We first construct a gadget $\mathcal{J} = (J, \{u\})$ which we use in the construction.



(a) The gadget $\mathcal{J}$ for the case when $s_{\min} > 0$. Here shown for $s_{\min} + 1 = 4$.



(b) The gadget $\mathcal{J}$ for the case when $s_{\min} = 0$. Here shown for $t = 4 \notin \sigma$ as the smallest (positive) integer missing from σ .

Figure 10.22: The gadgets $\mathcal{J}$ from the proof of Lemma 10.9.10 for the two cases. All vertices are (σ, ρ) -vertices where $\rho = \mathbb{N}$ and σ is cofinite.

The graph J contains a modified clique C on $s_{\min} + 1$ vertices where we remove the edge between two arbitrary vertices v_1 and v_2 . We create an additional vertex u and connect it to v_1 and v_2 . All vertices of $\mathcal{J}$ are (σ, ρ) -vertices and we use u as portal. See Figure 10.22a for an illustration of the construction.

Observe that v_1 and v_2 have $s_{\min} - 1$ neighbors in C , whereas the remaining vertices of C have $s_{\min}$ neighbors in C .

If the portal u is unselected, then each of the vertices v_1 and v_2 can have at most $s_{\min} - 1$ selected neighbors. Hence, they cannot be selected in this case. Then, if the vertices v_1 and v_2 are unselected, the rest of C has to be unselected as well. Therefore, there is exactly one partial solution of $\mathcal{J}$; all vertices of C are unselected which also satisfies the ρ -constraints as $\rho = \mathbb{N}$.

If the portal u is selected, then there are exactly two partial solutions. For the first solution, all vertices of C unselected and u obtains no selected neighbors. In the second solutions all vertices of C are selected and u obtains two selected neighbors.

Next, we use this gadget to construct the new instance.

Construction of the Graph. For some yet to be determined integer x , we define an instance I_x of (σ, ρ) -#DOMSET $^{\mathcal{P}}$ based on I where we turn u into a (σ, ρ) -vertex and identify u with the portals of x copies of the gadget $\mathcal{J}$.

We say that a subset S of the vertices of G is called *good* if all vertices in $V(G) \setminus \{u\}$ have a feasible number of neighbors in S according to λ , that is, vertex u might be selected or unselected. First, consider a good subset S of the vertices of G that does *not* contain u . By the properties of $\mathcal{J}$, there is exactly one way to extend S to a solution S' of I_x , and that is by not selecting any vertices of $\mathcal{J}$.

Now, consider a good subset S of the vertices of G that *does* contain u and u has i selected neighbors in G . We can extend S to a solution S' of I_x in a way such that vertex u receives two selected neighbors from s of the x attached gadgets $\mathcal{J}$ for $i + 2s \in \sigma$. So, there are

$$\sum_{\substack{s \in [0 \dots x]: \\ i+2s \in \sigma}} \binom{x}{s} = 2^x - \sum_{\substack{s \in [0 \dots x]: \\ i+2s \notin \sigma}} \binom{x}{s} = 2^x - p_i(x)$$

extensions for S to a solution of I_x , where

$$p_i(x) := \sum_{i+2s \notin \sigma} \binom{x}{s}$$

is a polynomial in x .

Suppose that g denotes the number of good sets $S \subseteq V(G)$ with $u \notin S$. Assume that u has degree δ in G . For all $i \in [0..d]$, we define f_i as the number of good sets $S \subseteq V(G)$ with $u \in S$ and $|N(u) \cap S| = i$. Hence, the sum $\sum_{i=0}^{\delta} f_i$ denotes precisely the number of solutions of I .

When we write $\#Sol(I_x)$ for the number of solutions of I_x for (σ, ρ) - $\#DOMSET^P$, we have

$$\#Sol(I_x) = g + \sum_{i=0}^d f_i \cdot (2^x - p_i(x)).$$

Denote by n be the number of vertices of G . We define

$$L(x) := \min_{i \in [0..d]} (2^x - p_i(x)) \quad (10.9.2)$$

and claim that, for sufficiently large $x \in \mathcal{O}(n)$, we can compute the value $\sum_{i=0}^d f_i$ from $\#Sol(I_x)$ by computing $\lfloor \#Sol(I_x)/L(x) \rfloor$.

Intuitively, the value g and each of the f_i 's are upper-bounded by 2^n , whereas $L(x)$ is exponential in x . Therefore, dividing by $L(x)$ for sufficiently large x , eliminates the influence of g and ensures that we differ from $\sum_{i=0}^d f_i$ by less than 1.

Assuming our claim is true for a given x , we can easily compute the value $\#Sol(I_x)$ by a single oracle call to (σ, ρ) - $\#DOMSET^P$ as the polynomials $p_i(x)$ depend only on σ .

To conclude this case, we need to formalize the intuition.

Claim 10.9.10.1. *There is an $x \in \mathcal{O}(n)$ such that $\lfloor \#Sol(I_x)/L(x) \rfloor = \sum_{i=0}^d f_i$.*

Proof of Claim. From the definition of $L(x)$ in Equation (10.9.2), we directly get that

$$\frac{\#Sol(I_x)}{L(x)} \geq \sum_{i=0}^d f_i \cdot \frac{(2^x - p_i(x))}{\min_{i \in [0..d]} (2^x - p_i(x))} \geq \sum_{i=0}^d f_i.$$

To prove the claim, it remains to show that, for sufficiently large $x \in \mathcal{O}(n)$,

$$\frac{g}{L(x)} < \frac{1}{\delta + 2} \quad (10.9.3)$$

and

$$\frac{2^x - p_i(x)}{L(x)} < 1 + \frac{1}{(\delta + 2) \cdot f_i}. \quad (10.9.4)$$

With these two bounds and the definition of $\#Sol(I_x)$, we get

$$\begin{aligned} \frac{\#Sol(I_x)}{L(x)} &\leq \frac{g}{L(x)} + \sum_{i=0}^{\delta} f_i \cdot \frac{2^x - p_i(x)}{L(x)} \\ &< \frac{1}{\delta + 2} + \sum_{i=0}^{\delta} \left(f_i + \frac{1}{\delta + 2} \right) \\ &\leq 1 + \sum_{i=0}^{\delta} f_i, \end{aligned}$$

and therefore, we have $\lfloor \#Sol(I_x)/L(x) \rfloor = \sum_{i=0}^{\delta} f_i$.

We can choose x sufficiently large in $\mathcal{O}(n)$ such that $(d + 2) \cdot 2^n < L(x)$ holds. Then, Equation (10.9.3) follows, by the observation that $g \leq 2^n$.

To prove Equation (10.9.4), we denote by ℓ the index from $[0.. \delta]$ such that $p_{\ell}(x) = \max_i p_i(x)$, that is, we have $L(x) = 2^x - p_{\ell}(x)$. We observe that

$$\frac{2^x - p_i(x)}{L(x)} \leq \frac{2^x}{2^x - p_{\ell}(x)} = 1 + \frac{p_{\ell}(x)}{2^x - p_{\ell}(x)}.$$

Since $f_i \leq 2^n$ and $p_{\ell}(x)$ is a polynomial in x , we can choose x sufficiently large in $\mathcal{O}(n)$ such that

$$\frac{p_j(x)}{2^x - p_j(x)} < \frac{1}{(d + 2) \cdot 2^n}$$

which concludes the proof. ◀

This concludes the reduction for the case when $s_{\min} > 0$.

Case 2: $s_{\min} = 0$. In the remainder, assume that $s_{\min} = 0$. We use a similar approach as in the previous case, but with a different gadget $\mathcal{J}$.

Let t be the minimum integer missing from σ . So, we have $t \geq 1$ since $s_{\min} = 0$. In this case, the gadget $\mathcal{J}$ consists of a clique on $t + 1$ vertices where we pick an arbitrary vertex as the portal u . All the vertices of $\mathcal{J}$ are (σ, ρ) -vertices. See Figure 10.22b for an illustration of the construction.

When the portal u is unselected, there are 2^t partial solutions for $\mathcal{J}$; all vertices of the clique different from u can either be selected or unselected, which is fine as $\rho = \mathbb{N}$ and $[0.. t - 1] \subseteq \sigma$. If the portal u is selected, there are $2^t - 1$ partial solutions of $\mathcal{J}$; all the previous selections except selecting the entire clique as $t \notin \sigma$.

Construction of the Graph. Again, for some yet to be determined integer x , we define an instance I_x of (σ, ρ) - $\#DOMSET^P$ based on I where we make u a (σ, ρ) -vertex and attach x copies of a gadget $\mathcal{J}$, where u acts as the portal to each of the copies.

Next, we analyze the number of extensions for $\mathcal{J}$ assuming that u is selected and already has i selected neighbors. That is, vertex u must receive a valid number of neighbors with regard to the set σ . In this case, the number of extensions, denoted by $\text{ext}_i(x)$, is given by

$$\begin{aligned}\text{ext}_i(x) &:= \sum_{i+s \in \sigma} \sum_{\substack{i_1 + \dots + i_x = s: \\ i_s \leq t-1}} \prod_{j=1}^x \binom{t}{i_j} \\ &= (2^t - 1)^x - \sum_{i+s \notin \sigma} \sum_{\substack{i_1 + \dots + i_x = s: \\ i_s \leq t-1}} \prod_{j=1}^x \binom{t}{i_j} \\ &= (2^t - 1)^x - p_i(x)\end{aligned}$$

where

$$p_i(x) := \sum_{i+s \notin \sigma} \sum_{\substack{i_1 + \dots + i_x = s: \\ i_s \leq t-1}} \prod_{j=1}^x \binom{t}{i_j}$$

Intuitively, $p_i(x)$ is the number of possibilities to select s vertices from the x cliques such that $i + s \notin \sigma$ and no clique is entirely selected. Hence, $p_i(x)$ is upper-bounded by $q_i(x) := \sum_{i+s \notin \sigma} \binom{x \cdot t}{s}$, which is the number of possibilities to select s vertices from the x cliques of size t . In particular, $q_i(x)$ is a polynomial in x with $q_i(x) \geq p_i(x)$.

We define g and f_i as in the previous case, that is, g denotes the number of good sets $S \subseteq V(G)$ with $u \notin S$ and f_i denotes the number of good sets $S \subseteq V(G)$ with $u \in S$ and $|N(u) \cap S| = i$. Then, we obtain

$$\#\text{Sol}(I_x) = g \cdot 2^{t \cdot x} + \sum_{i=0}^{\delta} f_i \cdot ((2^t - 1)^x - p_i(x)).$$

We claim that we can compute $\sum_{i=0}^{\delta} f_i$ from $\#\text{Sol}(I_x)$ as $\lfloor (\#\text{Sol}(I_x) \bmod 2^{t \cdot x}) / L(x) \rfloor$, where $L(x) := \min_{i \in [0..d]} \text{ext}_i(x)$. We know that we can compute $\#\text{Sol}(I_x)$ with a single oracle-call to $(\sigma, \rho)\text{-}\#\text{DOMSET}^P$. Thus, if the claim is true, this concludes the reduction as the sum $\sum_{i=0}^{\delta} f_i$ corresponds to the number of solutions of I .

Claim 10.9.10.2. *For large enough $x \in \mathcal{O}(n)$, we have $\sum_{i=0}^d f_i = \lfloor (\#\text{Sol}(I_x) \bmod 2^{t \cdot x}) / L(x) \rfloor$.*

Proof of Claim. To simplify notation, we define $\psi(x) := \#\text{Sol}(I_x) \bmod 2^{t \cdot x}$. The first step is to show that $\psi(x) = \sum_{i=0}^{\delta} f_i \cdot \text{ext}_i(x)$. By the earlier observations and the fact that G has n vertices, we have

$$\begin{aligned}\sum_{i=0}^{\delta} f_i \cdot \text{ext}_i(x) &= \sum_{i=0}^{\delta} f_i \cdot ((2^t - 1)^x - p_i(x)) \\ &< (2^t - 1)^x \cdot \sum_{i=0}^{\delta} f_i \\ &\leq (2^t - 1)^x \cdot 2^n.\end{aligned}$$

Then, for sufficiently large $x \in \mathcal{O}(n)$, we have $2^n < (2^t/(2^t - 1))^x$, and consequently, $(2^t - 1)^x \cdot 2^n < 2^{t \cdot x}$. Hence, the sum $\sum_{i=0}^{\delta} f_i$ is not affected by the computation modulo $2^{x \cdot t}$.

By our choice of $L(x)$ as the minimum over all $\text{ext}_i(x)$, we directly obtain that $\psi(x) \geq \sum_{i=0}^d f_i$. For the upper bound, we follow the ideas from the previous case. Again, let ℓ be the index such that $L(x) = (2^t - 1)^x - p_\ell(x)$. Thus, for all $i \in [0 .. \delta]$, we have

$$\begin{aligned} \frac{\text{ext}_i(x)}{L(x)} &= \frac{(2^t - 1)^x - p_i(x)}{(2^t - 1)^x - p_\ell(x)} \\ &\leq \frac{(2^t - 1)^x}{(2^t - 1)^x - p_\ell(x)} \\ &= 1 + \frac{p_\ell(x)}{(2^t - 1)^x - p_\ell(x)} \\ &\leq 1 + \frac{q_\ell(x)}{(2^t - 1)^x - q_\ell(x)} \end{aligned}$$

which is strictly smaller than $1 + 1/(d+1)f_i$ for sufficiently large x . Indeed, as $q_j(x)$ is a polynomial and $f_i \leq 2^n$, we can choose $x \in \mathcal{O}(n)$ such that the exponential term $(2^t - 1)^x$ again dominates the quotient. $\triangleleft$

In both cases, the reduction is pathwidth-preserving as we can add each copy of the gadget $\mathcal{J}$ one after the other to a new copy of a bag containing u from the original path decomposition. Thus, the pathwidth increases by at most the size of $\mathcal{J}$, which is bounded by a constant that depends on the fixed set σ . $\square$

Now we have everything ready to prove the main result of this section by a combination of the previous four reductions. We also refer to the bottom-most box in Figure 10.14 for a visualization.

Lemma 10.9.11. *Fix a non-trivial pair of sets $(\sigma, \rho) \in \mathbb{N}^2$ where σ is cofinite and $\rho = \mathbb{N}$. For all constants d , there is a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#\text{DOMSET}^{(\mathbb{N}_{\geq 1}, \mathbb{N}), \text{HW}_{=1}^{\ddagger}}$ with arity at most d to $(\sigma, \rho)\text{-}\#\text{DOMSET}$.*

Proof. The proof combines the previous reductions in the straightforward way.

With $\mathcal{P} = \{(\mathbb{N}_{\geq 1}, \mathbb{N})\}$, we obtain from Lemmas 10.9.7 and 10.9.8 a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#\text{DOMSET}^{(\mathbb{N}_{\geq 1}, \mathbb{N}), \text{HW}_{=1}^{\ddagger}}$ to $(\sigma, \rho)\text{-}\#\text{DOMSET}^{(\mathbb{N}_{\geq 1}, \mathbb{N}) + (\mathbb{N}, \emptyset)}$, where $(\mathbb{N}, \emptyset)$ is 1-bounded in the target problem.

For the next step, we set $c := 1$, $s := s_{\text{top}} - 1$, and $(\sigma', \rho') = (\sigma, \rho)$. Then, Lemma 10.9.9 yields a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#\text{DOMSET}^{(\mathbb{N}_{\geq 1}, \mathbb{N}) + (\mathbb{N}, \emptyset)}$ where $(\mathbb{N}, \emptyset)$ is 1-bounded to $(\sigma, \rho)\text{-}\#\text{DOMSET}^{(\mathbb{N}, \emptyset)}$ where $(\mathbb{N}, \emptyset)$ is s' -bounded with $s' = s + 1 = s_{\text{top}}$. Note that $(\sigma, \rho)\text{-}\#\text{DOMSET}^{(\sigma, \rho) + (\mathbb{N}, \emptyset)}$ is the same problem as $(\sigma, \rho)\text{-}\#\text{DOMSET}^{(\mathbb{N}, \emptyset)}$.

For the last step, we apply the reduction from Lemma 10.9.10 s' times to remove one $(\mathbb{N}, \emptyset)$ -vertex after the other. As s' only depends on σ and thus, is constant, this gives a pathwidth-preserving reduction from $(\sigma, \rho)\text{-}\#\text{DOMSET}^{(\mathbb{N}, \emptyset)}$ where $(\mathbb{N}, \emptyset)$ is s' -bounded to $(\sigma, \rho)\text{-}\#\text{DOMSET}$ which concludes the proof. $\square$

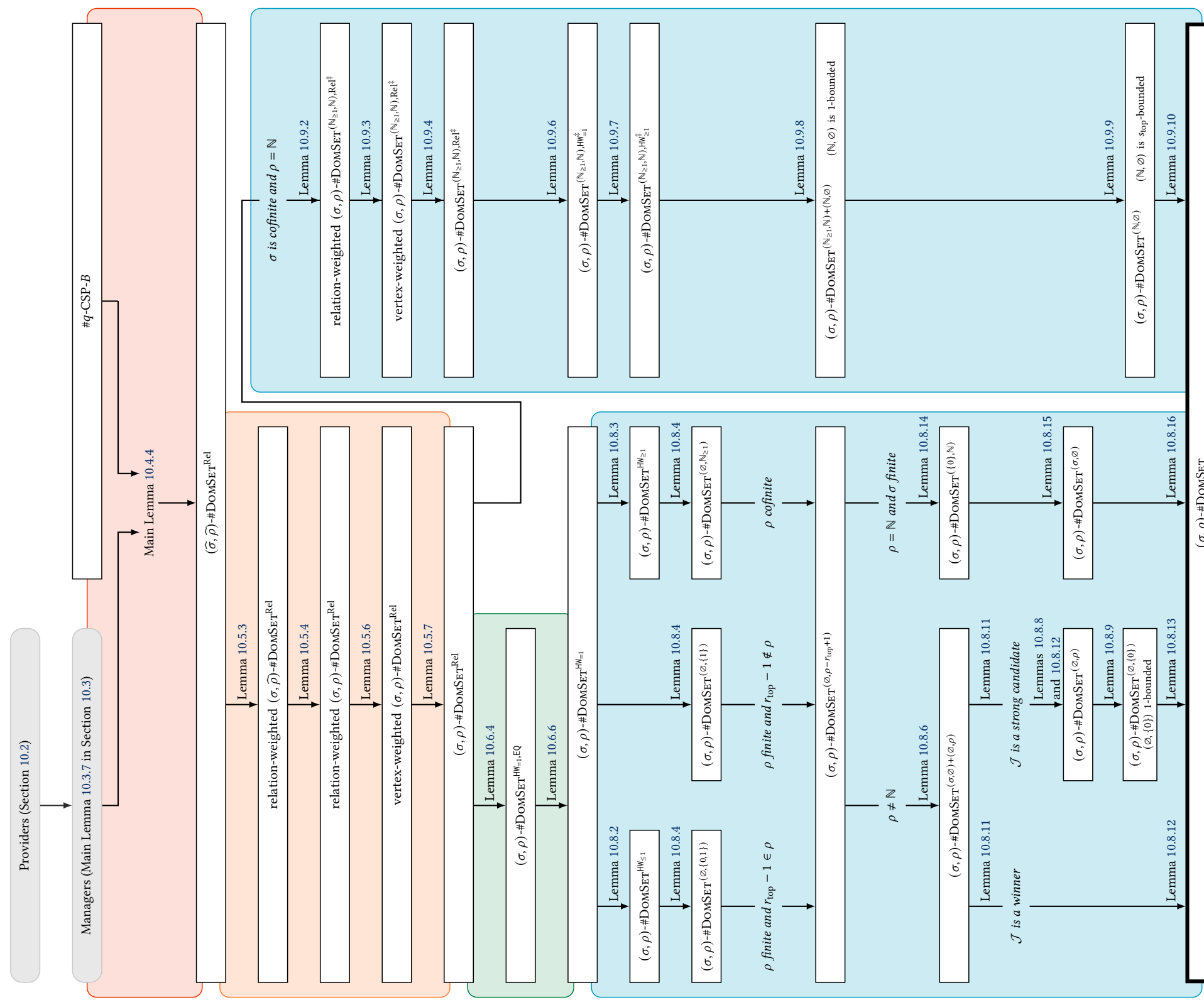
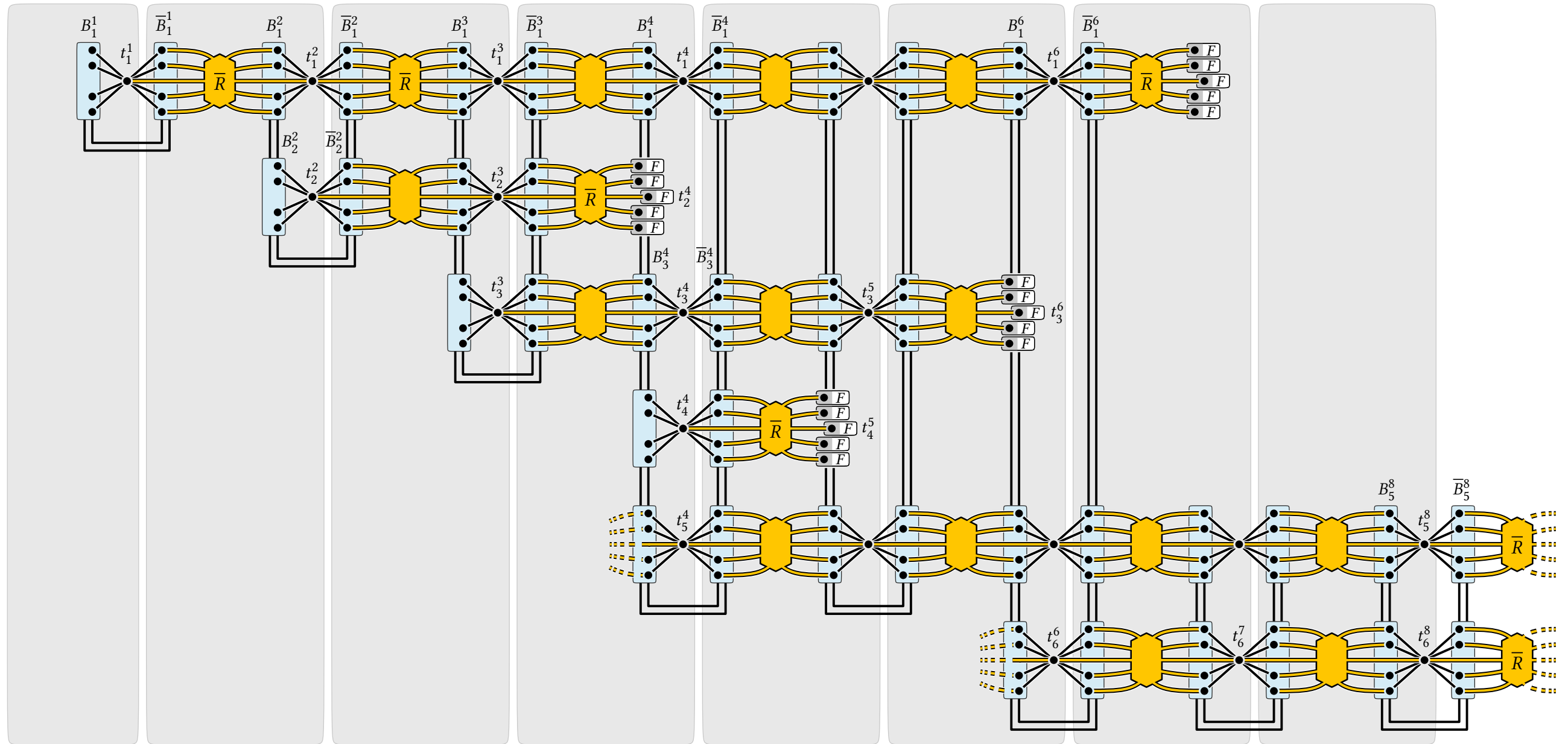


Figure 10.23: The detailed outline of the reductions used to prove the lower bound from Main Theorem III.4 for (σ, ρ) -DomSet in Chapter 10. Each colored box refers to an intermediate result which we prove in the respective section. The sets $\widehat{\sigma}$ and $\widehat{\rho}$ are finite or simple cofinite sets, and the sets σ and ρ are finite or cofinite sets (not necessarily simple cofinite). See Figure 10.1 for a simplified version of this figure.

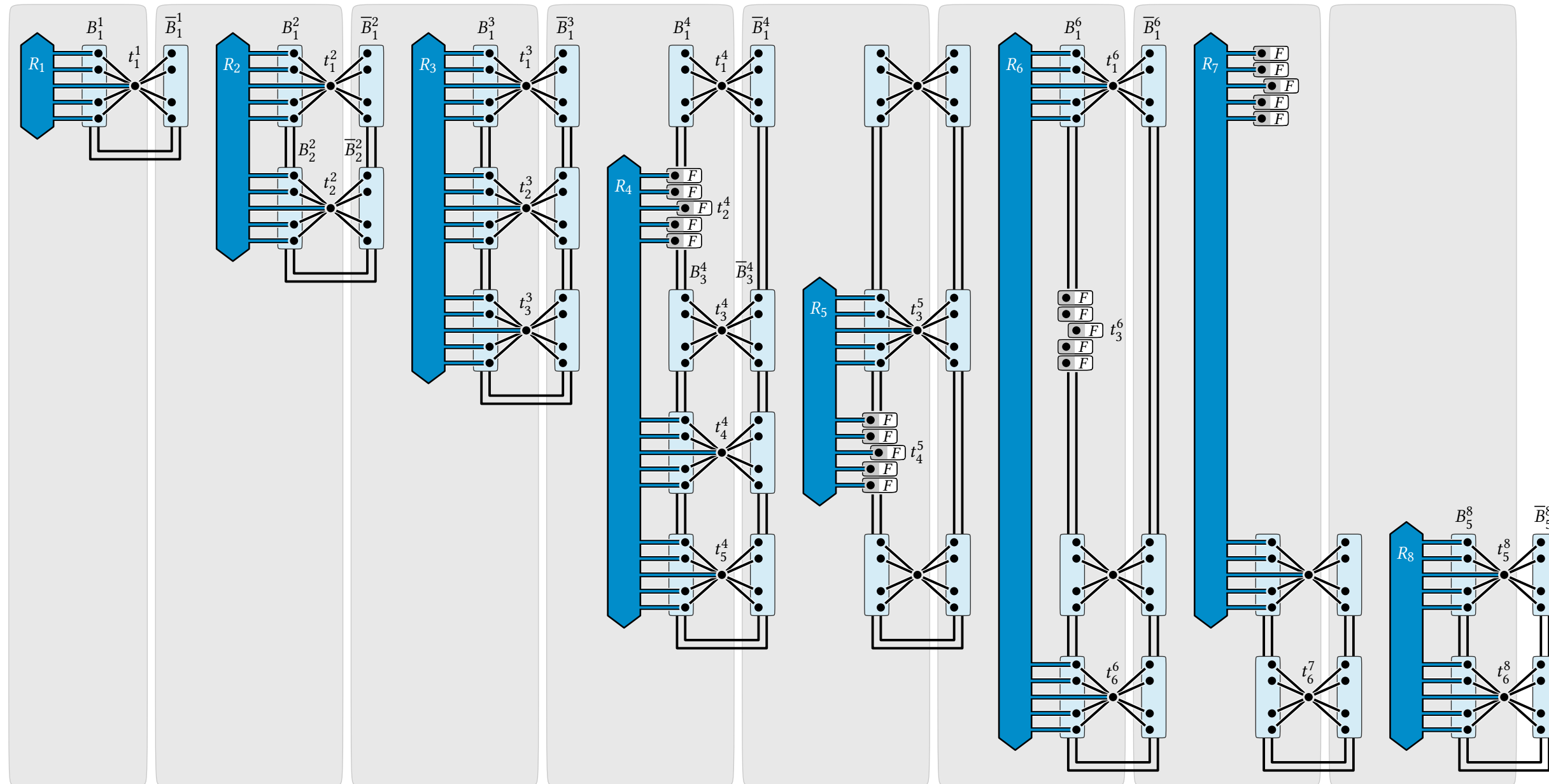


(a) The managers together with the relations that ensure that the states are transferred consistently.

Figure 10.24: An illustration of the reduction to $(\sigma, \rho)\text{-DOMSET}^{\text{Rel}}$ according to the proof of Main Lemma 10.4.3.

We start from the formula ϕ displayed in Figure 7.2 with 4 variables in V_1 and 2 variables in V_2 .

The blocks from the A -manager are shown as small boxes, the relations are shown as hexagons with the portals connected by thicker lines and the $\{\sigma_{\min}, \rho_{\min}\}$ -providers are denoted by F . The large rectangles indicate the bags of ϕ .



(b) The managers together with the relations that correspond to the constraints.

Conclusion

In this thesis we started, based on a parameterization by treewidth, a systematic study of vertex and edge selection problems where degree constraints have to be met. On the algorithmic side, we first presented efficient algorithms to compute convolutions and representative sets. Then, we combined these concepts with problem-specific observations to obtain faster algorithms for Λ -FACTOR and (σ, ρ) -DOMSET when the underlying sets are finite or cofinite.

Additionally, we complemented these algorithmic results by multiple conditional lower bounds. Here we first established a variant of a constraint satisfaction problem (CSP) that allows us to obtain easier but also more robust lower bounds. Based on this CSP, we introduced a framework that can be used to prove lower bounds for general problems in a modularized way. We used this framework to prove lower bounds for Λ -FACTOR and (σ, ρ) -DOMSET which imply that our algorithms are in most cases close to optimal, unless pw-SETH fails.

We refer to Figure 11.1 for a simplified overview of our main results.

Together with the definition of pw-SETH, Lampis [96] established an equivalence between this conjecture and the lower bound for various problems parameterized by pathwidth such as DOMINATING SET. A first next step could extend these results to the problems studied in this thesis.

Q1: Can we prove reductions from (σ, ρ) -DOMSET and Λ -FACTOR to the pathwidth variant of SAT? That is, can we establish an equivalence between pw-SETH and the lower bounds from this thesis?

Improving the Bounds. Especially when trying to prove these equivalences, we would observe that our lower bounds for the decision version (if they exist) do not match the upper bounds when the underlying sets are cofinite. More precisely, for the decision version of (σ, ρ) -DOMSET there is currently no general lower bound when one of the sets is cofinite. For Λ -FACTOR the known lower bounds depend on the *size of the half-induced matching*, while the upper bound is based on the *number of forbidden degrees*.

In Conjecture 4.2.5 we proposed that the size of the half-induced matching implies an upper bound on the size of the representative set which could be used to obtain a better algorithm.

Q2: Can we prove Conjecture 4.2.5 to obtain improved algorithms for Λ -FACTOR when the underlying sets are cofinite?

Q3: Can we find other properties of the sets (besides half-induced matchings) that are related to the size of representative sets?

Although the representative sets might lead to better bounds, the algorithms operate on tree decompositions, that is, they have to deal with join nodes. Although we provide a general

Problem	Upper Bound	Lower Bound	
#ANTI Λ FACTOR $_x$	n^{tw}	no $n^{\text{pw}-\varepsilon}$	
Λ -#FACTOR	$(\text{top } \Lambda + 1)^{\text{tw}}$	no $(\text{top } \Lambda + 1 - \varepsilon)^{\text{pw}}$	
Λ -FACTOR		no $(\text{top } \Lambda + 1 - \varepsilon)^{\text{pw}}$	Λ finite
		no $(h - \varepsilon)^{\text{pw}}$	Λ cofinite with a half-induced matching of size h
ANTI Λ FACTOR $_x$	$(x + 1)^{\mathcal{O}(\text{tw})}$	no $(x + 1 - \varepsilon)^{\text{pw}}$	
(σ, ρ) -#DOMSET	$(c_{\sigma, \rho})^{\text{tw}}$	no $(c_{\sigma, \rho} - \varepsilon)^{\text{pw}}$	
(σ, ρ) -DOMSET	$(c_{\sigma, \rho})^{\text{tw}}$	no $(c_{\sigma, \rho} - \varepsilon)^{\text{pw}}$	σ and ρ are finite
	$(\text{cost}_{\sigma, \rho})^{\mathcal{O}(\text{tw})}$	open	σ or ρ cofinite

Figure 11.1: A simplified overview of the main results ordered from the slowest to the fastest algorithm. We omit the conditions that exclude the trivial and polynomial-time solvable cases. Moreover, we ignore the polynomial-factors of the running time.

procedure that handles these nodes in the classical setting, it remains open to find an efficient algorithm when we consider representative sets. Currently, the only way is a naive computation of the convolution—improvements are likely to be possible.

Q4: Can we improve the algorithm that computes representative sets?

Can we extend this algorithm to efficiently handle the operations needed for join nodes?

Besides these questions for the computationally hard cases of Λ -FACTOR and (σ, ρ) -DOMSET, we might also consider the polynomial-time solvable cases and ask for a solution of a certain fixed size, that is, solve the *exact* decision version of the problem. While it is easy to find *some* solution in these cases, the exact version remains open. In the context of Λ -FACTOR even the special case of MATCHING is open. While Mulmuley, Vazirani, and Vazirani [109] provide a *randomized* algorithm for the problem, there is no derandomization known yet and the question remains open [85, 100, 101].

Q5: What is the complexity of the exact decision version of Λ -FACTOR and (σ, ρ) -DOMSET for those cases where polynomial-time algorithms are known for the decision version?

Beyond Finite and Cofinite. All previous cases consider the variants of Λ -FACTOR and (σ, ρ) -DOMSET where the sets are finite or cofinite. Naturally, we can also consider the problem for other families of sets, that is, sets that are neither finite nor cofinite; we usually refer to such sets as *infinite*. A natural example are the sets of even and odd numbers, but also less structured sets such as the prime numbers or exponential series.

While for (σ, ρ) -DOMSET the special case of even and odd numbers has been studied extensively [25, 26, 59, 73, 74, 124, 125], the case of more general sets is less explored [4, 18,

Recall, our algorithm
solves this variant in
FPT-time only.

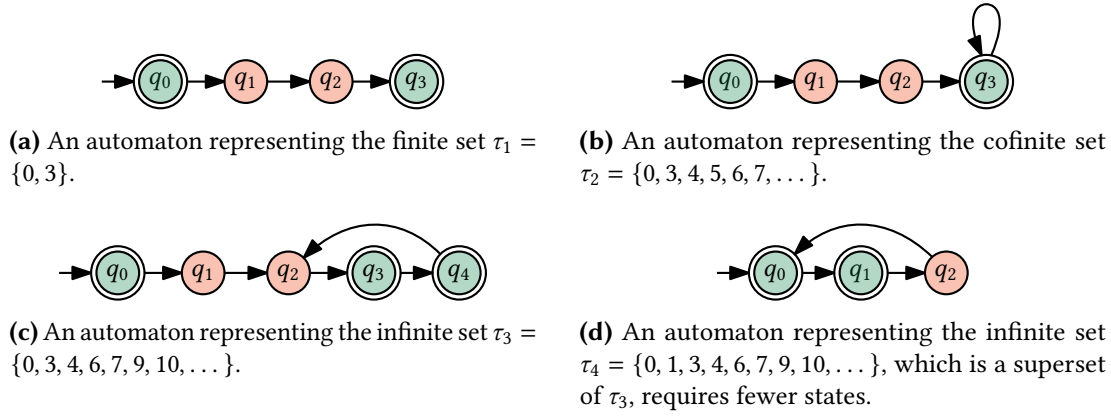


Figure 11.2: Illustration of four deterministic finite automata encoding finite, cofinite, or infinite sets. As the automata do not contain ε -transitions and the alphabet is unary, we omit the symbols from the transitions.

24, 58, 68]. In the parameterized setting even fewer results are known [28, 67]. However, a recent result [70] provides faster algorithms and matching lower bounds when σ and ρ are residue classes with same modulus.

Beyond residue classes very little is known. Chapelle [27, 28] considers (σ, ρ) -DOMSET when σ and ρ are *ultimately periodic* sets, which defines the current boundary of tractability.

A set $\tau \subseteq \mathbb{N}$ is ultimately periodic if there is a (deterministic) finite automaton $\mathcal{A}$ over a unary-alphabet and the length of the accepted words of $\mathcal{A}$ is equal to τ . We refer to Figure 11.2 for an illustration of such automata.

Suppose a deterministic finite automaton (DFA) with q_σ states generates σ , and suppose a DFA with q_ρ states generates ρ . Then, (σ, ρ) -DOMSET is fixed-parameter tractable parameterized by treewidth and $q_\sigma + q_\rho$.

Theorem 11.1.1 ([27, Theorem 2] and [28, Théorème 3.2.1]). *Denote by σ and ρ two ultimately periodic sets that can be enumerated by DFAs with q_σ and q_ρ states, respectively.*

There is a small polynomial $P(\cdot, \cdot)$ such that we can solve (σ, ρ) -DOMSET on a graph G that is given with a tree decomposition of width w in time $(P(q_\sigma, q_\rho))^w \cdot |G|^{\mathcal{O}(1)}$.

The idea of the algorithm is actually quite simple. As we know that there are automata for the sets σ and ρ , every state of a vertex with respect to a (partial) solution can be represented by a state of these automata. Then, the dynamic program stores the automaton-states instead of the vertex-states. Moreover, the dynamic program combines and modifies these states along the various nodes of the tree decomposition. We observe that for finite and cofinite sets, the number of vertex-states and the number of automaton-states precisely coincide.

As combining the states of the automaton is more difficult than combining numbers, which represent the vertex-states, and the convolution technique for efficiently handling join nodes are missing, the algorithm by Chapelle is presumably not optimal.

Q6: Can we improve the general algorithm by Chapelle [27, 28] to obtain a linear dependence on the treewidth?

The results in [70] are a follow-up of the original results for (σ, ρ) -DOMSET.

Even if we would manage to improve the convolution techniques to handle the join nodes, the algorithm would not be optimal in all cases. Indeed, our main algorithmic result for (σ, ρ) -DOMSET proves that significant improvements compared to the naive bound are possible. However, it is open how we could relate these improvements to the automata representation of the states. We observe that such a relation (if it exists) might be quite complicated; the cross product of the automata would not have a reduced number of states, but our results show that the number of states that actually correspond to partial solutions can be much smaller.

Instead of improving the general algorithm, one might also ask whether there are even more general sets that allow FPT-algorithms parameterized by treewidth. As a limitation, Chapelle provides a hardness results, when the set σ has large gaps that are not of constant size.

Theorem 11.1.2 ([27, Theorem 1] and [28, Théorème 3.3.1]). *Denote by $\sigma \subseteq \mathbb{N}$ a set with $\min \sigma \geq 1$ and arbitrarily large gaps between two consecutive elements (such that a gap of length t is at distance $\text{poly}(t)$ in σ), and denote by ρ a cofinite set with $\min \rho \geq 2$. Then, (σ, ρ) -DOMSET is $W[1]$ -hard parameterized by the treewidth of the input graph.*

Observe that this result provides another problem that is not FPT parameterized by treewidth. We have seen a similar result already for $\#ANTIFACTOR_x$ in Main Theorem II.6.

However, we also observe that this hardness result does not complement the algorithmic result from Theorem 11.1.1, that is, both results do *not* provide a dichotomy. Indeed, the set $\sigma = \mathbb{N}_{\geq 1} \setminus \{2^i \mid i \in \mathbb{N}\}$ has no arbitrary large gaps but still, is not ultimately periodic.

Q7: Which conditions of the sets σ and ρ provide a dichotomy for (σ, ρ) -DOMSET between the FPT cases and the $W[1]$ -hard cases parameterized by treewidth?

Naturally, we can also ask these questions for Λ -FACTOR. More precisely, can we extend the existing algorithm to also work with ultimately periodic sets? Surprisingly, when we consider the set $\Lambda = \mathbb{N}_{\geq 1} \setminus \{2^i \mid i \in \mathbb{N}\}$ (for which the complexity in the case of (σ, ρ) -DOMSET is unknown), the problem Λ -FACTOR can be solved in polynomial-time by the existing algorithm of Cornuéjols [30] as all gaps have size 1.

Q8: Which conditions on the set Λ provide a dichotomy for Λ -FACTOR between the FPT cases and the potential $W[1]$ -hard cases parameterized by treewidth?

Instead of considering different sets for Λ -FACTOR we can also generalize the problem itself.

Understanding Holants. An extension of Λ -FACTOR or rather its counting version leads to the *Holant framework* [19]. To prove the lower bound results of Λ -#FACTOR, we already considered specific versions of this problem to formalize the concept of weighted edges and weighted relations (see Section 7.7). In general, the Holant problem extends GENFAC such that

- we allow weighted edges which contribute by the product of their weight to the solution,
- each of the vertices can have a different (possibly weighted) function assigned—in the context of the Holant framework this is usually referred to as *signature*—and

- the signatures do not necessarily have to be symmetric, that is, they do not only depend on the *number* of selected edges but on the precise positions of the selected edges.

Due to the close relation to the counting version of constraint satisfaction problems, the MATCHING problem, or coloring problems, just to name a few examples, the problem has received significant attention [20–22, 71, 79, 92, 99].

We observe that most results for the Holant framework consider the non-parameterized setting (but see also [3] for recent results in the parameterized setting). Despite the fact that our results are specifically tailored for the context of a parameterization by treewidth, they can also be applied in other contexts as well. Although we mainly consider unweighted problems, our intermediate results prove that we can efficiently simulate weights (if there are not too many of them). Hence, we actually capture the Holant framework in full generality.

Phrased in the language of the Holant framework, we prove that we can replace arbitrary signatures by gates that consist of much simpler signatures. That is, our results possibly extend the list of (intractable) classes as they show that the presumably easier cases can simulate the harder cases.

A natural question would be to understand how our results could help to improve the dichotomy results for the Holant framework or how they could provide further insights.

From Treewidth to Hubs. When we design the conditional lower bounds, we prove them under the assumption of pw-SETH. Although this conjecture is less likely to fail (compared to SETH), we might also ask whether it is the “correct” hypothesis.

Precisely such a question inspired a recent work by Esmer, Focke, Marx, and Rzążewski [47]. They introduce the structural measure of (σ, δ) -hubs and provide lower bounds for several problems based on the size of these hubs.

For two non-negative integers σ and δ , a vertex set W of some graph G is a (σ, δ) -hub if every connected component C of $G - W$ contains at most σ vertices and each such component C is adjacent to at most δ vertices in W . Then, the size of these (σ, δ) -hubs serves as the parameter. Intuitively, the size of the hub determines the size of a central component that is loosely connected to the other parts of the graph.

In several cases the lower bounds based on (σ, δ) -hubs coincide with the result for a parameterization by treewidth (or pathwidth), but in other cases (for example for DOMINATING SET) a gap remains [47].

We observe that for the lower bounds proven under pw-SETH, the application of (σ, δ) -hubs seems inappropriate at a first glance. Indeed, the arrangement of the vertices and gadgets along a path does not give a central component that connects the other parts. However, one can also think of a construction, where we have a central component that encodes the assignments and the other parts are all connected to this central gadget to encode the clauses. In this case it might be possible to obtain lower bounds when parameterizing by the size of a given (σ, δ) -hub.

We use the original name although, this gives two meanings to σ in the context of (σ, ρ) -DOMSET.

Q9: Can we prove matching upper and lower bounds for Λ -FACTOR and (σ, ρ) -DOMSET parameterized by the size of (σ, δ) -hubs?

Beyond Λ -FACTOR and (σ, ρ) -DOMSET. In summary, we have shown some techniques that can lead to faster algorithms and proposed a framework that potentially helps to systematically prove lower bounds. We leave it to the reader as a (most likely not easy) exercise to find further examples where these techniques can be applied such that we obtain a more structured picture of the landscape of graph selection problems in the parameterized regime. ■

Part IV

Appendix

List of Figures

1.1	Primal Graph of a Formula	17
1.2	Overview of the Proofs of the Lower Bounds	20
1.3	Idea of the Main Construction for the Lower Bounds	23
1.4	Construction to Realize an Arbitrary Relation	24
1.5	Organization of This Thesis	27
4.1	Construction from Lemma 4.1.15	91
4.2	Illustration of a Half-Induced Matching from Definition 4.2.1	97
7.1	Overview of the Proofs of the Main Results in Chapter 7	129
7.2	Reduction from q -CSP- B to Λ -FACTOR ^{Rel} from Main Lemma 7.2.1	143
7.3	Reduction from Finite to Cofinite Sets from Main Lemma 7.3.1	149
7.4	Overview of the Proof of Main Lemma 7.4.6 in Section 7.4	153
7.5	Reduction to Allow Arbitrary Cofinite Sets from Lemma 7.4.3	157
7.6	Simulation of Weighted Relations from Lemma 7.4.4	159
7.7	Simulation of Weighted Edges from Lemma 7.4.5	161
7.8	Realization of a Relation from Main Lemma 7.5.3	166
7.9	Realization of Auxiliary Relations from Lemma 7.6.1	169
7.10	Realization of EQ_1 from Lemma 7.6.1	171
7.11	Inductive Realization of $HW_{=1}^{(k)}$ from Lemma 7.6.4	172
7.12	Realization of $HW_{=1}^{(3)}$ Together with $HW_{=1}^{(2)}$ from Lemma 7.6.5	173
7.13	Overview of the Proof of Main Lemma 7.7.14 in Section 7.7	178
7.14	Realization of the Auxiliary Gadget from Lemma 7.7.9	182
7.15	Gadget H_ℓ from Claims 7.7.10.1 and 7.7.10.3	185
7.16	Gadget H_ℓ from Claims 7.7.10.2 and 7.7.10.4	186
7.17	Gadget H_ℓ from Lemma 7.7.13	192
7.18	High-Level Construction from Lemma 7.8.5	198
7.19	Main Gadget from Lemma 7.8.5	201
7.20	Variable Gadget from Lemma 7.9.3	207
7.21	Clause Gadget from Lemma 7.9.3	208
7.22	Gadget to Obtain Equal Degree from Claim 7.9.4.1	212
8.1	Compatible Vector for a Graph with Portals from Definition 8.1.9	223
9.1	An Example for DOMINATING SET with Many Extensions	228
9.2	Intuition for Lemma 9.1.2	231
10.1	Overview of the Proofs of the Main Results in Chapter 10	265

List of Figures

10.2	Illustration of the Concept of a Manager from Definition 10.3.1	269
10.3	Idea of the Construction for the Intermediate Lower Bound	272
10.4	Construction of $\{\sigma_s, \rho_r\}$ -Providers from Lemmas 10.2.1 and 10.2.2	285
10.5	Construction of the $\mathbb{R}$ -Provider from Lemma 10.2.3	286
10.6	Construction of the Provider from Lemma 10.2.6	289
10.7	Construction of the Manager from Lemma 10.3.8	303
10.8	Overview of the Proof of Main Lemma 10.5.8 in Section 10.5	318
10.9	Realization of an Arbitrary Relation from Lemma 10.6.3	327
10.10	Realization of Equality from Lemma 10.6.5	329
10.11	Gadgets from Claim 10.7.1.1	332
10.12	Construction of the $\{\sigma_{\text{stop}}\}$ -Provider from Lemma 10.7.1	333
10.13	Realization of $\text{HW}_{=1}^{(k)}$ from Lemma 10.7.2	334
10.14	Overview of the Proof of Main Lemma 10.8.1 in Section 10.8	338
10.15	Construction to Simulate $\text{HW}_{=1}$ with $\text{HW}_{\leq 1}$ from Lemma 10.8.2	341
10.16	Construction to Simulate $\text{HW}_{=1}$ with $\text{HW}_{\geq 1}$ from Lemma 10.8.3	342
10.17	Auxiliary Gadgets from Lemma 10.8.11	348
10.18	Gadget to Simulate $(\emptyset, \{0\})$ -Vertices from Lemma 10.8.13	355
10.19	Overview of the Proof of Main Lemma 10.9.1 in Section 10.9	370
10.20	Auxiliary Gadget from Lemma 10.9.2	373
10.21	Gadget to Simulate $\text{HW}_{\geq 1}^{\ddagger}$ from Lemma 10.9.8	382
10.22	Gadgets to Simulate $(\mathbb{N}, \emptyset)$ -Vertices from Lemma 10.9.10	384
10.23	Detailed Overview of the Proofs of the Main Results in Chapter 10	389
10.24	Reduction from q -CSP- B to (σ, ρ) - $\text{DOMSET}^{\text{Rel}}$ from Main Lemma 10.4.3	390
11.1	Simplified Overview of the Main Results	394
11.2	Automata Encoding Sets of Numbers	395

Bibliography

- [1] Akanksha Agrawal, Pallavi Jain, Lawqueen Kanesh, and Saket Saurabh. Parameterized complexity of conflict-free matchings and paths. *Algorithmica*, 82(7):1939–1965, 2020. doi:[10.1007/s00453-020-00681-y](https://doi.org/10.1007/s00453-020-00681-y).
- [2] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004. ISSN 0003486X. URL <http://www.jstor.org/stable/3597229>.
- [3] Panagiotis Aivasiliotis, Andreas Göbel, Marc Roth, and Johannes Schmitt. Parameterised Holant problems. In Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis, editors, *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025, July 8-11, 2025, Aarhus, Denmark*, volume 334 of *LIPICs*, pages 7:1–7:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. doi:[10.4230/LIPICS.ICALP.2025.7](https://doi.org/10.4230/LIPICS.ICALP.2025.7).
- [4] Jochen Alber, Hans L. Bodlaender, Henning Fernau, Ton Kloks, and Rolf Niedermeier. Fixed parameter algorithms for DOMINATING SET and related problems on planar graphs. *Algorithmica*, 33(4):461–493, 2002. doi:[10.1007/S00453-001-0116-5](https://doi.org/10.1007/S00453-001-0116-5).
- [5] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987. doi:[10.1137/0608024](https://doi.org/10.1137/0608024).
- [6] Ashwin Arulselvan, Ágnes Cseh, Martin Groß, David F. Manlove, and Jannik Matuschke. Matchings with lower quotas: Algorithms and complexity. *Algorithmica*, 80(1):185–208, 2018. doi:[10.1007/s00453-016-0252-6](https://doi.org/10.1007/s00453-016-0252-6).
- [7] Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, and Yota Otachi. Grundy distinguishes treewidth from pathwidth. *SIAM J. Discret. Math.*, 36(3):1761–1787, 2022. doi:[10.1137/20M1385779](https://doi.org/10.1137/20M1385779).
- [8] Michael A. Bennett, Greg Martin, Kevin O’Bryant, and Andrew Rechnitzer. Explicit bounds for primes in arithmetic progressions. *Illinois J. Math.*, 62(1-4):427–532, 2018. ISSN 0019-2082. doi:[10.1215/ijm/1552442669](https://doi.org/10.1215/ijm/1552442669).
- [9] Claude Berge. *Graphs and Hypergraphs*. Elsevier Science Ltd., United Kingdom, 1985. ISBN 0-7204-0479-7.
- [10] Umberto Bertelè and Francesco Brioschi. On non-serial dynamic programming. *J. Comb. Theory A*, 14(2):137–148, 1973. doi:[10.1016/0097-3165\(73\)90016-2](https://doi.org/10.1016/0097-3165(73)90016-2).

- [11] Hans L. Bodlaender. On disjoint cycles. *Int. J. Found. Comput. Sci.*, 5(1):59–68, 1994. doi:[10.1142/S0129054194000049](https://doi.org/10.1142/S0129054194000049).
- [12] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996. doi:[10.1137/S0097539793251219](https://doi.org/10.1137/S0097539793251219).
- [13] Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:[10.1016/j.ic.2014.12.008](https://doi.org/10.1016/j.ic.2014.12.008).
- [14] Édouard Bonnet, Nick Brettell, O-jeong Kwon, and Dániel Marx. Generalized feedback vertex set problems on bounded-treewidth graphs: Chordality is the key to single-exponential parameterized algorithms. *Algorithmica*, 81(10):3890–3935, 2019. doi:[10.1007/s00453-019-00579-4](https://doi.org/10.1007/s00453-019-00579-4).
- [15] Magnus Bordewich, Martin E. Dyer, and Marek Karpinski. Path coupling using stopping times and counting independent sets and colorings in hypergraphs. *Random Struct. Algorithms*, 32(3):375–399, 2008. doi:[10.1002/rsa.20204](https://doi.org/10.1002/rsa.20204).
- [16] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24–26, 2016, Aarhus, Denmark*, volume 63 of *LIPIcs*, pages 8:1–8:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:[10.4230/LIPIcs.IPEC.2016.8](https://doi.org/10.4230/LIPIcs.IPEC.2016.8).
- [17] Russ Bubley and Martin E. Dyer. Graph orientations with no sink and an approximation for a hard case of #SAT. In Michael E. Saks, editor, *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, 5–7 January 1997, New Orleans, Louisiana, USA*, pages 248–257. ACM/SIAM, 1997. URL <http://dl.acm.org/citation.cfm?id=314161.314263>.
- [18] Binh-Minh Bui-Xuan, Jan Arne Telle, and Martin Vatshelle. Fast dynamic programming for locally checkable vertex subset and vertex partitioning problems. *Theor. Comput. Sci.*, 511:66–76, 2013. doi:[10.1016/j.tcs.2013.01.009](https://doi.org/10.1016/j.tcs.2013.01.009).
- [19] Jin-yi Cai, Pinyan Lu, and Mingji Xia. Holographic algorithms by Fibonacci gates and holographic reductions for hardness. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25–28, 2008, Philadelphia, PA, USA*, pages 644–653. IEEE Computer Society, 2008. doi:[10.1109/FOCS.2008.34](https://doi.org/10.1109/FOCS.2008.34).
- [20] Jin-Yi Cai, Pinyan Lu, and Mingji Xia. A computational proof of complexity of some restricted counting problems. *Theor. Comput. Sci.*, 412(23):2468–2485, 2011. doi:[10.1016/j.tcs.2010.10.039](https://doi.org/10.1016/j.tcs.2010.10.039).
- [21] Jin-Yi Cai, Sangxia Huang, and Pinyan Lu. From Holant to #CSP and back: Dichotomy for Holant^c problems. *Algorithmica*, 64(3):511–533, 2012. doi:[10.1007/S00453-012-9626-6](https://doi.org/10.1007/S00453-012-9626-6).

- [22] Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Dichotomy for Holant* problems on the boolean domain. *Theory Comput. Syst.*, 64(8):1362–1391, 2020. doi:[10.1007/S00224-020-09983-8](https://doi.org/10.1007/S00224-020-09983-8).
- [23] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, volume 5917 of *Lecture Notes in Computer Science*, pages 75–85. Springer, 2009. doi:[10.1007/978-3-642-11269-0_6](https://doi.org/10.1007/978-3-642-11269-0_6).
- [24] Yair Caro and Michael S. Jacobson. On non- $z(\bmod k)$ dominating sets. *Discuss. Math. Graph Theory*, 23(1):189–199, 2003. doi:[10.7151/DMGT.1195](https://doi.org/10.7151/DMGT.1195).
- [25] Yair Caro, John L. Goldwasser, and William Klostermeyer. Odd and residue domination numbers of a graph. *Discuss. Math. Graph Theory*, 21(1):119–136, 2001. doi:[10.7151/DMGT.1137](https://doi.org/10.7151/DMGT.1137).
- [26] David Cattanéo and Simon Perdrix. Parameterized complexity of weak odd domination problems. In Leszek Gasieniec and Frank Wolter, editors, *Fundamentals of Computation Theory - 19th International Symposium, FCT 2013, Liverpool, UK, August 19-21, 2013. Proceedings*, volume 8070 of *Lecture Notes in Computer Science*, pages 107–120. Springer, 2013. doi:[10.1007/978-3-642-40164-0_13](https://doi.org/10.1007/978-3-642-40164-0_13).
- [27] Mathieu Chapelle. Parameterized complexity of generalized domination problems on bounded tree-width graphs. *CoRR*, abs/1004.2642v5, 2010. URL <http://arxiv.org/abs/1004.2642v5>.
- [28] Mathieu Chapelle. *Décompositions de graphes: quelques limites et obstructions. (Graphs decompositions: some limites and obstructions)*. PhD thesis, University of Orléans, France, 2011. URL <https://tel.archives-ouvertes.fr/tel-00659666>.
- [29] Dmitry Chistikov, Rupak Majumdar, and Philipp Schepper. Subcubic certificates for CFL reachability. *Proc. ACM Program. Lang.*, 6(POPL):1–29, 2022. doi:[10.1145/3498702](https://doi.org/10.1145/3498702).
- [30] Gérard Cornuéjols. General factors of graphs. *J. Comb. Theory, Ser. B*, 45(2):185–198, 1988. doi:[10.1016/0095-8956\(88\)90068-8](https://doi.org/10.1016/0095-8956(88)90068-8).
- [31] Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. doi:[10.1016/0890-5401\(90\)90043-H](https://doi.org/10.1016/0890-5401(90)90043-H).
- [32] Radu Curticapean. *The simple, little and slow things count: on parameterized counting complexity*. PhD thesis, Saarland University, 2015. doi:[10.22028/D291-26612](https://doi.org/10.22028/D291-26612).
- [33] Radu Curticapean. Block interpolation: A framework for tight exponential-time counting complexity. *Inf. Comput.*, 261:265–280, 2018. doi:[10.1016/j.ic.2018.02.008](https://doi.org/10.1016/j.ic.2018.02.008).
- [34] Radu Curticapean and Dániel Marx. Complexity of counting subgraphs: Only the boundedness of the vertex-cover number counts. In *55th IEEE Annual Symposium on*

- Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 130–139. IEEE Computer Society, 2014. doi:[10.1109/FOCS.2014.22](https://doi.org/10.1109/FOCS.2014.22).
- [35] Radu Curticapean and Dániel Marx. Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1650–1669. SIAM, 2016. doi:[10.1137/1.9781611974331.ch113](https://doi.org/10.1137/1.9781611974331.ch113).
 - [36] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. ISBN 978-3-319-21274-6. doi:[10.1007/978-3-319-21275-3](https://doi.org/10.1007/978-3-319-21275-3).
 - [37] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. *ACM Trans. Algorithms*, 18(2):17:1–17:31, 2022. doi:[10.1145/3506707](https://doi.org/10.1145/3506707).
 - [38] Víctor Dalmau and Daniel K. Ford. Generalized satisfiability with limited occurrences per variable: A study through delta-matroid parity. In Branislav Rován and Peter Vojtás, editors, *Mathematical Foundations of Computer Science 2003, 28th International Symposium, MFCS 2003, Bratislava, Slovakia, August 25-29, 2003, Proceedings*, volume 2747 of *Lecture Notes in Computer Science*, pages 358–367. Springer, 2003. doi:[10.1007/978-3-540-45138-9_30](https://doi.org/10.1007/978-3-540-45138-9_30).
 - [39] Julian Dörfler, Marc Roth, Johannes Schmitt, and Philip Wellnitz. Counting induced subgraphs: An algebraic approach to $\#W[1]$ -hardness. *Algorithmica*, 84(2):379–404, 2022. doi:[10.1007/S00453-021-00894-9](https://doi.org/10.1007/S00453-021-00894-9).
 - [40] Simon Döring, Dániel Marx, and Philip Wellnitz. From graph properties to graph parameters: Tight bounds for counting on small subgraphs. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 3637–3676. SIAM, 2025. doi:[10.1137/1.9781611978322.121](https://doi.org/10.1137/1.9781611978322.121).
 - [41] Szymon Dudycz and Katarzyna Paluch. Optimal general matchings. In Andreas Brandstädt, Ekkehard Köhler, and Klaus Meer, editors, *Graph-Theoretic Concepts in Computer Science - 44th International Workshop, WG 2018, Cottbus, Germany, June 27-29, 2018, Proceedings*, volume 11159 of *Lecture Notes in Computer Science*, pages 176–189. Springer, 2018. doi:[10.1007/978-3-030-00256-5_15](https://doi.org/10.1007/978-3-030-00256-5_15).
 - [42] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. doi:[10.4153/CJM-1965-045-4](https://doi.org/10.4153/CJM-1965-045-4).
 - [43] Eduard Eiben and Iyad Kanj. A colored path problem and its applications. *ACM Trans. Algorithms*, 16(4):47:1–47:48, 2020. doi:[10.1145/3396573](https://doi.org/10.1145/3396573).

- [44] Paul Erdős. Some combinatorial, geometric and set theoretic problems in measure theory. In Dietrich Kölzow and Dorothy Maharam-Stone, editors, *Measure Theory Oberwolfach 1983*, pages 321–327, Berlin, Heidelberg, 1984. Springer Berlin Heidelberg. ISBN 978-3-540-39069-5.
- [45] Paul Erdős and Lajos Pósa. On independent circuits contained in a graph. *Canad. J. Math.*, 17:347–352, 1965. ISSN 0008-414X. doi:[10.4153/CJM-1965-035-8](https://doi.org/10.4153/CJM-1965-035-8).
- [46] Barış Can Esmer, Ariel Kulik, Dániel Marx, Philipp Schepper, and Karol Węgrzycki. Computing generalized convolutions faster than brute force. In Holger Dell and Jesper Nederlof, editors, *17th International Symposium on Parameterized and Exact Computation, IPEC 2022, September 7-9, 2022, Potsdam, Germany*, volume 249 of *LIPICs*, pages 12:1–12:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:[10.4230/LIPICs.IPEC.2022.12](https://doi.org/10.4230/LIPICs.IPEC.2022.12).
- [47] Barış Can Esmer, Jacob Focke, Dániel Marx, and Paweł Rzażewski. Fundamental problems on bounded-treewidth graphs: The real source of hardness. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPICs*, pages 34:1–34:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:[10.4230/LIPICs.ICALP.2024.34](https://doi.org/10.4230/LIPICs.ICALP.2024.34).
- [48] Barış Can Esmer, Ariel Kulik, Dániel Marx, Philipp Schepper, and Karol Węgrzycki. Computing generalized convolutions faster than brute force. *Algorithmica*, 86(1):334–366, 2024. doi:[10.1007/S00453-023-01176-2](https://doi.org/10.1007/S00453-023-01176-2).
- [49] Grzegorz Fabiański, Michał Pilipczuk, Sebastian Siebertz, and Szymon Toruńczyk. Progressive algorithms for domination and independence. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, volume 126 of *LIPICs*, pages 27:1–27:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:[10.4230/LIPICs.STACS.2019.27](https://doi.org/10.4230/LIPICs.STACS.2019.27).
- [50] Tomás Feder. Fanout limitations on constraint systems. *Theor. Comput. Sci.*, 255(1-2): 281–293, 2001. doi:[10.1016/S0304-3975\(99\)00288-1](https://doi.org/10.1016/S0304-3975(99)00288-1).
- [51] Jörg Flum and Martin Grohe. The parameterized complexity of counting problems. *SIAM J. Comput.*, 33(4):892–922, 2004. doi:[10.1137/S0097539703427203](https://doi.org/10.1137/S0097539703427203).
- [52] Jacob Focke and Marc Roth. Counting small induced subgraphs with hereditary properties. *SIAM J. Comput.*, 53(2):189–220, 2024. doi:[10.1137/22M1512211](https://doi.org/10.1137/22M1512211).
- [53] Jacob Focke, Dániel Marx, Fionn Mc Inerney, Daniel Neuen, Govind S. Sankar, Philipp Schepper, and Philip Wellnitz. Tight complexity bounds for counting generalized dominating sets in bounded-treewidth graphs. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 3664–3683. SIAM, 2023. doi:[10.1137/1.9781611977554.CH140](https://doi.org/10.1137/1.9781611977554.CH140).

- [54] Jacob Focke, Fabian Frei, Shaohua Li, Dániel Marx, Philipp Schepper, Roohani Sharma, and Karol Węgrzycki. Hitting meets packing: How hard can it be? In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, September 2-4, 2024, Royal Holloway, London, United Kingdom*, volume 308 of *LIPICs*, pages 55:1–55:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:[10.4230/LIPICs.ESA.2024.55](https://doi.org/10.4230/LIPICs.ESA.2024.55).
- [55] Jacob Focke, Dániel Marx, and Paweł Rzażewski. Counting list homomorphisms from graphs of bounded treewidth: Tight complexity bounds. *ACM Trans. Algorithms*, 20(2): 11, 2024. doi:[10.1145/3640814](https://doi.org/10.1145/3640814).
- [56] Jacob Focke, Dániel Marx, Fionn Mc Inerney, Daniel Neuen, Govind S. Sankar, Philipp Schepper, and Philip Wellnitz. Tight complexity bounds for counting generalized dominating sets in bounded-treewidth graphs—Part I: Algorithmic results. *ACM Trans. Algorithms*, 21(3), July 2025. ISSN 1549-6325. doi:[10.1145/3731452](https://doi.org/10.1145/3731452).
- [57] Jacob Focke, Dániel Marx, Fionn Mc Inerney, Daniel Neuen, Govind S. Sankar, Philipp Schepper, and Philip Wellnitz. Tight complexity bounds for counting generalized dominating sets in bounded-treewidth graphs Part II: Hardness results. *ACM Trans. Comput. Theory*, 17(2), April 2025. ISSN 1942-3454. doi:[10.1145/3708509](https://doi.org/10.1145/3708509).
- [58] Fedor V. Fomin and Dimitrios M. Thilikos. Dominating sets in planar graphs: Branch-width and exponential speed-up. *SIAM J. Comput.*, 36(2):281–309, 2006. doi:[10.1137/S0097539702419649](https://doi.org/10.1137/S0097539702419649).
- [59] Fedor V. Fomin, Petr A. Golovach, Jan Kratochvíl, Dieter Kratsch, and Mathieu Liedloff. Sort and search: Exact algorithms for generalized domination. *Inf. Process. Lett.*, 109(14): 795–798, 2009. doi:[10.1016/J.IPL.2009.03.023](https://doi.org/10.1016/J.IPL.2009.03.023).
- [60] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:[10.1145/2886094](https://doi.org/10.1145/2886094).
- [61] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Representative families of product families. *ACM Trans. Algorithms*, 13(3):36:1–36:29, 2017. doi:[10.1145/3039243](https://doi.org/10.1145/3039243).
- [62] Florent Foucaud, Esther Galby, Liana Khazaliya, Shaohua Li, Fionn Mc Inerney, Roohani Sharma, and Prafullkumar Tale. Problems in NP can admit double-exponential lower bounds when parameterized by treewidth or vertex cover. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPICs*, pages 66:1–66:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:[10.4230/LIPICs.ICALP.2024.66](https://doi.org/10.4230/LIPICs.ICALP.2024.66).
- [63] Esther Galby, Dániel Marx, Philipp Schepper, Roohani Sharma, and Prafullkumar Tale. Domination and cut problems on chordal graphs with bounded leafage. In Holger

- Dell and Jesper Nederlof, editors, *17th International Symposium on Parameterized and Exact Computation, IPEC 2022, September 7-9, 2022, Potsdam, Germany*, volume 249 of *LIPICs*, pages 14:1–14:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:[10.4230/LIPICs.IPEC.2022.14](https://doi.org/10.4230/LIPICs.IPEC.2022.14).
- [64] Esther Galby, Dániel Marx, Philipp Schepper, Roohani Sharma, and Prafullkumar Tale. Parameterized complexity of weighted multicut in trees. In Michael A. Bekos and Michael Kaufmann, editors, *Graph-Theoretic Concepts in Computer Science - 48th International Workshop, WG 2022, Tübingen, Germany, June 22-24, 2022, Revised Selected Papers*, volume 13453 of *Lecture Notes in Computer Science*, pages 257–270. Springer, 2022. doi:[10.1007/978-3-031-15914-5_19](https://doi.org/10.1007/978-3-031-15914-5_19).
- [65] Esther Galby, Dániel Marx, Philipp Schepper, Roohani Sharma, and Prafullkumar Tale. Parameterized complexity of multicut in weighted trees. *Theor. Comput. Sci.*, 978:114174, 2023. doi:[10.1016/J.TCS.2023.114174](https://doi.org/10.1016/J.TCS.2023.114174).
- [66] Esther Galby, Dániel Marx, Philipp Schepper, Roohani Sharma, and Prafullkumar Tale. Domination and cut problems on chordal graphs with bounded leafage. *Algorithmica*, 86(5):1428–1474, 2024. doi:[10.1007/S00453-023-01196-Y](https://doi.org/10.1007/S00453-023-01196-Y).
- [67] Elisabeth Gassner and Johannes Hatzl. A parity domination problem in graphs with bounded treewidth and distance-hereditary graphs. *Computing*, 82(2-3):171–187, 2008. doi:[10.1007/S00607-008-0005-8](https://doi.org/10.1007/S00607-008-0005-8).
- [68] Petr A. Golovach, Jan Kratochvíl, and Ondrej Suchý. Parameterized complexity of generalized domination problems. *Discret. Appl. Math.*, 160(6):780–792, 2012. doi:[10.1016/J.DAM.2010.11.012](https://doi.org/10.1016/J.DAM.2010.11.012).
- [69] Jakob Greilhuber, Philipp Schepper, and Philip Wellnitz. Residue domination in bounded-treewidth graphs. *CoRR*, abs/2403.07524, 2025. doi:[10.48550/ARXIV.2403.07524](https://doi.org/10.48550/ARXIV.2403.07524).
- [70] Jakob Greilhuber, Philipp Schepper, and Philip Wellnitz. Residue domination in bounded-treewidth graphs. In Olaf Beyersdorff, Michal Pilipczuk, Elaine Pimentel, and Kim Thang Nguyen, editors, *42nd International Symposium on Theoretical Aspects of Computer Science, STACS 2025, March 4-7, 2025, Jena, Germany*, volume 327 of *LIPICs*, pages 41:1–41:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. doi:[10.4230/LIPICs.STACS.2025.41](https://doi.org/10.4230/LIPICs.STACS.2025.41).
- [71] Heng Guo and Pinyan Lu. On the complexity of Holant problems. In Andrei A. Krokhin and Stanislav Zivný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 159–177. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:[10.4230/DFU.Vol17.15301.6](https://doi.org/10.4230/DFU.Vol17.15301.6).
- [72] Rudolf Halin. S-functions for graphs. *Journal of Geometry*, 8(1):171–186, 1976. doi:[10.1007/BF01917434](https://doi.org/10.1007/BF01917434).

- [73] Magnús M. Halldórsson, Jan Kratochvíl, and Jan Arne Telle. Independent sets with domination constraints. *Discret. Appl. Math.*, 99(1-3):39–54, 2000. doi:[10.1016/S0166-218X\(99\)00124-9](https://doi.org/10.1016/S0166-218X(99)00124-9).
- [74] Magnús M. Halldórsson, Jan Kratochvíl, and Jan Arne Telle. Mod-2 independence and domination in graphs. *Int. J. Found. Comput. Sci.*, 11(3):355–363, 2000. doi:[10.1142/S0129054100000272](https://doi.org/10.1142/S0129054100000272).
- [75] Frank Harary. Covering and packing in graphs, I. *Annals of the New York Academy of Sciences*, 175(1):198–205, 1970. doi:[10.1111/j.1749-6632.1970.tb56470.x](https://doi.org/10.1111/j.1749-6632.1970.tb56470.x).
- [76] Pavol Hell. Graph packings. *Electron. Notes Discret. Math.*, 5:170–173, 2000. doi:[10.1016/S1571-0653\(05\)80154-4](https://doi.org/10.1016/S1571-0653(05)80154-4).
- [77] Pavol Hell and Jaroslav Nešetřil. *Graphs and homomorphisms*, volume 28 of *Oxford lecture series in mathematics and its applications*. Oxford University Press, 2004. ISBN 978-0-19-852817-3.
- [78] Arne Hoffmann and Lutz Volkmann. On unique k -factors and unique $[1, k]$ -factors in graphs. *Discret. Math.*, 278(1-3):127–138, 2004. doi:[10.1016/S0012-365X\(03\)00248-6](https://doi.org/10.1016/S0012-365X(03)00248-6).
- [79] Sangxia Huang and Pinyan Lu. A dichotomy for real weighted Holant problems. *Comput. Complex.*, 25(1):255–304, 2016. doi:[10.1007/s00037-015-0118-3](https://doi.org/10.1007/s00037-015-0118-3).
- [80] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k -SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:[10.1006/jcss.2000.1727](https://doi.org/10.1006/jcss.2000.1727).
- [81] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:[10.1006/jcss.2001.1774](https://doi.org/10.1006/jcss.2001.1774).
- [82] Yoichi Iwata and Yuichi Yoshida. On the equivalence among problems of bounded width. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 754–765. Springer, 2015. doi:[10.1007/978-3-662-48350-3_63](https://doi.org/10.1007/978-3-662-48350-3_63).
- [83] Lars Jaffke and Bart M. P. Jansen. Fine-grained parameterized complexity analysis of graph coloring problems. *Discret. Appl. Math.*, 327:33–46, 2023. doi:[10.1016/J.DAM.2022.11.011](https://doi.org/10.1016/J.DAM.2022.11.011).
- [84] Mark Jerrum and Kitty Meeks. The parameterised complexity of counting connected subgraphs and graph motifs. *J. Comput. Syst. Sci.*, 81(4):702–716, 2015. doi:[10.1016/J.JCSS.2014.11.015](https://doi.org/10.1016/J.JCSS.2014.11.015).

- [85] Xinrui Jia, Ola Svensson, and Weiqiang Yuan. The exact bipartite matching polytope has exponential extension complexity. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 1635–1654. SIAM, 2023. doi:10.1137/1.9781611977554.CH61.
- [86] Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994. ISBN 3-540-58356-4. doi:10.1007/BFb0045375.
- [87] Donald E. Knuth. Two notes on notation. *The American Mathematical Monthly*, 99(5): 403–422, 1992. doi:10.1080/00029890.1992.11995869.
- [88] Yusuke Kobayashi. Optimal general factor problem and jump system intersection. In Alberto Del Pia and Volker Kaibel, editors, *Integer Programming and Combinatorial Optimization - 24th International Conference, IPCO 2023, Madison, WI, USA, June 21-23, 2023, Proceedings*, volume 13904 of *Lecture Notes in Computer Science*, pages 291–305. Springer, 2023. doi:10.1007/978-3-031-32726-1_21.
- [89] Sanjana Kolisetty, Linh Le, Ilya Volkovich, and Mihalis Yannakakis. The complexity of finding S -factors in regular graphs. *Electron. Colloquium Comput. Complex.*, 26:40, 2019. URL <https://eccc.weizmann.ac.il/report/2019/040>.
- [90] Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 184–192. IEEE, 2021. doi:10.1109/FOCS52979.2021.00026.
- [91] Tuukka Korhonen and Daniel Lokshtanov. An improved parameterized algorithm for treewidth. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 528–541. ACM, 2023. doi:10.1145/3564246.3585245.
- [92] Michael Kowalczyk and Jin-Yi Cai. Holant problems for 3-regular graphs with complex edge functions. *Theory Comput. Syst.*, 59(1):133–158, 2016. doi:10.1007/s00224-016-9671-7.
- [93] Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. *J. ACM*, 67(3):16:1–16:50, 2020. doi:10.1145/3390887.
- [94] Michael Lampis. Finer tight bounds for coloring on clique-width. *SIAM J. Discret. Math.*, 34(3):1538–1558, 2020. doi:10.1137/19M1280326.
- [95] Michael Lampis. First order logic on pathwidth revisited again. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPICs*, pages 132:1–132:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.ICALP.2023.132.

- [96] Michael Lampis. The primal pathwidth SETH. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 1494–1564. SIAM, 2025. doi:[10.1137/1.9781611978322.47](https://doi.org/10.1137/1.9781611978322.47).
- [97] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2):13:1–13:30, 2018. doi:[10.1145/3170442](https://doi.org/10.1145/3170442).
- [98] László Lovász. The factorization of graphs. II. *Acta Mathematica Hungarica*, 23(1-2): 223–246, 1972.
- [99] Pinyan Lu. Complexity dichotomies of counting problems. *Electron. Colloquium Comput. Complex.*, 18:93, 2011. URL <http://eccc.hpi-web.de/report/2011/093>.
- [100] Nicolas El Maalouly. Exact matching: Algorithms and related problems. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPIcs*, pages 29:1–29:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:[10.4230/LIPIcs.STACS.2023.29](https://doi.org/10.4230/LIPIcs.STACS.2023.29).
- [101] Nicolas El Maalouly, Raphael Steiner, and Lasse Wulf. Exact matching: Correct parity and FPT parameterized by independence number. In Satoru Iwata and Naonori Kakimura, editors, *34th International Symposium on Algorithms and Computation, ISAAC 2023, December 3-6, 2023, Kyoto, Japan*, volume 283 of *LIPIcs*, pages 28:1–28:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:[10.4230/LIPIcs.ISAAC.2023.28](https://doi.org/10.4230/LIPIcs.ISAAC.2023.28).
- [102] Dániel Marx and Paul Wollan. An exact characterization of tractable demand patterns for maximum disjoint path problems. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 642–661. SIAM, 2015. doi:[10.1137/1.9781611973730.44](https://doi.org/10.1137/1.9781611973730.44).
- [103] Dániel Marx, Govind S. Sankar, and Philipp Schepper. Degrees and gaps: Tight complexity results of general factor problems parameterized by treewidth and cutwidth. *CoRR*, abs/2105.08980, 2021. doi:[10.48550/ARXIV.2105.08980](https://doi.org/10.48550/ARXIV.2105.08980).
- [104] Dániel Marx, Govind S. Sankar, and Philipp Schepper. Degrees and gaps: Tight complexity results of general factor problems parameterized by treewidth and cutwidth. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 95:1–95:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:[10.4230/LIPIcs.ICALP.2021.95](https://doi.org/10.4230/LIPIcs.ICALP.2021.95).
- [105] Dániel Marx, Govind S. Sankar, and Philipp Schepper. Anti-factor is FPT parameterized by treewidth and list size (but counting is hard). In Holger Dell and Jesper Nederlof, editors, *17th International Symposium on Parameterized and Exact Computation, IPEC 2022, September 7-9, 2022, Potsdam, Germany*, volume 249 of

- LIPICs*, pages 22:1–22:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.IPEC.2022.22.
- [106] Dániel Marx, Govind S. Sankar, and Philipp Schepper. Anti-factor is FPT parameterized by treewidth and list size (but counting is hard). *Algorithmica*, 87(1):22–88, 2025. doi:10.1007/S00453-024-01265-W.
- [107] Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13-15 October 1980*, pages 17–27. IEEE Computer Society, 1980. doi:10.1109/SFCS.1980.12.
- [108] Burkhard Monien. How to find long paths efficiently. In G. Ausiello and M. Lucertini, editors, *Analysis and Design of Algorithms for Combinatorial Problems*, volume 109 of *North-Holland Mathematics Studies*, pages 239–254. North-Holland, 1985. doi:10.1016/S0304-0208(08)73110-4.
- [109] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Comb.*, 7(1):105–113, 1987. doi:10.1007/BF02579206.
- [110] James Oxley. *Matroid Theory*. Oxford University Press, 02 2011. ISBN 978-0-19-856694-6. doi:10.1093/acprof:oso/9780198566946.001.0001.
- [111] Mihai Pătraşcu and Ryan Williams. On the possibility of faster SAT algorithms. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1065–1075. SIAM, 2010. doi:10.1137/1.9781611973075.86.
- [112] Julius Petersen. Die Theorie der regulären graphs. *Acta Mathematica*, 15:193 – 220, 1891. doi:10.1007/BF02392606.
- [113] László Pyber. Covering the edges of a graph by In *Colloq. Math. Soc. János Bolyai, 60. Sets, graphs and numbers, Budapest, Hungary*, volume 60, pages 583–610, 1991.
- [114] Neil Robertson and Paul D. Seymour. Graph minors. III. Planar tree-width. *J. Comb. Theory B*, 36(1):49–64, 1984. doi:10.1016/0095-8956(84)90013-3.
- [115] Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory B*, 63(1):65–110, 1995. doi:10.1006/JCTB.1995.1006.
- [116] Marc Roth and Philip Wellnitz. Counting and finding homomorphisms is universal for parameterized complexity theory. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2161–2180. SIAM, 2020. doi:10.1137/1.9781611975994.133.
- [117] Marc Roth, Johannes Schmitt, and Philip Wellnitz. Counting small induced subgraphs satisfying monotone properties. *SIAM J. Comput.*, 53(6):S20–139, 2024. doi:10.1137/20M1365624.

- [118] Philipp Schepper. The complexity of formal language decision problems. Bachelor's Thesis, Technische Universität Kaiserslautern, Kaiserslautern, 2018.
- [119] Philipp Schepper. Fine-grained complexity of regular expression pattern matching and membership. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPICs*, pages 80:1–80:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:[10.4230/LIPICS.ESA.2020.80](https://doi.org/10.4230/LIPICS.ESA.2020.80).
- [120] Philipp Schepper. Fine-grained complexity and shaving log-factors for regular expression pattern matching and membership. Master's Thesis, Saarland University, Saarbrücken, 2020.
- [121] András Sebő. General antifactors of graphs. *J. Comb. Theory B*, 58(2):174–184, 1993. doi:[10.1006/JCTB.1993.1035](https://doi.org/10.1006/JCTB.1993.1035).
- [122] Hadas Shachnai and Meirav Zehavi. Representative families: A unified tradeoff-based approach. *J. Comput. Syst. Sci.*, 82(3):488–502, 2016. doi:[10.1016/j.jcss.2015.11.008](https://doi.org/10.1016/j.jcss.2015.11.008).
- [123] Yossi Shiloach. Another look at the degree constrained subgraph problem. *Inf. Process. Lett.*, 12(2):89–92, 1981. doi:[10.1016/0020-0190\(81\)90009-0](https://doi.org/10.1016/0020-0190(81)90009-0).
- [124] Klaus Sutner. Additive automata on graphs. *Complex Syst.*, 2(6), 1988. URL http://www.complex-systems.com/abstracts/v02_i06_a03.html.
- [125] Klaus Sutner. Linear cellular automata and the garden-of-eden. *The Mathematical Intelligencer*, 11(2):49–53, 1989. doi:[10.1007/BF03023823](https://doi.org/10.1007/BF03023823).
- [126] Jan Arne Telle. Complexity of domination-type problems in graphs. *Nord. J. Comput.*, 1(1):157–171, 1994.
- [127] Jan Arne Telle and Andrzej Proskurowski. Practical algorithms on partial k-trees with an application to domination-like problems. In Frank K. H. A. Dehne, Jörg-Rüdiger Sack, Nicola Santoro, and Sue Whitesides, editors, *Algorithms and Data Structures, Third Workshop, WADS '93, Montréal, Canada, August 11-13, 1993, Proceedings*, volume 709 of *Lecture Notes in Computer Science*, pages 610–621. Springer, 1993. doi:[10.1007/3-540-57155-8_284](https://doi.org/10.1007/3-540-57155-8_284).
- [128] William T. Tutte. The factors of graphs. *Canadian Journal of Mathematics*, 4:314–328, 1952. doi:[10.4153/CJM-1952-028-2](https://doi.org/10.4153/CJM-1952-028-2).
- [129] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979. doi:[10.1137/0208032](https://doi.org/10.1137/0208032).
- [130] Johan M. M. van Rooij. Fast algorithms for join operations on tree decompositions. In Fedor V. Fomin, Stefan Kratsch, and Erik Jan van Leeuwen, editors, *Treewidth, Kernels, and Algorithms - Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, volume 12160 of *Lecture Notes in Computer Science*, pages 262–297. Springer, 2020. doi:[10.1007/978-3-030-42071-0_18](https://doi.org/10.1007/978-3-030-42071-0_18).

- [131] Johan M. M. van Rooij. A generic convolution algorithm for join operations on tree decompositions. In Rahul Santhanam and Daniil Musatov, editors, *Computer Science - Theory and Applications - 16th International Computer Science Symposium in Russia, CSR 2021, Sochi, Russia, June 28 - July 2, 2021, Proceedings*, volume 12730 of *Lecture Notes in Computer Science*, pages 435–459. Springer, 2021. doi:[10.1007/978-3-030-79416-3_27](https://doi.org/10.1007/978-3-030-79416-3_27).
- [132] Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009. doi:[10.1007/978-3-642-04128-0_51](https://doi.org/10.1007/978-3-642-04128-0_51).
- [133] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra (3. ed.)*. Cambridge University Press, 2013. ISBN 978-1-107-03903-2.
- [134] D.J.A. Welsh. *Matroid Theory*. Dover books on mathematics. Dover Publications, 2010. ISBN 978-0-486-47439-7.
- [135] Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 3792–3835. SIAM, 2024. doi:[10.1137/1.9781611977912.134](https://doi.org/10.1137/1.9781611977912.134).

Notation

Problems and Hypotheses

- X -ANTI FAC The variant of $GENFAC$ where all vertices are assigned the set $\mathbb{N} \setminus X$; see Definition 5.1.2
- ANTI FAC $_x$ The variant of $GENFAC$ where all vertices are assigned a cofinite set with at most x forbidden degrees; see Definition 5.1.2
- q -CSP- B The constraint satisfaction problem where each variable can take B values and the constraints contain up to q variables; see Definition 4.1.6
- (σ, ρ) -DOM SET The (σ, ρ) -DOMINATING SET problem; see Definition 8.1.1
- #EDGE $COVER$ The problem of counting edge covers, that is, $\{0\}$ -#ANTI FAC
- Λ - FAC The variant of $GENFAC$ where every vertex is assigned the set Λ ; see Definition 5.1.2
- $GENFAC$ The GENERALIZED FACTOR problem; see Definition 5.1.1
- #MAX $INDSET$ The problem of counting maximum independent sets; see Definition 7.9.1
- SETH The STRONG EXPONENTIAL TIME HYPOTHESIS; see Conjecture 4.1.1
- #SETH The #STRONG EXPONENTIAL TIME HYPOTHESIS; see Conjecture 4.1.2
- pw-SETH The primal pathwidth SETH; see Conjecture 4.1.4
- pw-#SETH The counting variant of the primal pathwidth SETH; see Conjecture 4.1.5
- CSPH The derived variant of SETH for q -CSP- B ; see Conjecture 4.1.7
- pw-CSPH The primal pathwidth CSPH; see Conjecture 4.1.8
- pw-#CSPH The counting variant of the primal pathwidth CSPH; see Conjecture 4.1.9
- MONO-pw-CSPH The monotone variant of pw-CSPH; see Conjecture 4.1.11
- MONO-pw-#CSPH The monotone variant of pw-#CSPH; see Conjecture 4.1.12

General Notation

- $\mathcal{C}$ A compatibility graph; see Definition 3.2.14
- ε The vector with no elements, that is, the dimension-0 vector
- $\mathbb{N}$ The natural numbers starting at 0
- $\emptyset$ The set with no elements, that is, the empty set

Notation

- $\mathcal{O}$ Landau Big-O notation
- ω The matrix multiplication exponent with $\omega \leq 2.371552$ [135]
- $\mathbb{Q}$ The rational numbers
- $\mathbb{R}$ The real numbers
- $\mathbb{R}^+$ The non-negative real numbers, that is, including 0
- $\top$ A state that is considered to be the maximum value; see Definition 3.1.1
- $\mathbb{Z}$ The integers
- cost The cost of a finite or cofinite set; see Definition 8.1.7
- max-gap The largest gap of a set S ; see Definition 2.1.2
- $\#\text{Sol}(I)$ The number of solutions for instance I of some counting problem
- top The largest reasonable element of a finite or cofinite set; see Definition 2.1.3
- $\langle a \rangle$ The binary encoding of the number a
- $\llbracket c \rrbracket$ The number represented by the binary vector c
- $\llbracket P \rrbracket$ Iverson bracket notation:
when predicate P is true, $\llbracket P \rrbracket$ returns 1, otherwise $\llbracket P \rrbracket$ returns 0
- $\binom{S}{k}$ The subsets of S of size exactly k .
- $\sim_H$ H -compatibility of two vectors; see Definition 3.2.1
- $\sim_r^k$ k - r -compatibility of two vectors; see Definition 3.2.3
- $\dot{\cup}$ The union of two disjoint sets
- $[i..j]$ The set of integers from i to j , including i and j
- $[i..j)$ The set of integers from i to j but not including j
- $\#_a(s)$ The number of occurrences of symbol a in vector s
- $\#_A(s)$ The number of occurrences of some symbol from A in vector s
- $\subseteq_{H\text{-rep}}$ H -representation; see Definition 3.2.2
- $\subseteq_{r\text{-rep}}^k$ k - r -representation; see Definition 3.2.3
- $\sqcup$ The combination of two vectors (with disjoint domains); see Definition 2.1.4
- $\oplus_t$ The truncated addition with respect to the number t ; see Definition 3.1.1
- $(a_1 \mapsto v_1, \dots)$ The explicitly defined function that maps the value a_1 to v_1 , and so on

Relations

- EQ The equality relation; either all or no elements are selected;
see Definition 2.1.6

- $\text{hw}(x)$ The Hamming-weight of vector x , that is, the number of 1s; see Definition 2.1.6
- $\text{HW}_{=1}$ The relation where exactly one element is selected; see Definition 2.1.6
- $\text{HW}_{\geq 1}$ The relation where at least one element is selected; see Section 10.8.1
- $\text{HW}_{\leq 1}$ The relation where at most one element is selected; see Section 10.8.1

Treewidth and Pathwidth

- $\beta(\cdot)$ A bag of a tree/path decomposition unless mentioned otherwise
- pw The pathwidth of some instance
- tw The treewidth of some instance
- $\leq_{\text{pw}}$ A pathwidth-preserving Turing-reduction; see Definition 2.2.2
- $\leq^{\text{ar}}$ An arity-preserving Turing-reduction; see Definition 2.2.3
- $\leq_{\text{pw}}^{\text{ar}}$ A pathwidth- and arity-preserving Turing-reduction; see Definition 2.2.4

General Factor

- Δ^* The total arity of an instance; see Definition 7.1.3

General Dominating Set

- m The value denoting that σ and ρ are m -structured; see Definition 8.1.3
- r_{top} The value $\text{top } \rho$
- s_{top} The value $\text{top } \sigma$
- t_{top} The maximum among s_{top} and r_{top}
- $\mathbb{R}$ The set of all ρ -states
- $\mathbb{S}$ The set of all σ -states
- $\mathbb{A}$ The set of all states, that is, the union of $\mathbb{S}$ and $\mathbb{R}$
- $\oplus$ The operation to combine two (vectors of) states; see Definition 9.1.12
- $\text{inv}(\tau)$ The inverse of the state τ ; see Definition 10.1.6
- $\vec{\sigma}(x)$ The σ -vector of the state-vector x ; see Definition 9.1.1
- $\vec{w}(x)$ The weight-component of the state-vector x ; see Definition 9.1.1
- $\vec{w}_m(x)$ The weight-component of the state-vector x modulo m ; see Definition 9.1.1
- $\text{cap}(\vec{s})$ The capacity vector based on the σ -vector $\vec{s}$; see Definition 9.1.1
- $\text{rem}_{\mathcal{W}_S}(u, o)$ The remainder vector of state-vector u with respect to origin vector o and witness-vectors $\mathcal{W}_S$; see Definition 9.1.16

Notation

- $z \downarrow_o$ The compression of vector z with respect to origin vector o ; see Definition 9.1.18
- $z \uparrow_o$ The decompression of vector z with respect to origin vector o ; see Definition 9.1.21
- $\text{scp}(C)$ The scope of a relational constraint C ; see Definition 10.1.1
- $R(C)$ The relation of a relational constraint C ; see Definition 10.1.1
- Ω_0 The set of all finite or cofinite subsets of $\mathbb{N}$
- Ω The set of all non-empty finite or cofinite subsets of $\mathbb{N}$, that is, $\Omega = \Omega_0 \setminus \{\emptyset\}$
- $\text{ext}_G(\tau)$ The number of extension of state τ to gadget G

*Ich weiß, was ich kann
Trotzdem ab und an
Komm'n Zweifel daran
Ich frage mich dann
Wofür ich das alles mache*

*Und mir ist mittlerweile klar, das waren Jahre ohne Pause
Und eigentlich weiß ich genau, was ich brauche
Eigentlich weiß ich genau, was ich brauche
'Ne Pause*

*Drei Tage am Meer
Und ich weiß wieder, wer ich bin
Drei Tage am Meer
Und ich weiß wieder, wer ich bin*

— AnnenMayKantereit; 3 Tage am Meer