Modeling and Formal Verification of Probabilistic Reconfigurable Systems

Dissertation
zur Erlangung des Grades
des Doktors der Ingenieurwissenschaften
der Naturwissenschaftlich-Technischen
Fakultät der Universität des Saarlandes
&
Tunisia Polytechnic School
University of Carthage

von

Oussama **KHLIFI**

Saarbrücken
2020

Tag des Kolloquiums: 10. Juli 2020

Dekan:	Prof. Dr. rer. nat. Guido Kickelbick
Berichterstatter:	Prof. Dr.-Ing. Georg Frey
	Prof. Dr.-Ing. Stefan Seelecke
	Prof. Mohamed Abid
Akad. Mitglied:	Dr.-Ing. Emanuele Grasso
Weitere Mitglieder:	Prof. Dr. Mohamed Khalgui
	Dr. Olfa Mosbahi
Vorsitz:	Prof. Dr.-Ing. Matthias Nienhaus

Abstract

In this thesis, we propose a new approach for formal modeling and verification of adaptive probabilistic systems. Dynamic reconfigurable systems are the trend of all future technological systems, such as flight control systems, vehicle electronic systems, and manufacturing systems. In order to meet user and environmental requirements, such a dynamic reconfigurable system has to actively adjust its configuration at run-time by modifying its components and connections, while changes are detected in the internal/external execution environment. On the other hand, these changes may violate the memory usage, the required energy and the concerned real-time constraints since the behavior of the system is unpredictable. It might also make the system's functions unavailable for some time and make potential harm to human life or large financial investments. Thus, updating a system with any new configuration requires that the post reconfigurable system fully satisfies the related constraints. We introduce GR-TNCES formalism for the optimal functional and temporal specification of probabilistic reconfigurable systems under resources constraints. It enables the optimal specification of a probabilistic, energetic and memory constraints of such a system. To formally verify the correctness and the safety of such a probabilistic system specification, and the non-violation of its properties, an automatic transformation from GR-TNCES models into PRISM models is introduced. Moreover, a new approach XCTL is also proposed to formally verify reconfigurable systems. It enables the formal certification of uncompleted and reconfigurable systems. A new version of the software ZIZO is also proposed to model, simulate and verify such GR-TNCES model. To prove its relevance, the latter was applied to case studies; it was used to model and simulate the behavior of an IPV4 protocol to prevent the energy and memory resources violation. It was also used to optimize energy consumption of an automotive skid conveyor.

Keywords: Reconfiguration, GR-TNCES, Petri net, Modeling, Simulation, Model checking.

Abstrakt

In dieser Arbeit wird ein neuer Ansatz zur formalen Modellierung und Verifikation dynamisch rekonfigurierbarer Systeme vorgestellt. Dynamische rekonfigurierbare Systeme sind in vielen aktuellen und zukünftigen Anwendungen, wie beispielsweise Flugsteuerungssystemen, Fahrzeugelektronik und Fertigungssysteme zu finden. Diese Systeme weisen ein probabilistisches, adaptives Verhalten auf. Um die Benutzer- und Umgebungsbedingungen kontinuierlich zu erfüllen, muss ein solches System seine Konfiguration zur Laufzeit aktiv anpassen, indem es seine Komponenten, Verbindungen zwischen Komponenten und seine Daten modifiziert (adaptiv), sobald Änderungen in der internen oder externen Ausführungsumgebung erkannt werden (probabilistisch). Diese Anpassungen dürfen Beschränkungen bei der Speichernutzung, der erforderlichen Energie und bestehende Echtzeitbedingungen nicht verletzen. Eine nicht geprüfte Rekonfiguration könnte dazu führen, dass die Funktionen des Systems für einige Zeit nicht verfügbar wären und potenziell menschliches Leben gefährdet würde oder großer finanzieller Schaden entstünde. Somit erfordert das Aktualisieren eines Systems mit einer neuen Konfiguration, dass das rekonfigurierte System die zugehörigen Beschränkungen vollständig einhält. Um dies zu überprüfen, wird in dieser Arbeit der GR-TNCES-Formalismus, eine Erweiterung von Petrinetzen, für die optimale funktionale und zeitliche Spezifikation probabilistischer rekonfigurierbarer Systeme unter Ressourcenbeschränkungen vorgeschlagen. Die entstehenden Modelle sollen über probabilistische model checking verifiziert werden. Dazu eignet sich die etablierte Software PRISM. Um die Verifikation zu ermöglichen wird in dieser Arbeit ein Verfahren zur Transformation von GR-TNCES-Modellen in PRISM-Modelle beschrieben. Eine neu eingeführte Logik (XCTL) erlaubt zudem die einfache Beschreibung der zu prüfenden Eigenschaften. Die genannten Schritte wurden in einer Softwareumgebung für den automatisierten Entwurf, die Simulation und die formale Verifikation (durch eine automatische Transformation nach PRISM) umgesetzt. Eine Fallstudie zeigt die Anwendung des Verfahren.

LIST OF FIGURES

Figure 1 Statechart example **13**

Figure 2 Timed Automaton Example **14**

Figure 3 A composite module of an NCES **15**

Figure 4 Schematic view of the model-checking approach, [Baier 2008] **21**

Figure 5 Characteristics of Model Checking, adapted from [Baier 2008] **22**

Figure 6 Example of a program in PRISM **23**

Figure 7 Sustainable energy hierarchy, [Sustainable 2014] **24**

Figure 8 CAD model of the skid conveyor **25**

Figure 9 Example of Public Network **26**

Figure 10 Macro-step, micro-step **31**

Figure 11 Example of a GR-TNCES architecture **32**

Figure 12 Skid conveyor system at ZeMA. **36**

Figure 13 GR-TNCES System model **37**

Figure 14 Simulation step **44**

Figure 15 High Probabilistic Simulation **45**

Figure 16 Low Probability Simulation **46**

Figure 17 Algorithm of Average Probability Simulation **47**

Figure 18 Energy and Memory simulation. **48**

Figure 19 Example of a System Model. **50**

Figure 20 Component of a Module. **50**

Figure 21 Example of ZIZO-PRISM Transformation **52**

Figure 22 ZIZO-PRISM Transformation Algorithm **52**

Figure 23 Reconfigurable railway network structure **55**

Figure 24 New Verification Approach **56**

Figure 25 ILTS of the railway model **57**

Figure 26 Degradation approach **61**

Figure 27 Main window of ZIZO1.0. **64**

Figure 28 Packages of ZIZO1.1. **65**

Figure 29 Package Components **67**

Figure 30 Module Example. **69**

Figure 31 Control system of the skid conveyor. **70**

Figure 32 Proposed Transport Model. **71**

Figure 33 Model on motors. **72**

Figure 34 Model of the PLC. **72**

Figure 35 Standard system's model. **73**

Figure 36 Optimization of energy consumption. **74**

Figure 37 Devices network **75**

Figure 38 Memory warning message **77**

Figure 39 Curves for memory and energy consumption **77**

Figure 40 PRISM code of ZeroConf protocol mode. **78**

Figure 41 Probabilistic Property. **79**

Figure 42 Simulation at step 334. **80**

LIST OF TABLES

Table 1 Comparison Table **40**

Table 2 Syntax of PRISM symbols **51**

Table 3 ZIZO-PRISM Correspondence **51**

Table 4 Links Description **66**

LIST OF SYMBOLS AND ABBREVIATIONS

RDECS: Reconfigurable Discrete Event Control System

PRDECS: Probabilistic Reconfigurable DECS

NCES: Net Condition/Event System

TNCES: Timed Net Condition/Event System

R-TNCES: Reconfigurable Timed Net Condition/Event System

GR-TNCES: Generalized Reconfigurable Timed Net Condition/Event System

IPV4: Internet Protocol Version 4

RMS: Reconfigurable Manufacturing System

CTL: Computation Tree Logic

ECTL: Extended Computation Tree Logic

TCTL: Timed Computation Tree Logic

PCTL: Probabilistic Computation Tree Logic

XCTL: X Computation Tree Logic

CPN: Coloured Petri Nets

ZeMA: Zentrum für Mechatronik und Automatisierungstechnik

WSN: Wireless Sensor Network

OS: Operating System

RSML: Requirements State Machine Language

ViVe: Visual Verification

MDP: Markov Decision Processes

PTA: Probabilistic Timed Automata

LTL: Linear Time Logic

PCTL*: which uses both PCTL and LTL

CSL: Continuous Stochastic Logic

DTMC: Discrete-Time Markov Chains

CTMC: Continuous Time Markov Chains

MTBDD: Multi-Terminal Binary Decision Diagram

trig: a trigger event

cond: an optional guarding condition

acts: a possibly empty list of action events

x: a clock

y: a clock

P: non-empty finite set of places

T: non-empty finite set of transitions

F: a set of flow arcs

CN: a set of conditions

EN: a set of events

C_{in} : a set of condition inputs

E_{in} : a set of event inputs

C_{out} : a set of condition outputs

E_{out}: a set of event outputs

B_c : is a set of condition input arcs in a NCES module

B_e : is a set of event input arcs in a NCES module

C_s : is a set of condition output arcs,

D_s: is a set of event output arcs,

m_0 : is the initial marking

w, eft , lft : are a given integer

Ψ *:* is input/output structure of TNCES module

DR: represents the minimum time interval to fire a token

DL: represents the maximum time interval to fire a token

D_0: is the initial set of the clocks

R: The control module consisting of a set of reconfiguration functions R={r1,...,rn}

B: is the behavior module

DC: is a superset of time constraints on output arcs

V: a function which maps an event-processing mode for every transition

W: a function which maps a weight to a flow arc

Z: is a finite set of states

E: is a finite set of transitions between states

φ: a CTL formula.

M: a Kripke structure

f: a formula of temporal logic

s: a states s of M

ARP: Address Resolution Protocol

GARP: Gratuitous ARP

QW: is the probability on the arcs

TN: the set of all system net structures

r: a reconfiguration function

•*r:* original RTNCES before applying the reconfiguration function *r*

r•: source RTNCES after applying the reconfiguration function *r*

Cond: The precondition of *r*

Q: represents the probability to reach each TNCES branch.

E_0': The energy requirements of the chosen probabilistic RTNCES scenario,

M_0': The memory requirements of the chosen probabilistic RTNCES scenario,

S: the structure modification instruction of the reconfiguration scenario.

X: the state processing function before and after *r*

ß: a TNCES

Cost TNCES: the needed resources by this TNCES.

R_{Max} : the most probabilistic reconfiguration scenario

Cost $TNCES_{Max}$ *:* the resources required by the most probabilistic reconfigurable scenarios.

C: is a set of macro-steps

In: the initial standard configuration

Rec: a reconfiguration function

I: set of initial states

var, var': a variable

src(*t*) : the source local state

dst(*t*) : destination local state

evt(*t*) : trigger event

cond(*t*) : guarding condition,

time(*t*) : the firing time interval

mode(*t*) : the firing mode{AND, OR}

prob(*t*) : the firing probability

curr(*t*): the current local state of the system

m: micro state of the system

$micro_m$ *:* the progress of the system at run-time process

Macro: represents the macro-step

RTN_{ski}: skid conveyor system

B_{skid}*:* Behavioral module of the skid

R_{skid}*:* Control module of the skid

E_{skid}: energy reserve of the skid

M_{skid}: Memory reserve of the skid

EN1: external event to activate Rec1

EN2: external event to activate Rec2

Inhalt

Introduction **2**
1.1 Context **2**
1.2 Problems **4**
1.3 Contribution **4**
1.4 Outline **6**
1.5 Publications **6**
1.6 Collaboration **7**

State of the art **9**
2.1 Probabilistic Reconfigurable Systems under Resources Constraints **9**
2.1.1 Probabilistic Reconfigurable Systems 9
2.1.2 Energy and Memory Resources 10
2.2 Modeling of Discrete Event Systems **12**
2.2.1 Net Condition/Event Systems 14
2.2.2 Timed Net Condition/Event Systems 15
2.2.3 Reconfigurable Timed Net Condition/Event System 16
2.2.4 Statecharts 12
2.2.5 Timed Autamata 13
2.2.6 Tools Modeling Petri Nets 17
2.3 Temporal Logic **17**
2.3.1 Computation Tree Logic 18
2.3.2 Extended Computation Tree Logic 18
2.3.3 Timed Computation Tree Logic 19
2.3.4 Probabilistic Computation Tree Logic 19
2.4 Formal Verification **20**
2.4.1 Model Checking 20
2.4.2 PRISM Model Checker 22
2.5 Case Studies **23**
2.5.1 Automotive Transport System 23
2.5.2 IPV4 ZeroConf 25
2.6 Summary **27**

Modeling and Specification **29**
3.1 Introduction **29**
3.2 GR-TNCES **29**
3.2.1 Motivation 29
3.2.2 Formalization 29
3.2.3 Dynamics of GR-TNCES 31
3.3 Specification Approach **33**
3.3.1 Motivation 33
3.3.2 System Specification 33

3.4 Case Study: Skid Conveyor **35**
3.4.1 Description 35
3.4.2 Specification 36
3.5 Discussion **40**
3.6 Summary **41**

Simulation and Formal Verification **44**
4.1 Introduction **44**
4.2 Simulation **44**
4.2.1 Probabilistic Simulation 45
4.2.2 Energy Simulation 46
4.2.3 Memory Simulation 47
4.3 Formal Verification **48**
4.3.1 Formal verification: Export to PRISM 49
4.3.2 New Verification Approach 53
4.3.3 Incomplete Labeled Transition System 56
4.3.4 XCTL Model Checker 58
4.3.5 Marking Algorithms 59
4.3.6 Degraded Verification Mode 60
4.3.7 Discussion 61
4.4 Summary **62**

ZIZO1.1: New Environment for Modeling, Simulation and Verification of APDECS 64
5.1 Introduction **64**
5.2 New Environment ZIZO1.1 **64**
5.2.1 ZIZO1.0 64
5.2.2 New Version of ZIZO: Architecture 65
5.2.3 Implementation 67
5.2.4 System Modeling and Simulation 68
5.3 Case Study 1: Skid Conveyor System **69**
5.3.1 Structure 69
5.3.2 System Modeling 70
5.3.3 Simulation and Optimization 73
5.3.4 Discussion 74
5.4 Case Study 2: IPV4 ZeroConf **74**
5.4.1 Principles and Challegnes 74
5.4.2 IPV4 ZeroConf: Modeling 75
5.4.3 IPV4 ZeroConf: Simulation 76
5.4.4 IPV4 ZeroConf: Formal Verification 77
5.5 Summary **80**

Conclusion and Perspectives **82**
6.1 Context **82**

6.2 Problems **82**
6.3 Output and Originalities **83**
6.4 Tool **83**
6.5 Perspectives **84**

ACKNOWLEDGEMENTS **85**

REFERENCES **86**

Chapter 1
Introduction

Contents

1.1 Context..... **2**
1.2 Problems..... **4**
1.3 Contribution..... **4**
1.4 Outline..... **6**
1.5 Publications..... **6**
1.6 Collaboration..... **7**

INTRODUCTION

Reconfigurability is an expected feature of all future technological systems since it can increase system flexibility and reliability and decrease time cost in developing new products. These systems are man-made and rely on complex automatic control technologies and they are always considered as reconfigurable discrete event control systems (RDECS). This thesis reports the modeling, simulation and formal verification [Tong 2017] [Preuße 2012] of probabilistic reconfigurable DECS (PRDECS) based on the formalism Net Condition/Event System (NCES) [Hanisch 1999] which is a modular extension of Petri nets [Genter 2007], [Zhang 2017], [Cong 2017] and [Ma 2008]. As the beginning of a dissertation, this chapter introduces the study object, state of the art on the topic, and the organization of this dissertation.

1.1 Context

Recently, the need for reconfigurable manufacturing systems comes from unpredictable market changes that are occurring increasingly. These changes are driven by aggressive economic competition on a global scale. It includes: increasing frequency introduction of new products, changing the existing products, instable demand and government regulations (safety and environment), and changes in process technology [Koren 1999]. To stay competitive in this environment; industrials should be able to react to changes rapidly and cost-effectively. Moreover, the manufacturing lead-time can be also reduced through the rapid design of systems that are created from modular components, or by the reconfiguration [Wang 2015], [Wang 2016] and [Salem 2014] of an existing system to produce new products. Thus, the challenge of coping with large fluctuations in product demand cannot be solved with dedicated lines that are not scalable. Moreover, the available production capacity is not fully used, e.g., a research study carried out on a manufacturer of components for the car industry has shown that the average utilization of the transport lines available was only 53% [Koren 1999]. The reason for this low utilization is that some products being used at the early stages or at the end of their life cycle are needed in low quantities. Even products in the maturity phase do not always reach the production volumes predicted at the design time of the dedicated manufacturing line [Koren 1999].

Reconfigurability is defined as the ability to repeatedly change and rearrange the components of a system in a cost-effective way. This concept is presented through its application in computing, automated assembly and robotics [Zhang 2015]. More recently, with the development of information technologies, dynamic reconfigurability has begun to draw more and more attention in industrial and academic communities, due to increased safety and reliability demands beyond what a conventional control system can offer. Dynamic reconfigurable systems are no longer limited to high-end systems such as aerospace and nuclear power systems. Common products, such as automobiles, are

increasingly dependent on microelectronic/mechatronic systems, onboard communication networks, and software, requiring new techniques for achieving dynamic reconfigurability. The objectives of dynamic reconfigurations [Hamid 2010] are not limited to fault tolerance but also to actively adjust system configurations to adapt to frequently changed user requirements or environment [Zhang 2015], [Schlegel 2004].

A Reconfigurable Manufacturing System (RMS) [Koren 1999] is designed to deal with various changes in the structure [Chen 2014], as well as in the hardware and software components, in order to quickly adjust the production capacity and functionality [Ben Salem 2016]. A reconfiguration scenario is any automatic run-time operation that adds/removes hardware/software components in the system [Salem 2014]. It can also modify the connections between them and change the states in response to errors or satisfying unpredictable user requirements. It is also the qualitative changes in the structure, functionality, and algorithms of the control systems [Dumitrache 2000]. It is due to changes of user requirements, the controlled system or of the environment the system behaves within [Ben Salem 2016]. Adaptive systems have been deeply studied over the last decade as a means for developing dependable applications, always more flexible and dynamic. Adaptive probabilistic systems are able to modify their behaviors to cope with unpredictable significant changes at runtime such as component failures. A probabilistic system [Forejt 2011] is one in which the occurrence of events/conditions signals cannot be predicted in advance. A reconfiguration scenario in a Petri net model [Tong 2016] [Wu 2012a] could be introduced as: (i) any addition/removal of places, (ii) any addition/removal of transitions, and (iii) any update of marking. The system can be specified by different sub-TNCESs defining different possible behaviors/scenarios to be followed under well-defined events/conditions signals [Khalgui 2011].

Computing and control systems have to adapt to systems' changing conditions in order to fulfill new demanding requirements. Reconfiguration is often a major issue for some critical systems and other intelligent systems since it can make the system unstable or violate its requirements [Khlifi 2018a]. Thus, the new configuration should fully satisfy the expected requirements such as the energy, memory resources, real-time constraints [Gherbi 2006], [Kopetz 2003] and functional safety i.e., a part of our work try to focus on: (i) how can we check if the system specification satisfies the available energy and memory constraints after any unpredictable reconfiguration? (ii) How can we guarantee the non-violation of the resources after applying the most probabilistic scenario or the lowest probabilistic scenario? (iii) How can we formally prove the correctness of a reconfigurable or an incomplete system specification? In general, all the system requirements can be reduced to two general properties: functional and temporal correctness [Khlifi, 2015]. These can be further split up into two corresponding questions: Will the system respond to any input change with the correct output change (value correctness)? And will it do so within the correct time bounds (temporal correctness)? [Khlifi, 2015].

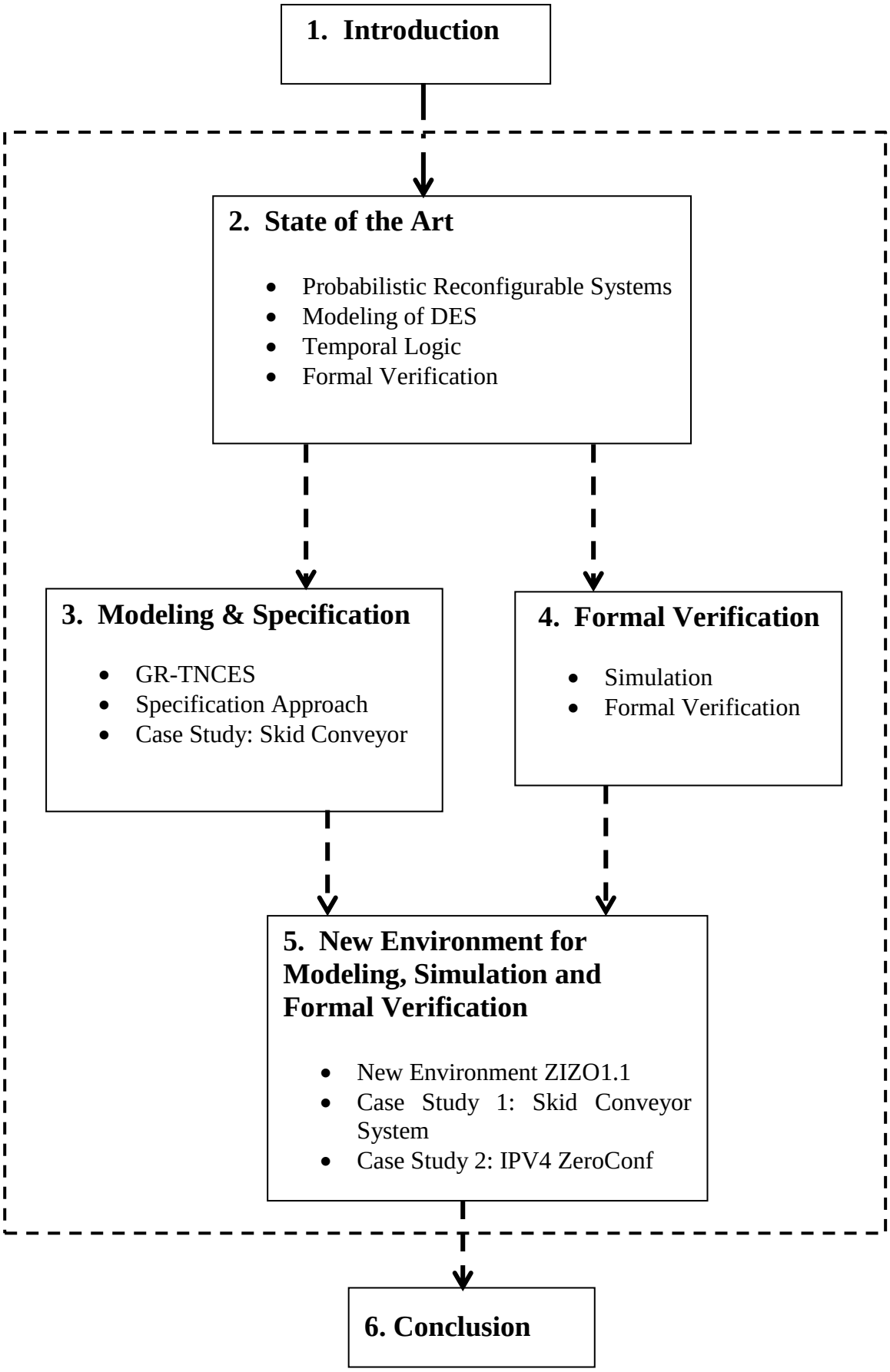
1.2 Problems

The development of safe distributed reconfigurable control systems is not a trivial activity because a failure can be critical for the safety of human beings, e.g., air and railway traffic control [Hanisch 1997]. These systems also should be easily modified and extended after any evolution of the environment within the system behaves. Moreover, each reconfiguration scenario should meet energy, memory and real time constraints since the system could violate its resources during run-time process which lead to a blocking situation and dangerous effects.

- The first problem of this thesis is to extend the formalism R-TNCES, which is not able to specify probabilistic systems and express energy and memory constraints, i.e., we extend it with new solutions for optimal specification and control of predictable as well as unpredictable behaviors and try to cover the problem of memory and energy specification. More precisely how to enable the modeling of probabilistic behavior, energy and memory resources using R-TNCES formalism.
- The second reveals the problem of formal verification of reconfigurable properties; i.e., how can we formally certify such a system with the functional properties that change during run-time operation because the current verification approach cannot deal with reconfigurable properties.
- The third focuses on the technical part of the previous theoretical issues; i.e., we want to present a complete environment to model and verify probabilistic reconfigurable systems running under limited energy and memory resources.

1.3 Contribution

This thesis focuses on modeling, simulation and formal verification of probabilistic adaptive systems running under resources and real-time constraints. The contributions consist on presenting a complete approach running from system specification, simulation to formal verification. The following organizational chart illustrates the link between the different contribution parts: (i) we started by presenting a new extension of the formalism R-TNCES and a new based specification approach. It enables the specification and supervision of probabilistic systems under resources constraints. This contribution is applied later to an automotive transport system. (ii) The next focus is the verification part: a new algorithm for probabilistic simulation of system behavior and the energy and memory resources supervision is implemented to guarantee its non-violation. A new model checking approach dealing with uncompleted and adaptive systems is also presented. A mapping algorithm connecting GR-TNCES models to PRISM model checker is implemented here. (iii), all these previous contributions were integrated in a new software tool, baptized ZIZO1.1, which permits modeling, simulating and verifying reconfigurable real-time control tasks. Finally, the relevance of the methodology is proved by applying it to case studies to prove its effectiveness.


1. Introduction
2. State of the Art
Probabilistic Reconfigurable Systems
Modeling of DES
Temporal Logic
Formal Verification
3. Modeling & Specification
GR-TNCES
Specification Approach
Case Study: Skid Conveyor
4. Formal Verification
Simulation
Formal Verification
5. New Environment for Modeling, Simulation and Formal Verification
New Environment ZIZO1.1
Case Study 1: Skid Conveyor System
Case Study 2: IPV4 ZeroConf
6. Conclusion

1.4 Outline

In Chapter 2, we present the state of the art in several areas on which we work throughout this thesis. We recall basic definition and properties of probabilistic reconfigurable systems and modeling formalisms. We explore as well as computation tree logic and its extensions and introduce the subject of model checking. We also introduce case studies to apply our contributions.

Chapter 3 defines a new Petri nets-based formalism to model probabilistic reconfigurable systems running under resources constraints. In addition, a new specification approach is proposed here and applied to a skid conveyor system.

Chapter 4 proposes a new CTL profile, baptized XCTL (Reconfigurable CTL), to formally verify flexible control systems. The profile is presented through a marking algorithm and a verification mode. The chapter also presents an automatic transformation of R-TNCES to PRISM to support model checking.

Chapter 5 discusses the implementation steps of ZIZO which is a new Petri nets-based editor and probabilistic-simulator. It exposes the models of various systems and evaluates the results of the simulated models.

In Chapter 6, the results are discussed before we conclude. Future improvements that could enrich our work during this dissertation are proposed.

1.5 Publications

The outcomes of this thesis are published in the hereafter list of publications:

Journal papers and book chapters:

- ✓ Khlifi, Oussama; Mosbahi, Olfa; Khalgui, Mohamed; Frey, Georg; Li, Zhiwu: Modeling, Simulation and Verification of Probaliistic Reconfigurable Discrete-Event Systems under Energy and Memory Constraints, Iranian Journal of Science and Technology, Transactions of Electrical Engineering, 2018, **IF= 0.51.**
- ✓ Khlifi, Oussama; Siegwart, Christian; Mosbahi, Olfa; Khalgui, Mohamed; Frey, Georg: From Specification to Implementation of An Automotive Transport System, submitted in "Communications in Computer and Information Science (CCIS)" published by Springer, 2018, https://doi.org/10.1007/978-3-319-93641-3_3,

Conference Papers:

- ✓ Khlifi, Oussama; Mosbahi, Olfa; Khalgui, Mohamed; Frey, Georg: New Verification Approach for Reconfigurable Distributed Systems, The 12th International Conference on Software Engineering and Applications (ICSOFT 2017), Madrid, Spain, 2017, **Class B core ranking system.**
- ✓ Khlifi, Oussama; Siegwart, Christian; Mosbahi, Olfa; Khalgui, Mohamed; Frey, Georg: Specification Approach Using GR-TNCES -Application to an Automotive Transport System-, The 12th International Conference on Software Engineering and Applications (ICSOFT 2017), Madrid, Spain, 2017, **Class B core ranking system.**
- ✓ Khlifi, Oussama; Siegwart, Christian; Mosbahi, Olfa; Khalgui, Mohamed; Frey, Georg: Modeling and Simulation of an Energy Efficient Skid Conveyor using ZIZO, 13th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2016), Lisbon, Portugal, 2016.
- ✓ Khlifi, Oussama; Mosbahi, Olfa; Khalgui, Mohamed; Frey, Georg: GR-TNCES: New Extensions of R-TNCES for Modelling and Verification of Flexible Systems under Energy and Memory Constraints, 10th International Conference on Software Engineering and Applications (ICSOFT-EA 2015), pp. 373-380, Colmar/Alsace, France, 2015, **Class B core ranking system.**

1.6 Collaboration

This research work is carried out within Tunisia Polytechnic School and Saarland University. A part of the research is supported by the "Zentrum für Mechatronik und Automatisierungstechnik" (ZeMA) in Saarbrücken, Germany.

Chapter 2
State of the art

Contents

2.1 Probabilistic Reconfigurable Systems under Resources Constraints……..9
2.1.1 Probabilistic Reconfigurable Systems……...……9
2.1.2 Energy and Memory Resources…….....10
2.2 Modeling of Discrete Event Systems……12
2.2.1 Statecharts ……12
2.2.2 Timed Autamata ……..….....13
2.2.3 Net Condition/Event Systems …….....14
2.2.4 Timed Net Condition/Event Systems ……....15
2.2.5 Reconfigurable Timed Net Condition/Event System …..……....16
2.2.6 Tools Modeling Petri Nets……...17
2.3 Temporal Logic……18
2.3.1 Computation Tree Logic……...18
2.3.2 Extended Computation Tree Logic…….....18
2.3.3 Timed Computation Tree Logic……..19
2.3.4 Probabilistic Computation Tree Logic…….....19
2.4 Formal Verification…….....20
2.4.1 Model Checking…….....20
2.4.2 PRISM Model Checker…….....22
2.5 Case Studies…….....23
2.5.1 Automotive Transport System.....23
2.5.2 IPV4 ZeroConf……..26
2.6 Summary…….....27

STATE OF THE ART

This dissertation deals with formal modeling and verification of probabilistic adaptive discrete event control systems. All independent innovations related to modeling are based on the formalism Reconfigurable Timed Net Condition/Event System (R-TNCES) [Zhang 2013] and they are applied to different case studies. For a better understanding of this dissertation, relevant elemental knowledge on modeling formalisms, temporal logic and model checking technologies [Norman 2013] are recalled in this chapter. The case studies are also introduced at the end.

2.1 Probabilistic Reconfigurable Systems under Resources Constraints

We present in this section the characteristics of probabilistic adaptive discrete event systems running under energy and memory constraints.

2.1.1 Probabilistic Reconfigurable Systems

Reconfigurable systems topic is an earlier issue, historically; various researches were initiated by aircraft flight control systems for the purpose of fault-tolerance [Steinberg 2005]. The aim was to provide "self-repairing" capability to build a safe landing in the event of severe faults in the aircraft. Such efforts have been launched mainly after two aircraft accidents in the late 1970s [Zhang 2008]. A recent study provides other reasons for the need of reconfigurable control systems. It shows that the fatal crash of EL AL Flight 1862 of a Boeing 747-200F freighter could have been avoided if reconfigurable technologies could be applied [Zhang 2015]. Therefore, such a system is highly desirable for various aircrafts. Since the Three Mile Island incident and the accident at the Chornobyl nuclear power plant on April 26 1986, interests in diagnosis and fault tolerant control have been intensified [Zhang 2015]. Similar research works had appeared with the initial research on reconfigurable control and self-repairing flight control systems [Chandler 1984], [John 1985]. More recently, with the development of communication technologies and computer science, dynamic and static reconfiguration got more and more attention in industrial and academic communities, due to increased safety and reliability demands beyond what a conventional control system can offer. Dynamic reconfigurable systems are no longer limited to high-end systems such as aerospace and nuclear power systems [Zhang 2015]. Various products are increasingly dependent on microelectronic/mechatronic systems, onboard communication networks, and software that require new algorithms for achieving dynamic reconfigurability. The task of installing a new configuration is defined as reconfiguration of an adaptive system [Kumar 2015], [Murata 2002]. In general, reconfiguration methods can be divided into two groups: predictable and unpredictable configurations. In the predictable reconfiguration, only one new post-configuration is selected and is used in a specific algorithm to reach further states

up to an improvement point considering the constraints of the problem. In the unpredictable method, an algorithm is used for solving the problem and a large number of probabilistic possibilities are obtained, among which the most appropriate is selected as the final configuration [Khlifi 2015].

2.1.2 Energy and Memory Resources

In many-core systems, memory and energy resources must be even more abundant than processing elements [El-kustaban 2012]. Memory and energy control strategies [El-kustaban 2012] are essential for the resource-constrained systems such as wireless sensor network nodes (WSN) [Gustavo 2017], [Shareef 2010] and [Gasmi 2016]. The resources control and optimization strategies affect the life-time of WSN nodes and make the multithreaded OS feasible to run on memory-constrained WSN nodes. For the goal of improving the overall performance and energy savings, researchers introduced various approaches for resources management. e.g., memory clustering approach that acts on the addressed space of all running applications. Moreover, its running application has different processing needs and then different priorities for resources consumption [Gustavo 2017]. Thus, the resources availability is of great interest and any resources violation could stop the whole or partial system. For these critical systems, an adaptation/reconfiguration process should be done only after checking the resources' availability. As presented in [Daniel 2016], the memory system of a modern embedded processor consumes a large fraction of total system energy. Different configuration options are explored showing that a reconfigurable design can make better use of the available resources than any fixed implementation and provide a large improvement in both performance and energy consumption. Reconfigurability [Salem 2014] becomes increasingly useful in constrained resources, so it is particularly relevant in the embedded space. For an optimized architectural configuration, it has been showing that a reconfigurable cache system performs an average of 20% (maximum 70%) better than the best static implementation when two programs are competing for the same resources, and reduces cache miss rate by an average of 70% (maximum 90%) [Daniel 2016]. A case study of the Advanced Encryption Standard is presented and found that a custom memory configuration can almost double performance, with further benefits being achieved by specializing the task of each core when parallelizing the program [Daniel 2016]. Moreover, the performance of the memory system correlates with the performance of running applications and with their energy efficiency (a higher hit rate means less data movement and fewer off-chip memory accesses) [Daniel 2016].

Equipment in the manufacturing industry often display modes of operation in which less energy is used. The simplest way is *On*/*Off* equipment. Other equipments can deal with more possibility and flexibility such as pause/idle modes where it is possible to switch the equipment without turning it *Off* completely [Boussahel 2016]. High availability requirements are crucial in industry where some strict deadlines have to be respected in

order to achieve the demands; if an equipment is switched to an energy-efficient mode, then it is expected to be switched back *On* mode correctly on time when the production is restarted. An interesting application based on PROFIbus has been presented [PROFIBUS 2011] which enables switching operations on a technological level and facilitates the integration of the operations in an environment where all equipments could be turned to the target mode over the network without adding external hardware or doing it manually [Boussahel 2016]. However, this makes only a possible technological solution for the problem; PROFIenergy only addresses the issue on a software-based level but it does not offer any framework for the modeling concepts related to the optimization [PROFIBUS 2011].

Related to the formalization step, the specification has to be clarified by introducing the necessary assumptions to meet all requirements of the practical problem. Given a certain *idle* time of a single entity between two operational shifts in a fixed manufacturing scenario, let us present the following problem. An *idle* time is a time interval where the entity is, not needed to be in the run mode for any given reason. Under the first assumption that the instant power consumption of the energy consuming unit according to this manufacturing scenario is well identified and representative of the real power consumption of the studied entity, and under the further assumption that the time intervals needed for the entire necessary switching are well identified. The question is whether it is possible to regroup some information to know if it is needed to switch this entity into a power saving mode and switch it back to its operational mode whenever needed. A model is considered a mathematical abstraction of the real system needed for an evaluation and optimization purposes. If this is theoretically possible to achieve and if it is reasonable to do that on a practical level (the technological aspect is crucial when it comes to short timing constraints: for instance too many switching operations or more generally said when some safety-related issues have to be considered), then reducing the energy consumption during *idle* times to improve the energy efficiency of the single entity according to the proposed manufacturing scenario.

The improvement is to put into perspective the parallel scenario where the entity is left on its operational mode (i.e., not doing anything besides waiting for the start of the next shift) and the scenario where some switching operations are performed [Boussahel 2016]. The second question clarifies the possibility of enabling the model of a single entity to meet the required entities in a manufacturing system. Under some assumption that the process-related dependencies between all entities are well identified, the models are enhanced with additional information that are required to express all of these relations in order to form a global descriptive model. This model encloses all of the parametric information (related to time, energy consumption and inter-dependability) despite the high complexity inherent to such systems. At the end, this model should be used for the evaluation and optimization to improve the energy efficiency of the whole system [Boussahel 2016].

2.2 Modeling of Discrete Event Systems

Modeling real-time systems is a widely discussed topic in the literature. Various important formalisms were proposed around this subject. The most significant ones are cited in the following: timed process algebras [Davies 1992], [Nicollin 1990], [Pedro 1996], [Hansson 1994], timed Petri net models [Sifakis 1980], [Berthomieu 1991], real-time logics [Alur 1991] duration calculus [Chaochen 1991], timed automata [Alur 1990], state-based approaches [Alur 1993]. Concerning timed automata, scheduling was explored in [Abdeddaim 2006] and the extensions proposed for optimal scheduling including an additional cost function were investigated in [Alur 2001], [Behrmann 2001] and [Bai 2016]. We try in this section to present some of them.

2.2.1 Statecharts

The statecharts language [Klotzbücher 2012] is defined for the specification complex reactive systems [Zhang 2018], [Chan 2001]. It is a graphical representations of discrete-state transition systems [Wasserman 1985], [Harel 1987] based on hierarchical state machines. The state of a statechart structure is commonly referred to a configuration and the state transition [Wasserman 1985] is referred to a step. Steps occur in response to input events [Qianchuan 2006]. There are various semantic interpretations for statecharts thanks to the possibility of having multiple states enabled at one time. Moreover, the changes in variables and configurations can introduce a new enabled transition. Various step-semantics have been introduced to define the steps for a given statechart structure. RSML (Requirements State Machine Language) is another language based on statecharts with slightly different syntax and semantics [Leveson 1994]. They both extend state-machine diagrams with parallelism, super-states, and broadcast communications. The STATEMATE toolset implements a particular semantics of statecharts [Chan 2001], [Harel 1990]. It offers a system model based on a finite number of parallel local state machines with a finite set of events and inputs interacting with a nondeterministic environment.

Figure 1 [Chan 2001] presents a simple example with two parallel state machines A and B which are synchronized using events. The initial local states are represented by arrows without sources. Other arrows indicate local transitions which are identified with the form *trig* [*cond*]/*acts*, where *trig* is a trigger event, *cond* is an optional guarding condition, and *acts* is a (possibly empty) list of action events. The guarding condition is a predicate on local states of other state machines and/or inputs to the system. If the trigger event occurs and the guarding condition either is absent or is evaluated to true, then the transition is enabled. Initially, some external events, along with some inputs from the environment, arrive, marking the beginning of a step. The system leaves the source local states, enters the destination local states, and generates the action events (if any). These events are used to enable some transitions as described above. A statechart responds to a set of input events by firing a sequence of chart-transitions, and then the system configuration and variables

change accordingly. A complete transition in a statechart state in response to an input event is called a step. Configurations are defined as maximally consistent sets of chart-states, which are sets of chart-states satisfying the following conditions: the root state is included; each AND-state in the configuration includes all of its children, and each OR-state included in the configuration also includes exactly one of its children.

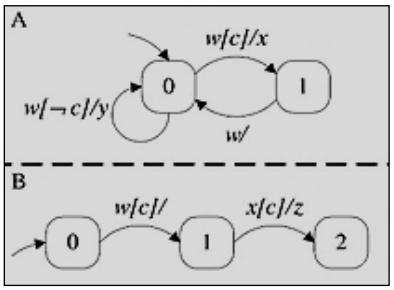


Figure 1 Statechart example

2.2.2 Timed Autamata

Here, we try to understand the extent of what could be done with timed automata where only time is considered as a constraint above the basic reachability aspect [Ma 2017a]. Time is a variable that increases according to a fixed positive rate. It could be considered as an additional constraint when strengthening the automaton model. Reachability of such a state consists on finding a path in a transition system between an initial and the target state. The reachability problem can be then enhanced with further constraints considering that the automaton exhibits more variables in its definition. Timed automata are an appropriate approach for scheduling problems [Wu 2016]. A scheduling problem is the general problem of a set of tasks to be performed using different resources. The main task is their duration, the resources they need and the precedence relationships they have with other tasks. A schedule is a solution to a task execution; different schedules lead to different solutions and optimal scheduling is the problem of finding the best schedule according to a given criteria. The scheduling problem [Wu 2012b] is oriented towards alternative strategies in manufacturing systems for the purpose of improving their energy-efficiency. The important aspects to highlight are considering temporal operations with time-dependability and process-related dependencies. The discrete-event systems theory allows dealing with this problem in the sense that temporal switching behaviors can be represented by automata. States of the automata allow to model operation modes while transitions permit modeling all the additional constraints. Figure 2 shows an example of timed automaton, i.e., the timing behavior of the automaton is controlled by two clocks x and y. The clock x is used to control the self-loop in the location loop. The single transition of the loop may occur if x=1. The clock y controls the execution of the entire automaton. The automaton may leave start at any time point when y is in the interval between 10 and 20; it can transit from loop to end when y is between 40 and 50, etc.

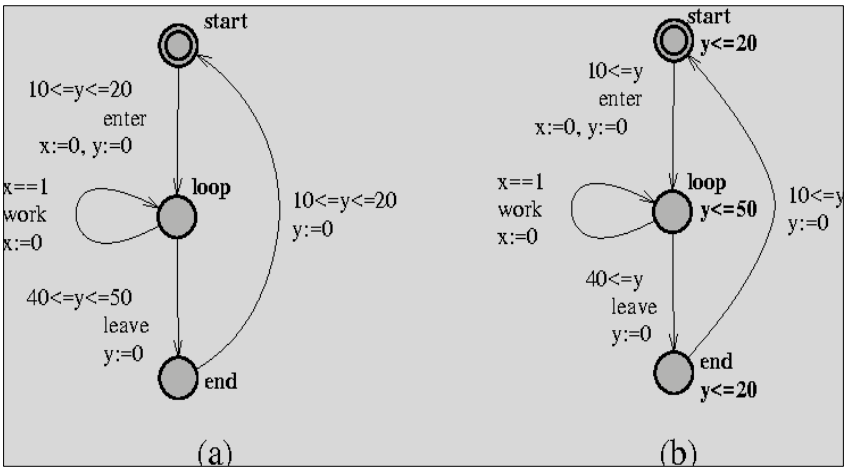


Figure 2 Timed Automaton Example

2.2.3 Net Condition/Event Systems

The formalism of Net Condition/Event Systems (NCES) is an extension of the well-known Petri net formalism [Ye 2015], [Wu 2015]. It was introduced by Rausch and Hanisch in [Rausch 1995] and further developed through the last years, in particular in [Hanisch 1999], A basic module of an NCES is a typical Petri net, i.e., it is composed of places, transitions, flow arcs, and tokens. Each basic module of an NCES interconnects with other modules via special condition/event signals, which make an NCES different from a Petri net, see Figure 1. An NCES is a place-transition net formally represented by a tuple:

$NCES = (P, T, F, CN, EN, C_{in}, E_{in}, C_{out}, E_{out}, B_c, B_e, C_s, D_t, m_0)$ where :

- P (resp. T) is a non-empty finite set of places (resp. transitions),
- F is a set of flow arcs, $F : (P \times T) \cup (T \times P)$,
- CN (resp. EN) is a set of condition (resp. event) arcs, $CN \subseteq (P \times T)$ (resp. $EN \subseteq (T \times T)$),
- C_{in} (resp. E_{in}) is a set of condition (resp. event) inputs,
- C_{out} (resp. E_{out}) is a set of condition (resp. event) outputs,
- B_c (resp. B_e) is a set of condition (resp. event) input arcs in a NCES module,
- $B_c \subseteq (C_{in} \times T)$ (resp. $B_e \subseteq (E_{in} \times T)$),
- C_s (resp. D_t) is a set of condition (resp. event) output arcs,
- $C_s \subseteq (P \times E_{out})$ (resp. $D_t \subseteq (T \times E_{out})$),
- $m_0 : P \rightarrow \{0, 1\}$ is the initial marking.

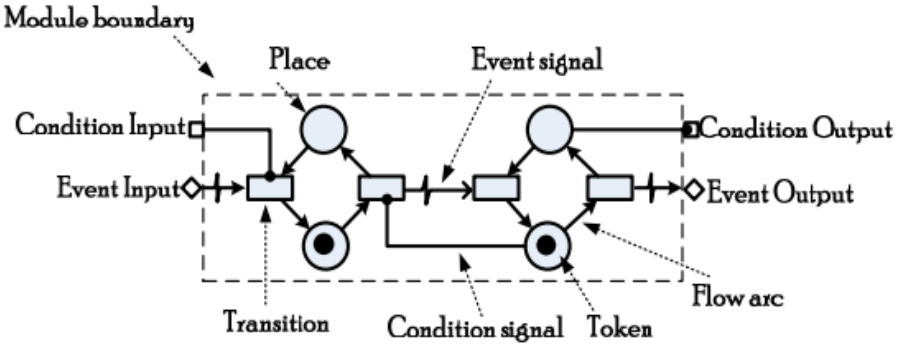


Figure 3 A composite module of an NCES

2.2.4 Timed Net Condition/Event Systems

This formalism was introduced by [Hanisch 1997] and it was extended to consider time constraints that are applied to input arcs of transitions: to every pre-arc of a transition, an interval [*eft*, *lft*] of natural numbers is attached with $0 \le eft < lft \le w$ (w is a given integer). The interpretation is as follows: Every place p bears a clock that is running (*resp*, switched) if the place is marked (*resp*, unmarked). All running clocks run at the same speed measuring the time of the token states. If a firing transition t removes a token from a place p or adds a token to p, the clock of p is initialized back to 0. In addition, a transition t is able to remove tokens from its pre-places (i.e., to fire) only if $\forall\ p \in {\bullet}t$, the clock at the place p shows a time $D(p)$ such that $eft(p, t) \le D(p) \le lft(p, t)$. A TNCES is a tuple :

$$\text{TNCES} = (P, T, F, m_0, \Psi, CN, EN, DC)$$

where:

- $P = \{p_1, p_2, ..., p_n\}$ is a finite set of places;
- $T = \{t_1, t_2, ..., t_m\}$ is a finite set of transitions;
- $F \subseteq (P \times T) \cup (T \times P)$ is a finite set of flow arcs between places and transitions;
- m_0 is initial marking;
- $CN \subseteq (P \times T)$ is a finite set of condition arcs;
- $EN \subseteq (T \times T)$ is a finite set of event arcs.

Ψ is input/output structure of TNCES module which is represented by the following tuple:

$$\Psi = (C_{in}, E_{in}, C_{out}, E_{out}, B_c, B_e, C_s, D_t)$$

where:

- C_{in} defines a finite set of TNCES module condition input signals;
- E_{in} defines a finite set of TCNES module event input signals;
- C_{out} defines a finite set of TNCES module condition output signals;
- E_{out} defines a finite set of TCNES module event output signals;

- $B_c \subseteq C_{in} \times T$ is a set of TNCES module input condition arcs;
- $B_e \subseteq E_{in} \times T$ is a set of TNCES module input event arcs;
- $C_s \subseteq P \times C_{out}$ is TNCES module output condition arcs;
- $D_t \subseteq T \times E_{out}$ is a set of TNCES module output event arcs. Time intervals are assigned to the pre-transition flow arcs $F \subseteq P \times T$, which impose time constrains to the firing of the transition:

$$DC = (DR, DL, D_0)$$

where:

- DR represents the set of minimum times that the token should spend at particular place before the transition can fire;
- DL is the final set of limitation time that defines maximum time that the place may hold a token (if all the other conditions for transition firing are met);
- D_0 is the initial set of the clocks associated with the places.

2.2.5 Reconfigurable Timed Net Condition/Event System

Reconfigurable control systems are characterized by clear modular structure. Reconfigurable timed net condition/event system (R-TNCES) [Zhang 2013] is such a modular formalism that was developed for modeling and analyzing adaptive distributed control systems [Bastide 1998], [Kumar 2015]. Assuming that an industrial control system is expected to be reconfigurable; it means that the controllers of its distributed physical components should be able to change themselves actively. According to the changed execution environment or the user requirements, these controllers should be able to be standby, activated, or even be removed from the system. In addition, they should also be able to change their connection relation with other controllers, or be able to modify their own behavior modes, or just update some shared data. If TNCES are applied to model such reconfigurable systems, components of TNCES such as places, transitions, flow arcs, and markings within particular basic modules or condition/event signals among these modules should be modified at run-time. This formalism focuses on dynamic reconfigurations and control of TNCES. An R-TNCES is an extension of the formalism TNCES with a specific function of self-reconfiguration [Zhang 2013, Zhang 2015]. It is defined as a structure R-TNCES = (B, R), where R is the control module consisting of a set of reconfiguration functions $R = r_1,...,r_n$ and B is the behavior module that is a union of multi TNCES, represented as:

$$B = (P, T, F, W, CN, EN, DC, V, Z)$$

where:

- P (respectively, T) is a superset of places (respectively, transitions),
- $F \subseteq (P \times T) \cup (T \times P)$ is a superset of flow arcs,

- $W : (P \times T) \cup (T \times P) \rightarrow \{0, 1\}$ maps a weight to a flow arc, $W(x, y) > 0$, if $(x, y) \in F$, and $W(x, y)=0$ otherwise, where $x, y \in P \cup T$,
- $CN \subseteq (P \times T)$ (respectively, $EN \subseteq (T \times T)$) is a superset of condition signals (respectively, event signals),
- $DC : F \cap (P \times T) \rightarrow \{ [l_1, h_1], ..., [l\ |F \cap (P \times T)|, h\ |F \cap (P \times T)|] \}$ is a superset of time constraints on output arcs, where $i \in [1, |F \cap (P \times T)|]$, $l_i, h_i \in N$, and $l_i < h_i$,
- $V : T \rightarrow \{\vee, \wedge\}$ maps an event-processing mode (AND or OR) for every transition,
- $Z = (M_0, D_0)$, where $M_0 : P \rightarrow \{0, 1\}$ is the initial marking and $D_0 : P \rightarrow \{0\}$ is the initial clock position.

2.2.6 Tools Modeling Petri Nets

Several tools already exist to model and/or simulate Petri nets [Chen 2015] and their extensions. For example, CPN tools is a software for editing, simulating and analysing Coloured Petri Nets [Ratzer 2003]. It features a fast simulator that efficiently handles both untimed and timed nets. Full and partial state spaces can be generated and analysed, and a standard state space report contains information such as boundedness properties and liveness properties [Ratzer 2003] [Koh 1991]. Petri.NET is another tool which allows modeling, simulation and real-time implementation of static and dynamic Petri nets [Genter 2007]. The results of a Petri net model simulation are presented to the user in the form of a token game and in the graphical form showing diagrams of a state vector. Nevertheless, neither CPN tools nor Petri.NET can support R-TNCES with their condition and event signals. The Visual Verification (ViVe) toolset is a tool chain for automatic verification of distributed control systems. It allows creation and modification of model components in modelling language of Net Condition/Event Systems (NCES) [Suender 2011]. Nevertheless, it does not deal with the time constraints in NCES and the reconfiguration features they may have. The TNCES-Editor, developed at the Martin Luther University Halle-Wittenberg, allows the graphical modeling of all NCES based subtypes, including R-TNCES [Dubinin 2006]. To support interpretation and reachable state analysis, the TNCES-Editor offers an optional labeling of transitions. The whole net structure including the labels will be stored in a special file format (*.pnt) which can be used as an import file for the model-checker SESA [Starke 2002] for the formal verification. However, TNCES-Editor doesn't feature the simulation of a built R-TNCES, nor highlights the reconfiguration aspect.

2.3 Temporal Logic

Computation Tree Logic (CTL) offers facilities for the specification of properties to fulfill the system behavior [roch 2000a, roch 2000b]. We present here this logic and three of its extensions: Extended Computation Tree Logic (ECTL) [Axelsson 2010], the Timed Computation Tree Logic (TCTL) [Bouyer 2007] and Probabilistic Computation Tree Logic (PCTL) [Brázdil 2008]. This logic is used to formally prove that the specification satisfies the desired properties of the product.

2.3.1 Computation Tree Logic

In CTL, all formulae specify behaviors of the system starting from an assigned state in which the formula is evaluated by taking paths (e.g. sequence of states) into account. The semantics of formulae is defined with respect to a reachability graph where states and paths are used for the evaluation. A reachability graph M consists of all global states that the system can reach from a given initial state. It is formally presented as a tuple $M = [Z; E]$ where:

- Z is a finite set of states,
- E is a finite set of transitions between states, e.g. a set of edges $(z; z_0)$, such that $z, z_0 \in Z$ and z_0 is reachable from z.

In CTL, paths play a key role in the definition and evaluation of formulae. A path denoted by (z_i) starting from the state z_0 is a sequence of states, $(z_i) = z_0, z_1 ...$ such that $\forall\, j \in \mathbb{N}$, there is an edge $(z_j; z_{j+1}) \in E$. The truth value of a CTL formula is evaluated with respect to a certain state of the reachability graph. Let $z_0 \in Z$ be a state of the reachability graph and φ be a CTL formula. The relation $z_0 \models \varphi$ means that the CTL formula φ is satisfied in the state z_0. Then the relation $\models$ for a CTL formula is defined as follows:

- $z_0 \models EF\varphi$, if there is a path (z_i) and $j > 0$ such that $z_j \models \varphi$,
- $z_0 \models AF\varphi$, if for all paths (z_i), there exists $j > 0$ such that $z_j \models \varphi$,
- $z_0 \models AG\varphi$, if for all paths (z_i) and for all $j > 0$, it holds $z_j \models \varphi$.

2.3.2 Extended Computation Tree Logic

In CTL, it is rather complicated to refer to information contained in certain transitions between states of a reachability graph. A solution is given in [Roch 2000a, Roch 2000b] for this problem by proposing an extension of CTL called Extended Computation Tree Logic ECTL. A transition formula is introduced in ECTL to refer to the transition information contained in the edges of the reachability graph. Since it is wanted to refer not only to the state information but also to the steps between states, the structure of the reachability graph $M = [Z, E]$ is changed as follows:

- Z is a finite set of states,
- E is a finite set of transitions between states, e.g. a set of labeled edges (z, s, z_0), such that $z, z_0 \in Z$ and z_0 is reachable from z by executing the step s.

Let $z_0 \in Z$ be a state of the reachability graph, τ a transition formula and φ an ECTL formula. The relation for ECTL formulae is defined inductively:

- $z_0\ E\tau X\varphi$: if there exists a successor state z_1 such that there is an edge $(z_0, s, z_1) \in E$ where $(z_0, s, z_1)\ \tau$ and $z_1\ \varphi$ holds,

- z_0 $A\tau X\varphi$: if z_1 φ holds for all successors states z_1 with an edge $(z_0, s, z_1) \in E$ such that (z_0, s, z_1) τ holds.

2.3.3 *Timed Computation Tree Logic*

TCTL is an extension of CTL to model qualitative temporal assertions together with time constraints. The extension focuses on attaching a time bound to the modalities and we note that a good survey can be found in [Alur 1991]. For a reachability graph $M = [Z, E]$, the state delay D is defined as a mapping $D: Z \rightarrow N_0$ and for any state $z = [m, u]$ the number $D(z)$ is the number of time units which have to elapse at z before firing any transition from this state. For any path (z_i) and any state $z \in Z$, we put:

- $D\ [(z_i, z)] = 0$, if $z_0 = z$,
- $D\ [(z_i, z)] = D(z_0) + D(z_1) + ... + D(z_{k-1})$, if $z_k = z$ and $z_0, ..., z_{k-1} \neq z$.

In other words, $D\ [(z_i, z)]$ is the number of time units after which the state z on the path (z_i) is reached the first time, e.g. the minimal time distance from z_0. Let $z_0 \in Z$ be a state of the reachability graph and φ a TCTL formula. The relation for TCTL is presented as follows:

- z_0 EF $[l, h]$ φ, if there is a path (z_i) and a $j > 0$ such that z_j φ and $l \leq D((z_i), z_j) \leq h$,
- z_0 AF $[l, h]$ φ, if for all paths (z_i) there is a $j > 0$ such that z_j φ and $l \leq D((z_i), z_j) \leq h$.

2.3.4 *Probabilistic Computation Tree Logic*

Probabilistic Computation Tree Logic (PCTL) is the established temporal logic for probabilistic verification of discrete-time Markov chains [Brázdil 2008]. The probabilistic branching-time logic PCTL was introduced by Hans Hansson and Bengt Jonsson in 1994 [Hansson 1994]. It was subsequently used for probabilistic model checking [Bianco 1995], [Christel 2008] and is now widely used in probabilistic model-checking tools, for example in PRISM [PRISM 2015] and Verus [Christel 1997], [Sérgio 1995]. Hansson and Jonsson define PCTL without the Next modality "X". Leslie Lamport even argues in [Leslie 1983] that the Next modality should be excluded from any temporal modal logic. Let $M = (S, P, L)$ be a Markov chain. The semantics of PCTL is defined inductively as:

$$\llbracket q \rrbracket = \{s \in S \mid L(s, q) = tt\}$$
$$\llbracket \phi \wedge \psi \rrbracket = \llbracket \phi \rrbracket \cap \llbracket \psi \rrbracket$$
$$\llbracket \neg \phi \rrbracket = S \setminus \llbracket \phi \rrbracket$$
$$\llbracket [\alpha] \bowtie p \rrbracket = \{s \in S \mid Prob_M(s, \alpha) \bowtie p\}$$

where $Prob_M(s, \alpha)$ is the probability of the measurable set $Path(s, \alpha)$ of paths in M that begin in s and satisfy the path formula α where the semantics for path formulae is as follows:

- $\pi \models X \phi$ if $\pi[1] \in \llbracket \phi \rrbracket_M$;

- $\pi \models \phi U^{\leq k} \psi$ if there is a $l \in \mathbb{N}$ such that $0 \leq l \leq k$, $\pi[l] \in [\![\psi]\!]_M$ and for all $0 \leq j < l$, we have $\pi[j] \in [\![\phi]\!]_M$;
- $\pi \models \phi W^{\leq k} \psi$ if for all $l \in \mathbb{N}$ such that $0 \leq l \leq k$, we have either $\pi[l] \in [\![\phi]\!]_M$, or there is $0 \leq j \leq l$, with $\pi[j] \in [\![\psi]\!]_M$.

Occasionally, we need to assert that a PCTL (sub-) formula ϕ holds at a state s of a Markov chain M. We denote this $s \models \phi$, and we write $s \models_M \phi$ to clarify which Markov chain M the state s belongs to.

2.4 Formal Verification

We presented specification facilities to specify and verify such a probabilistic, functional or temporal property. The mathematical basis of formal method provides a way to deal with abstraction, modularity and hierarchy with typical engineering problems such as quality goals, optimization and maintainability. Manufacturing systems are man-made systems that exhibit complex structures in order to deal with problems related to modeling and evaluation. The theory of discrete-event systems is recommended to tackle such systems; a discrete-event system is understood to be a system in which states evolve according to the occurrence of asynchronous events [Wang 2016]. Formal methods [Zedan 1999] often supported by tools, allows for a deep understanding of manufacturing systems in order to improve their reliability with the help of verification and validation of properties.

2.4.1 Model Checking

Model checking was introduced by Clarke and Emerson [Clarke 2008]. The Model checking problem can be stated according to these authors, which are one of the pioneers in this field, as:

"*Let M be a Kripke structure (i.e., state-transition graph). Let f be a formula of temporal logic (i.e., the specification). Find all states s of M such that M, s* $\models$ *f.*" [Clarke 2008].

The expression "$s \models f$" represents that the state s satisfies the property f. The term Model was used in the sense of whether the structure M was a model for a formula f. Model Checking is a verification technique in which all possible system states are explored. It is a technique to automatically verifying the correctness of properties of finite-state systems [Rouff 2012] [Clarke 1986]. A general verification approach that is applicable to a wide range of applications such as embedded systems, software engineering, and hardware design. It also supports partial verification, i.e., properties can be checked individually, thus allowing focus on the essential properties first. Model-checking is a potential "push-button" technology; the use of model checking requires neither a high degree of user interaction nor a high degree of expertise. It can be also easily integrated in existing development cycles since its learning curve is not very steep, and empirical studies indicate that it may lead to shorter development times [Baier 2008].

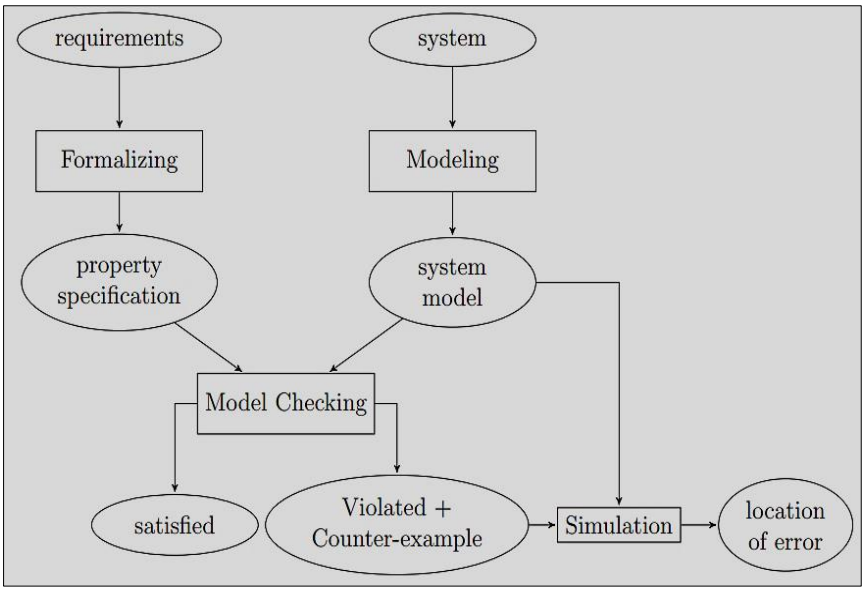


Figure 4 Schematic view of the model-checking approach, [Baier 2008]

The goal is to prove, formally, that all possible executions of the system are conform to the requirements. The generation of the system model comes from a model description which translates how the system behaves while the property specification prescribes all the properties of the system, namely what it should or not do, see Figure 4 for the general approach schematic view. Different phases appeared when applying model checking: Modeling, running and analysis phase [Ross 1997]. The system is modeled using the model description language of the model checker [Model 2007] and properties are formalized to be checked using the property specification language. The model checker is running in order to check the validity of the properties. The analysis presents different possibilities: the property can be satisfied, violated or the model is too big to analyze; see the schematization in Figure 5 adapted from the aforementioned phases [Baier 2008]. The advantages of Model Checking are many and can be summarized in the following: no proofs are needed and the checking process is automatic. Counter-examples are given if the property is not satisfied. Moreover, partial specifications are allowed and temporal logics are of an advantage when reasoning about concurrent systems [Clarke 2008]. In addition, it is a general verification approach, applicable to diverse applications and it has a strong mathematical foundation (it is based on the theory of graph algorithms, logic and data structures) [Baier 2008]. The disadvantages are not to be underestimated: there is no guarantee of completeness provided that only the stated requirements are checked, and the obtained results are also as good as the model itself (the actual system is not verified but only its model is verified) [Baier 2008].

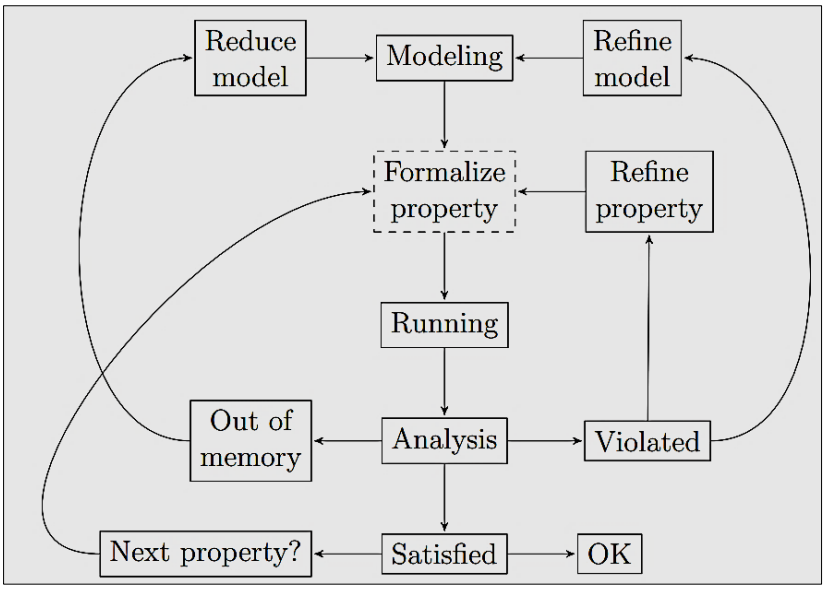


Figure 5 Characteristics of Model Checking, adapted from [Baier 2008]

Finally, the state explosion problem is a major problem, i.e., the total number of concurrent system states with complicated data structures can be enormous. Various researches have been dedicated to tackle this particular problem since the beginning of model checking.

2.4.2 PRISM Model Checker

PRISM is a probabilistic model checker [PRISM 2015] which offers a formal verification method for the analysis of quantitative properties of stochastic systems [PRISM 2015]. The official website offers a user's manual and some basic use case examples. Two user interface types are offered: command line and GUI (Graphical User Interface). A text editor, property editor and plot capability are offered by the GUI which is user-friendlier interface. It is an open source tool, developed using Java/C++. The user interface and parsers are written in Java whereas the core algorithms are implemented for the most part in C++. The PRISM programming language is a high-level state-based description language based on the reactive modules formalism. Each module is determined by a set of finite-range variables and a guarded-command based notation describes its behaviour. Each system is presented as a parallel composition of a set of modules. Global variables or synchronization over common action labels are added for the communication between modules. Several types of probabilistic models are supported: Discrete-Time Markov Chains (DTMCs), Continuous-Time Markov Chains (CTMC), Markov Decision Processes (MDP), Probabilistic Timed Automata (PTA) [PRISM 2015]. See Figure 6 for an example of a PRISM program where the

```
mdp      // model type
rewards // reward block
x=0: 10;  // an assigned reward
endrewards
module M1 // module M1 definition

    x : [0..2] init 0; // variable definition

    [s1] x=0 -> 0.8:(x'=0) + 0.2:(x'=1); // a line:
       guard->actualization
endmodule
```

Figure 6 Example of a program in PRISM

properties of the models are written in the PRISM language: PCTL (probabilistic computation tree logic), CSL (Continuous Stochastic Logic), LTL (Linear Time Logic), PCTL* (which uses both PCTL and LTL) [PRISM 2015]. A choice is offered between the following data structures for model checking: MTBDD (Multi-Terminal Binary Decision Diagram)/BDD, Sparse Matrix, Hybrid (a combination of MTBDD and Sparse Matrix). The analysis of probabilistic models is based on some mathematical logics in order to evaluate its properties. The so-called property specification language uses the temporal logics: CTL, PCTL, probabilistic LTL and PCTL*: PCTL is used for DTMCs, MDPs and PTAs, and CSL (an extension of PCTL) is used for CTMCs. For more information on the syntaxes and semantics related of these logics, some key works are a good introduction to this topic such as: [Clarke 2008] for CTL, [Baier 1998] for LTL and PCTL*, [Aziz 1996] for CSL and [Hansson 1994], [Bianco 1995] for PCTL. A property is used for different features such as identifying a particular set of states, probabilities and rewards. The question is to get an answer about whether for a certain model a property is true or false, or to get an evaluation consisting of a numerical value.

2.5 Case Studies

We present in this section two case studies where our contributions will be applied. An automotive transport system [Khlifi 2016] will be introduced for the aim to save energy of the current system model, i.e., we would like to optimize its energy consumption. An IPV4 protocol [Bohnenkamp 2003] is presented to show how GR-TNCES and ZIZO1.1 could present an optimal model while guaranteeing the non-violation of energy and memory resources.

2.5.1 Automotive Transport System

In PROFIenergy, the term 'Energy Consuming Unit' is used to denote any independent equipment where energy is consumed. An energy consuming unit could represent a simple energy saving modes (such as standby, off) or various measurements of savings modes in complex machines [PROFIBUS, 2010]. An energy consuming unit is a candidate for energy

saving whenever such a mode dealing with 'pause' or 'idle time' occurs. Let us consider that 'standby' is a time period when an energy consuming unit is not operating for any given reason. These time intervals can be classified related to their duration, i.e., if it is known in advance, two scenarios can be derived: brief pauses last typically up to one hour and longer pause if it lasts longer than one hour [PROFIBUS, 2010]. PROFIenergy identifies short pauses with a maximum period of 5 minutes. Moreover, pauses that last enough time are candidates for energy saving goal and should be considered carefully. As mentioned before, the role of idle times is important for energy savings because the equipment is not used. Thus, manufacturers need to give attention to these intervals of times where there is a neglected potential because the plant has not been enough investigated. As the demand for energy increases, it causes more pollution and conducts to climate change. That's why energy efficiency come into the issue and it is defined in various ways. According to ISO 50001, energy efficiency is a "ratio or a qualitative relationship between an output of performance, service, goods or energy, and an input of energy" [International 2015]. A more intuitive operational definition of energy efficiency can be directly stated for "using less energy to provide the same level of service". It is important to identify the difference between energy efficiency and energy conservation which is reducing or going without a service to save energy. For example, turning *Off* a motor and avoiding the service is energy conservation. However, changing an old motor by a new high-efficient one is energy efficiency. The same services are met while consuming less energy. The common denominator in both cases is then saving energy. Energy efficiency is ranked high in the hierarchy of sustainable energy, see Figure 7. Moreover, reducing energy demand is hugely important: when the demand for energy on different levels is reduced, less energy can be generated in order to meet the demands.

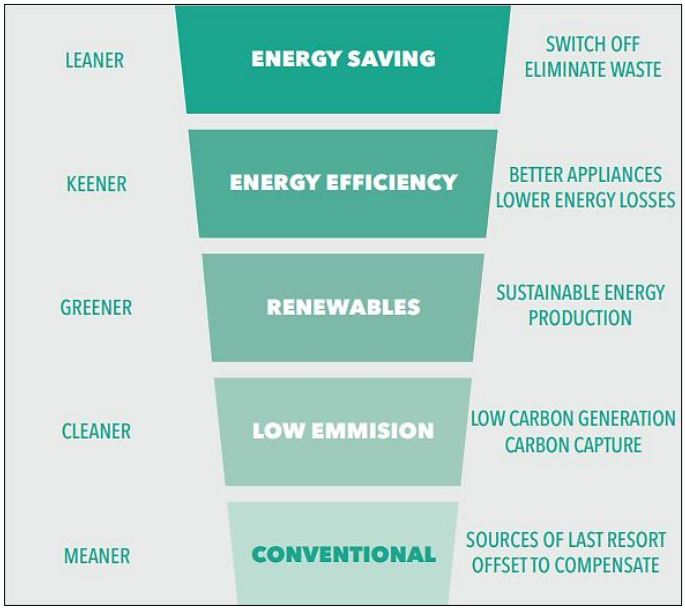


Figure 7 Sustainable energy hierarchy, [Sustainable 2014]

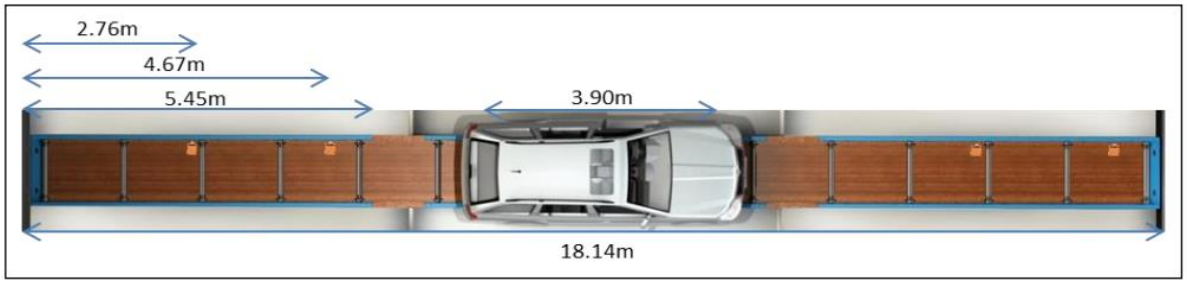


Figure 8 CAD model of the skid conveyor

Skid conveyors are widely used to move materials over a fixed path in the automotive industry. Transporting a body in the paint shop or a chassis from one workstation to another in the final assemblies are typical examples. For this purpose, we define an extended skid conveyor system showed in Figure 8, which will be one part of the automated commissioning line in the "Zentrum für Mechatronik und Automatisierungstechnik" (ZeMA) in Saarbrücken, Germany. The transport system should consist of three conveyor parts. Currently, there is an old system where all the motors are switched together and manually from one mode to another. Each one is equipped with one motor, the overall length is 18.14m and each part has a length of 5.45m. Each motor drives five rollers transporting a skid of 3.90m with a chassis on it. We aim to introduce new functional modes which offer the user to localize the chassis on every part. Moreover, in each conveyor part, the chassis should wait for a predefined time to establish other tasks by various robots. Realising these tasks should be done with the aim to minimize the consumed energy of the system during the movement of the chassis; each unused actor should be switched off or to a standby mode [Khlifi 2016].

2.5.2 IPV4 ZeroConf

Communication networks need to be error-free and efficient. In addition, they often operate under real-time constraints implying that they must meet certain deadlines in order to satisfy the quality of service. Let us assume an IPV4 ZeroConf network [Bohnenkamp 2003] of different devices such as iPhones, tablets, DVD players etc. In fact, if a new device connects to the network, it has to randomly choose an IP address from a pool of 65024 available addresses. The Internet assigned number authority has allocated the addresses from 169.254.1.0 to 169.254.254.255 for the purpose of such link-local networks. Following the standard, we suppose that it takes zero to one second to send a message between the hosts. The device has to guarantee the uniqueness of its chosen address. Thus, it sends messages to the rest of devices in the network asking whether any of them is currently using the chosen IP address. If no reply is received before two seconds, the device starts using the IP address. Otherwise, a reconfiguration scenario takes place or in certain circumstances, it

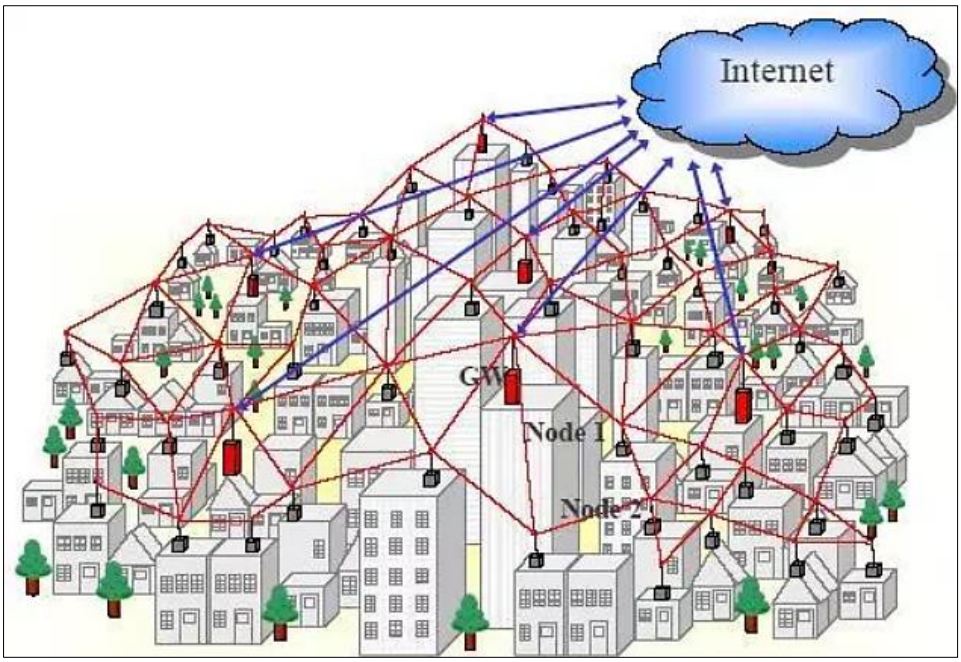


Figure 9 Example of Public Network [Gustavo 2017]

can defend its ownership of the address. The successive reconfiguration features established by the devices can bring the latter to a blocking situation that does not respect real-time properties. We note that IPV4 ZeroConf protocol is a fully connected network which is known as mesh network topologies see Figure 9. It is a topology with a point-to-point link, i.e., with N nodes, there are N*(N-1)/2 direct branches. For example, for N=500, we have 124750 connection links. This is possible for the data to be simultaneously transmitted from any node to all of the other nodes. The memory resources are essential for maintaining the network traffic since thousands of messages are transferred continuously. The energy resources are mandatory to run the protocol. In particularly, we suppose that one token of memory and energy resources are consumed to transfer each message between the devices. The connection process is described as follows: (i) A new device chooses a random IP address from a pool of 65024 addresses; (ii) It sends four ARP packets called probes. These probes contain the chosen IP address; (iii) If another device is using the chosen address, it must send ARP reply before two seconds and the new device restarts the protocol. Once sending four ARP probes and the new device does not receive an ARP reply, it starts to use the address. The new device must send a using confirmation for all the rest of devices through a gratuitous ARP (two gratuitous ARP are sent in two-second interval). (v) The new device must now respond to received ARP packets: If it receives a probe that has the same IP address, then it must send an ARP-Reply containing its address before 10 seconds. (vi) If it receives a gratuitous ARP (GARP) containing the same address, then, two scenarios could take place: once it receives the GARP in the first 10 seconds of the use of the chosen address, it must defer by restarting the protocol. Otherwise, it will defend its ownership of the address by resending GARPs.

2.6 Summary

We introduced relevant elemental knowledge on modeling-based formalisms which are not able to deal with reconfigurable probabilistic systems under memory and energy constraints, i.e., the current modelling formalisms extending Petri nets are not able to sufficiently describe all the properties of adaptive probabilistic systems. We presented also the current model checking technologies and temporal logics, nevertheless they does not cover the formal verification of reconfigurable systems and uncompleted specification.

Chapter 3

Modeling and Specification

Contents

3.1 Introduction ……29
3.2 GR-TNCES ……29
3.2.1 Motivation ……29
3.2.2 Formalization……29
3.2.3 Dynamics of GR-TNCES……31
3.3 Specification Approach ……33
3.3.1 Motivation ……33
3.3.2 System Specification ……33
3.4 Case Study: Skid Conveyor……35
3.4.1 Description……35
3.4.2 Specification ……36
3.5 Discussion ……40
3.6 Summary ……41

MODELING AND SPECIFICATION

3.1 Introduction

In this chapter, we propose to enrich the formalism Reconfigurable Timed Net Condition/Event Systems (R-TNCES) with the possibilities of modeling energy, memory and probabilistic behavior in order to model and verify the safety of unpredictable reconfiguration scenarios running under resources constraints [Khlifi 2015]. A specification approach based on this formalism is also presented and applied to a case study illustrating our contribution [Khlifi 2017a]. This chapter is detailed in the recently cited papers.

3.2 GR-TNCES

Since R-TNCES is a useful formalism to model reconfigurable systems, we aim in this section to enlarge its usability for other complex systems knowing as probabilistic distributed discrete event systems running under energy and memory constraints.

3.2.1 Motivation

Probabilistic systems have an unpredictable behavior, i.e., the whole sequence of tasks is not predefined, and it is not possible to fix the required resources in advance. Memory and Energy resources are mandatory to run such a system. Thus, before applying any reconfiguration scenario, the resources' availability should be checked. The proposed formalism tries to deal with these research problems: How can we extend Petri nets formalisms to model various features of APDECS systems? How can we specify probabilistic reconfigurations as well as memory and energy resources? How can we check if a system satisfies the available energy and memory constraints after any unpredictable reconfiguration?

3.2.2 Formalization

The formalism GR-TNCES is a network of R-TNCES introduced to model and control APDECS running under memory and energy constraints. It is a structure $G = \sum$ R-TNCES where R-TNCES = (B, R), such that R is the control module consisting of a set of reconfiguration functions {$r1$,..,rn} managed under a memory and energy controllers, and B is the behavior module which is a union of multi TNCES, represented as follows:

$$B = (P, T, F, QW, CN, EN, DC, V, Z_0)$$

where:

- P (*respectively*, T) is a non-empty finite set of places (*respectively*, transitions),

- F is a set of flow arcs with $F \subseteq (P \times T) \cup (T \times P)$,
- QW= (Q, W) where Q: $F \rightarrow [0, 1]$ is the probability on the arcs and W: $(P \times T) \cup (T \times P) \rightarrow \{0, 1\}$ maps a weight to a flow arc. Specifically, $W(x, y) > 0$ if $(x, y) \in F$, and $W(x, y)=0$ otherwise, where $x, y \in P \cup T$,
- CN (*respectively*, EN) is a set of condition (*respectively*, event) signals with $CN \subseteq (P \times T)$ (respectively, $EN \subseteq (T \times T)$),
- DC: $F \rightarrow [l, h]$ is a superset of time constraints on output arcs. F is a flow arcs with $F \subseteq (P \times T)$,
- V: $T \rightarrow \{\vee, \wedge\}$ maps an event-processing mode (AND or OR) to each transition,
- Z_0 = (T_0, D_0), where T_0: $P \rightarrow \{0, 1\}$ is the initial marking and D_0: $P \rightarrow \{0\}$ is the initial clock position.

Let $TN = P \times T \times F \times QW \times CN \times EN \times DC \times V$ be the set of all feasible net structures that can be performed by a system. Let $\bullet r$ (*respectively*, $r\bullet$) denote the original (*respectively*, target) R-TNCES before (*respectively*, after) the reconfiguration function r is applied, where $TN(\bullet r)$, $TN(r\bullet) \in TN$. Each reconfiguration is controlled by the controller module R. It is a set of structure R = {Condition $Cond$, Probability Q, Energy E', Memory M', Structure S, State X}. A reconfiguration function r is a structure r = ($Cond$, Q, E_0', M_0', S, X), where:

(i). $Cond$: $CN \rightarrow$ {true, false}: The precondition $Cond$ of r can be evaluated to either true or false and it can be modeled by external condition signals,

(ii). Q: $F \rightarrow$ [0..1]: Represents the probability to reach each TNCES branch. It could be a functional (internal to the TNCES) or a reconfiguration probability. It enables to describe the nondeterministic behavior of the system,

(iii). E_0': $P \rightarrow$ [0..max]: The energy requirements of the chosen probabilistic scenario,

(iv). M_0': $P \rightarrow$ [0..max]: The memory requirements of the chosen probabilistic scenario,

(v). S: $TN(\bullet r) \rightarrow TN(r\bullet)$: Is the structure modification instruction of the reconfiguration scenario. It contains the reconfiguration structure process, i.e., the information about the current state and the destination.

(vi). X: $\bullet r \rightarrow r\bullet$: Is the state processing function, where the last state ($\bullet r$), (*respectively*, the initial state ($r\bullet$)) denotes the last (*respectively*, initial) state of ($\bullet r$), (*respectively*, $r\bullet$) before (*respectively*, after) the application of r.

The reconfiguration is performed according to the desired probability and the system's resources at the desired instant, i.e., if the user aims to run the most probabilistic reconfiguration while there are no sufficient resources in its reserves, then it has to degrade the mode to the next probabilistic scenario. A state machine specified by an R-TNCES, which is called *Structure_changer*, is introduced to guide the control module following the reconfiguration process. In this state machine, each place corresponds to a specific TNCES of the GR-TNCES model. This place can be introduced as a macro-step which is composed of a set of micro-steps as shown in Figure 10.

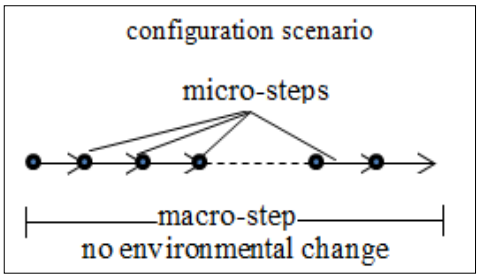


Figure 10 Macro-step, micro-step

Thus, each transition corresponds to a reconfiguration function. A place *sp* gets a token, implying that the TNCES (reconfiguration) to which *sp* corresponds is selected. If a transition *st* ($\forall\ st \in sp\bullet$) is fired, then it removes the token from *sp* to a place *sp'* with $sp' \in st\bullet$ and the TNCES to which *sp'* corresponds is selected. The *Structure_changer* is formalized as follows:

$$Structure\_changer = (P, T, F, Q, E', M')$$

where $\forall\ t \in T$, $|\bullet t| = |t\bullet| = 1$, and only one TNCES is performed at any time. Each place of this structure contains all information about the corresponding TNCES e.g., its energy and memory requirements (number of states in this TNCES). Each state consumes one token from the energy and memory reserve. Thus, before enabling the probabilistic reconfiguration, the availability of energy and memory reserves has to be checked. Only the memory tokens are added back to the model's memory reserve at the end of the adaptation process. The energy reserve will be removed from the battery. Then, the battery will be recharged periodically.

3.2.3 Dynamics of GR-TNCES

The dynamic of a GR-TNCES describes the control operation. To move a token from one state to a next one, the structure modification instruction *S* guides the GR-TNCES from $TN(\bullet r)$ to $TN(r\bullet)$, including the condition/event signals among them. The state processing function *X* maps the last state of $\bullet r$ before the application of *r* to a feasible initial state of $r\bullet$. The dynamics of a GR-TNCES is represented by referring to self-modification nets and net rewriting systems. Figure 11 shows an example of a GR-TNCES model of four R-TNCES. *Mem* and *Eng* are the memory and the energy reserve of the controller and the parameter $Q \in [0, 1]$ is the corresponding probability for each R-TNCES branch that represents the chance to run such a scenario. Let *ß* be a TNCES and Cost TNCES be the needed resources by this TNCES. The states of a GR-TNCES are defined as follows; A state of *G* is a pair (*TN* (*ß*), *State* (*ß*)), where *TN* (*ß*) denotes the net structure of *G* and *State* (*ß*) denotes a state of *G*. The evolution of a GR-TNCES depends on what events, energy and memory constraints take place. A reconfiguration function $r = (Cond, Q, E_0', M_0', S, X)$ is enabled at state (*TN* (*ß*), *State* (*ß*)) only if:

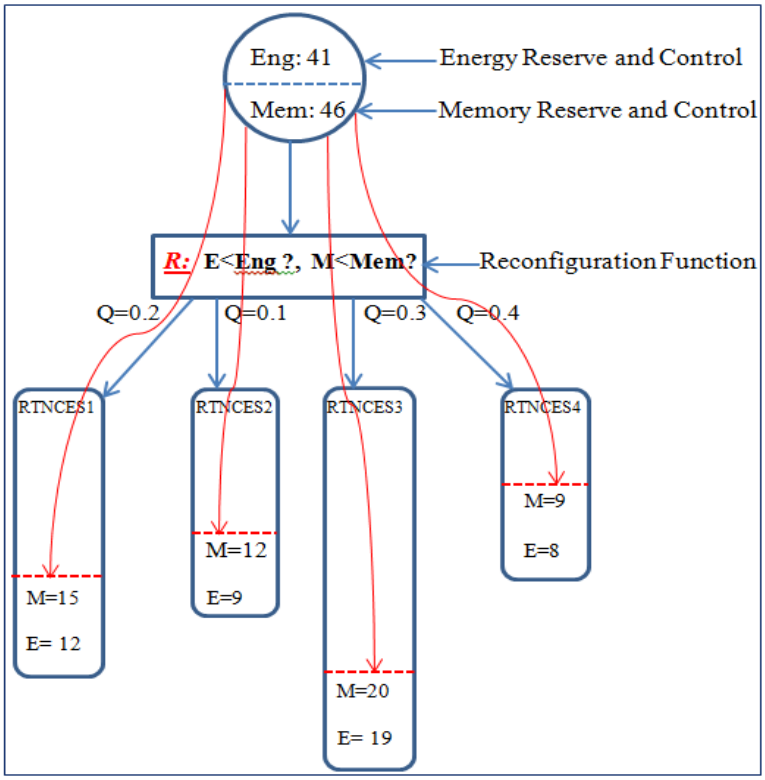


Figure 11 Example of a GR-TNCES architecture

(i). TN ($ß$) = TN ($•r$), i.e., TN ($ß$) is equal to the net structure of $•r$ and the firing time constraints are valid,
(ii). $Cond$ = true: The precondition is fulfilled,
(iii). The energy reserves E' are enough: i.e., $E' > Cost$ TNCES (E_0').
(iv). The memory reserves M' are enough: i.e., $M' > Cost$ TNCES (M_0').

The reconfiguration function is a tuple composed of the required energy and memory resources compared with the current resources storage as well as the events and conditions signals. For example, to select the most probabilistic reconfiguration scenario R_{Max}, the controller chooses the maximal probabilistic transition to be fired in the next step. Let (i) '$\cap e \in EN\ e$' and '$\cap c \in CN\ c$' be respectively the set of all possible *Event-In* and *Condition-In* of the desired transition, (ii) E' and M' are respectively the energy and memory reserves, (iii) 'Cost TNCES$_{Max}$ (E_0')' and 'Cost TNCES$_{Max}$ (M_0')' are respectively the energy and memory required by the most probabilistic reconfigurable scenarios. The reconfiguration is applied by respecting this formula:

$$R_{Max} \equiv (E' > \text{Cost TNCES}_{Max}(E_0')) \wedge (M' > \text{Cost TNCES}_{Max}(M_0')) \wedge \cap e \in EN\ e \wedge \cap c \in CN\ c$$

Indeed, the highest probabilistic scenario has to guarantee that: (i) the needed resource related to the energy and memory resources are available, and (ii) the events and conditions should also occur at the firing time.

3.3 Specification Approach

The languages in which adaptive probabilistic systems are specified should be clear and intuitive, and thus accessible to generation, inspection and modification by humans. We introduce a new specification approach for adaptive probabilistic discrete event systems running under resources constraints.

3.3.1 Motivation

Reconfiguration is often a major undertaking for systems because the post-reconfigurable mode can violate memory usage, the required energy and the concerned real-time constraints. The languages in which probabilistic reconfigurable systems are specified should be clear and intuitive, and thus accessible to generation, inspection and modification, as well as precise and conscientious to ensure the maintenance, analysis and simulation by computers. We introduce a new specification approach for adaptive probabilistic discrete event systems running under resources constraints. The semantics of the formalism GR-TNCES are presented to optimize the specification approach and applied to specify the requirements of an automotive transport system to prove its relevance.

3.3.2 System Specification

To analyse GR-TNCES using state-exploration techniques, we focus separately on the behavior and the control module. We consider the control module as a transition system (C, Rec, In) where C is a set of macro-steps, $Rec \subseteq C \times C$ a transition relation or reconfiguration function. It maps the reconfiguration scenario to the respected constrains (energy, memory, probability). In represents the initial standard configuration, i.e., the start point is a static state. The reconfiguration function is a tuple of the current configuration (macro-step), the corresponding events and conditions, the desired probability, and the needed energy and memory resources compared to the current storage. To execute the highest probabilistic reconfiguration scenario, the controller has to choose the maximum probabilistic transition for the next step respecting this formula:

$$Rec_{Max} \equiv (E' > \text{Cost TNCES}_{Max}(E_0)) \wedge (M' > \text{Cost TNCES}_{Max}(M_0)) \wedge \cap e \in EN\ e \wedge \cap c \in CN\ c \quad (1)$$

which describes how the macro-steps are selected [Khlifi 2018b], i.e., the highest probabilistic scenario has to guarantee the resource constraints [Andrade 2009] related to energy and memory reserves. Moreover, the events and conditions should also occur, otherwise they are considered to be true. For the low probabilistic reconfiguration, the transition relation will be introduced as follows:

$$Rec_{Min} \equiv (E' > \text{Cost TNCES}_{Min}(E_0)) \wedge (M' > \text{Cost TNCES}_{Min}(M_0)) \wedge \cap e \in EN\ e \wedge \cap c \in CN\ c \quad (2)$$

which describes how the macro-steps are selected, i.e., the lowest probabilistic scenario has also to satisfy the resource constraints related to the energy and memory reserves. The events and conditions should also occur, i.e., if there is no event, then the input is considered to be true. Once the macro-step is selected, the system executes the micro-steps of the selected configuration. The behavior module is considered as a transition system (P, R, I) where P is a set of global states, $R \subseteq P \times P$ a transition relation. It is a labeling function that maps each transition to the holding properties in the corresponding transition, and $I \subseteq P$ a set of initial states. A transition in R is a tuple of the current local state (system source state), the events and conditions occurring, the probabilistic value of the environment inputs and the time period in which the transition could be fired. A path is a sequence of states that belongs to P, i.e., a state is reachable only if it appears on such trace path execution. We symbolically encode the global state space P of a GR-TNCES system using a set of variables Y as follows: For each system state m, we consider a state variable from the local states of m. The set of initial states I is represented as:

$$I \equiv \cap_{m \in P}\, m \equiv m_0 \wedge \cap e \in Ei\, \neg e \wedge \cap c \in CNi\, \neg c \wedge (T_0=\{1\}) \wedge (D_0=\{0\}) \quad (3)$$

where m_0 is the initial local state, Ei and CNi are respectively the set of internal events and guarding condition. Initially, the system is in its initial local state, all internal events and guarding conditions do not occur, the state is marked and the clock position is null. The most important thing is the encoding of the nondeterministic transition relation R. We focus on the encoding of the micro-step transition, i.e., for each state variable $var \in Y$, we present a variable var' that has the same range as var and intuitively represents its next-state value. Let Y_0 be the set of all these primed variables. We define an expression over $Y \cup Y_0$ to specify R, then for each local transition t, let $src(t)$, $dst(t)$, $evt(t)$, $cond(t)$, $time(t)$, $mode(t)$, and $prob(t)$, be respectively the source local state, destination local state, trigger event, guarding condition, and the firing time interval, the firing mode{AND, OR}, and the firing probability. The expression $evt(t)$ and $cond(t)$ could be true if the transition t does not have a guarding condition and event inputs. Let $curr(t)$ be the current local state of the system and $enb_{prob}(t)$ to be represented as:

$$enb_{prob}(t) \equiv curr(t) \wedge evt(t) \wedge cond(t) \wedge time(t) \quad (5)$$

It is enabled once the trigger events and guarding conditions occurs simultaneously at the desired running time if the firing mode is 'AND'. We could deal with other firing mode as described here:

$$enb_{prob}(t) \equiv curr(t) \wedge time(t) \wedge (evt(t) \vee cond(t)) \quad (6)$$

It presents how the transition could be enabled if the firing mode is 'OR', i.e., it is considered to be true if one trigger event or guarding condition occurs at the required running time period of the selected transition [Khlifi 2018b]. Once the system executes a configuration

scenario, we aim to describe how the system deals with the micro-steps. For each state m of the system, $micro_m$ describes the progress of the system at run-time process:

$$micro_m \equiv (\cap_{t/curr(t)=m} (enb_{prob}(t) \rightarrow curr'(t)=dst(t))) \vee (\cap_{t/curr(t)=m}$$

$$(\neg enb(t) \rightarrow curr'(t)=curr(t))) \quad (7)$$

Indeed, the first conjunct guides the system states from the enabled transition to the destination state, while the second conjunct blocks the system on the same position if none of the transitions are enabled [Khlifi 2017a]. The fired transition can generate various events. Moreover, the generation of events $evt(t)$ and conditions $cond(t)$ are introduced respectively as follows:

$$micro_e \equiv (\cup_{t/e \in Evt(t)} enb_{prob}(t)) \leftrightarrow e' \quad (8)$$

The event is delivered by the union of the enabled transition that can send events to activate different states of the system. Similarly, the micro-step generates guarding condition and it is represented as:

$$micro_c \equiv (\cup_{m/c \in Cnd(t)} micro_m(t)) \leftrightarrow c' \quad (9)$$

It is generated by a union of states to control the execution of various tasks of the system. Then, we introduce *Macro* to encode the macro-step which is a conjunction of micro states, micro events and micro conditions as follows:

$$Macro \equiv \cap_{e \in CN} micro_c \wedge \cap c \in EN\ micro_c \wedge \cap m \in P\ micro_m \quad (10)$$

The presented specification approach makes possible to deal simultaneously with unpredictable reconfiguration scenario, time constraints, and limited energy and memory resources. It is useful to specify the system requirements in an optimized way.

3.4 Case Study: Skid Conveyor

We describe in this part the case study of our thesis with the aim to save its energy consumption. The specification of the functionalities and the different operations mode is presented and discussed.

3.4.1 Description

The transport system should consist of three conveyor parts [Khlifi 2016]. Currently, there is an old system where all the motors are switched together and manually from one mode to another. We aim to introduce new functional modes which offer the user to localize the chassis on every part, i.e., in each conveyor part, the chassis should wait for 7 seconds to establish some other tasks by various robots.

Figure 12 Skid conveyor system at ZeMA.

To minimize the consumed energy of the system during the movement of the chassis, each unused actor should be switched off or to a standby mode. For example, once the chassis is in the third conveyor part, the motor of the first one should be switched off. The activation/deactivation of the motors is controlled based on the car position. Moreover, the worker should control all the possible positions showed in Figure 12 with a control panel. Using this approach, it is possible to choose one operation mode for the system. The requirements for these operation modes are explained in the following part. The system is reconfigurable and we consider three possible reconfigurations:

- **Automatic mode**: The worker activates and stops this mode using the panel. The speed of the skid should be as well controlled. Then, all other sensors and actors operate automatically, i.e., (i) the chassis position has to be clear, (ii) then, the chassis moves from one workstation to another without user interaction. Since the position of the chassis is logged, all unused actors can be switched *Off*. As soon as the chassis is at the third position it should move backwards to the start position and start again.
- **Manual mode**: In this mode, the worker should manually control the system's functionalities, i.e., to start and stop all the conveyor parts individually and together and controls the speed of the chassis.
- **Pause mode**: In order to save energy, the worker can activate and deactivate this mode with the panel. If the mode is activated, then all sensors and actors are switched *Off* or changed to a *standby* mode according to the duration of the Pause mode.

If the worker uses this option, then he can visualize all the relevant sensor and actor data. For example, whether the motor is *ON* or *Off* and its speed.

3.4.2 Specification

The skid conveyor is supervised by a centralized controller, i.e., it enables to control and switch the system from one configuration to a second one. To simplify the use case specification, we consider that the system is not probabilistic and that the switching mode is chosen by the user to be denoted by $RTN_{skid} = \{B_{skid}, R_{skid}\}$. Let E_{skid} and M_{skid} be respectively the energy and memory reserves of the skid system. We use the proposed specification

approach to specify the system, i.e., each mode is represented by a macro-step. We identify three macro-steps for the different modes:

Rec1= *Macro1*: Automatic mode, *Rec2*= *Macro2*: Manual mode, *Rec3*= *Macro3*: Pause mode. This is a reconfigurable system, i.e., it can switch the behavior from one mode to another mode. R_{skid} represents the control module of the system as:

$$R_{skid} = Rec1 \cup Rec2 \cup Rec3$$
$$= \{r_{Rec1,\ Rec2},\ r_{Rec1,\ Rec3},\ r_{Rec2,\ Rec1},\ r_{Rec2,\ Rec3},\ r_{Rec3,\ Rec1},\ r_{Rec3,\ Rec2}\}$$

The reconfiguration: "$r_{Rec1,\ Rec3}$" implies that "•*r*"= "*Rec1*" and "*r*•"= "*Rec3*". It enables to switch mode from the current "*Rec1*" to the next configuration"*Rec3*". Figure 13 helps to explain the structure of the system and the possible reconfiguration processes, i.e., it shows the possible switching mode between all the macro-steps. The *Idle* position refers to initial state where the system clock is null and the initial marking is true. It could be specified as follow:

$$I \equiv \cap_{m \in P}\ m$$
$$\equiv Idle \wedge \cap e \in Ei\ \neg e \wedge \cap c \in CNi\ \neg c \wedge (T_0=\{1\}) \wedge (D_0=\{0\})$$

The initial states are the set of states that does not have any input events and conditions signals, i.e., it does not need any stimulator to be activated. Then, according to the choice of the user, the system reacts to the received command. Let *EN1*, *EN2*, *EN3* be respectively the external events to activate *Rec1*, *Rec2*, and *Rec3*. *Marco1* is introduced as the conjunction of the energy constraint presented as "(E_{skid} > Cost '*Macro1*' (E_0))", the memory constraint and the trigger event that will initiate the desired configuration:

$$Macro1 \equiv (E_{skid} > \text{Cost '}Macro1\text{'}\ (E_0)) \wedge (M_{skid} > \text{Cost '}Macro1\text{'}\ (M_0)) \wedge EN1.$$

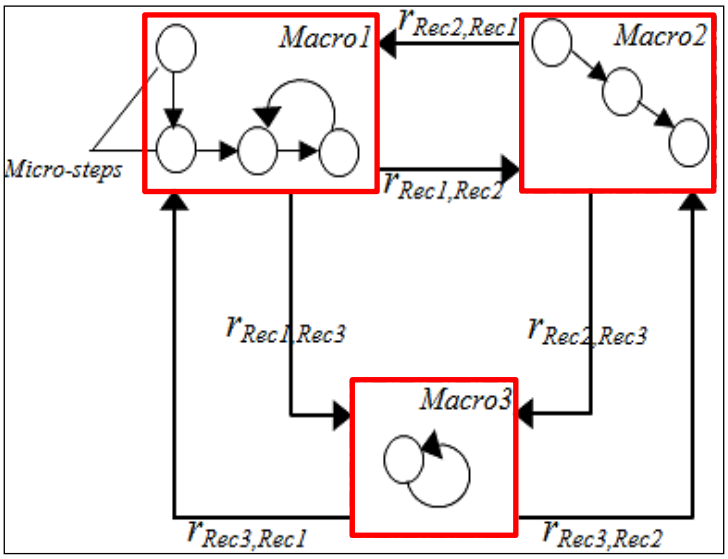


Figure 13 GR-TNCES System model

The system keeps the same running mode "*Rec1*" till it receives a trigger event from the user to change the operational mode. The second reconfiguration "*Rec2"* is also introduced as follow:

$$Macro2 \equiv (E_{skid} > \text{Cost } `Macro2' (E_0)) \wedge (M_{skid} > \text{Cost } `Macro2' (M_0)) \wedge EN2.$$

This macro is executed once the energy and memory resources are available, i.e., the constraints are respected and the corresponding event *EN2* is received. The system could move for the third configuration once its conjunctions are validated. This configuration is introduced as followed:

$$Macro3 \equiv (E_{skid} > \text{Cost } `Macro3' (E_0)) \wedge (M_{skid} > \text{Cost } `Macro2' (M_0)) \wedge EN3.$$

Once the configuration is selected, the system executes the different internal tasks of the concerned macro-step. The behavior module of RTN_{skid} is formally presented as follows:

B_{skid} = (P, T, F, QW, CN, EN, DC, V, Z_0) where the network structure of the system is listed as: TN_{Maco1}, TN_{Maco2}, $TN_{Maco3} \in TN_{skid}$. We have:

- $P = P1 \cup P2 \cup P3$, where *P1*, *P2* and *P3* are respectively the set of states of *Rec1*, *Rec2*, and *Rec3*.
- $T = T1 \cup T2 \cup T3$, where *T1*, *T2* and *T3* are respectively the set of transitions of *Rec1*, *Rec2*, and *Rec3*.
- $F = F1 \cup F2 \cup F3$, where *F1*, *F2* and *F3* are respectively the set of flow arcs of *Rec1*, *Rec2*, and *Rec3*.
- $W = W1 \cup W2 \cup W3$, where *W1*, *W2* and *W3* maps a weight to the flow arcs respectively in *Rec1*, *Rec2*, and *Rec3*.
- $CN = CN1 \cup CN2 \cup CN3$, where *CN1*, *CN2* and *CN3* are respectively the set of guarding conditions to initiate *Rec1*, *Rec2*, and *Rec3*.
- $EN = EN1 \cup EN2 \cup EN3$, where *EN1*, *EN2* and *EN3* are respectively the set of events to initiate *Rec1*, *Rec2*, and *Rec3*
- $DC = DC1 \cup DC2 \cup DC3$, where *DC1*, *DC2* and *DC3* are respectively subset of time constraints on output arcs of *Rec1*, *Rec2*, and *Rec3*
- $V(t) = V1(t) \cup V2(t) \cup V3(t)$ where $V1(t)$, $V2(t)$, $V3(t)$ maps respectively an event-processing mode (AND or OR) for every transition, on *Rec1*, *Rec2*, and *Rec3*,
- $\forall\, p \in P1 \cap P2 \cap P3$, $Z_0(p)$ is the initial clock position for the system.

We focus on the behavioral module for the specification of the system requirements, and then we introduce the micro-steps of the first macro-step. The authors try to identify some properties of the system, e.g., 'it should be possible to localize the chassis on every part of the conveyor'. Let $curr(t)$ be the system state that describes the position of chassis and $Pos1_{enb}$ be the micro-step that represents the car position in the first part of the skid. In case that the chassis should be in the first position at a prefixed time period $[a_1,b_1]$, the trigger events $E_{1.1}$

and $E_{1.2}$ should be detected at the same period. We can formally introduce this micro-state as:

$$Pos1_{enb} \equiv curr(t) \wedge E_{1.1} \wedge E_{1.2} \wedge time[a_1,b_1]$$

which evaluates the transition, i.e., it could be enabled only if all the listed conjunctions are simultaneously true. For the second position of the skid, it is formalized based on the same rules as follow:

$$Pos2_{enb} \equiv curr(t) \wedge E_{2.1} \wedge E_{2.2} \wedge time[a_2,b_2]$$

Where (i) $E_{2.1}$ and $E_{2.2}$ are the corresponding events to detect the considered position, and (ii) $[a_2,b_2]$ is the time period for this scenario. The proposed system aims to save the energy consumption of the transport system, i.e., the corresponding motor for each conveyor part should be *Off* if there is no car at that position. Let $m1_{act}(t)$ be the active state of the first motor and $m2_{act}(t)$ the active state of the second motor. Here we define the rules for the activation of motor 2 $m2_{act}(t)$ as:

$$m2_{act}(t) \equiv m1_{act}(t) \wedge curr(t) \wedge E_{2.1} \wedge E_{1.2}$$

which define the active state of the second motor as a conjunction of the active state of the first motor, the presence of the chassis in the conveyor, the occurrence of the $E_{1.2}$: (chassis at the end of conveyor 1) and $E_{2.1}$: (chassis at the beginning of conveyor 2). To optimize the energy consumption, the system has to switch *ON*/*Off* the motors according to the position of the chassis. Once the second motor is turned *ON*, the first should be *Off*. We formalize the deactivation of the first motor as:

$$\neg m1_{act}(t) \equiv m2_{act}(t) \wedge \neg curr(t) \wedge E_{2.1} \wedge \neg E_{1.2}$$

which represents that stopping the first motor is initiated by the activation of the second motor. The absence of the chassis in that position is confirmed by the non-occurrence of the event $E_{1.2}$ and the occurrence of the event $E_{2.1}$. The enabled transitions can generate many events for the synchronization of the system parts. Then, the micro event $E_{2.1}$ is delivered after the movement of the chassis from the first skid to second one ($curr(t) \rightarrow curr'_{enb}(t)$). The formalization is as follow:

$$micro_e \equiv (\cup_{t/e \in EN1(t)} curr'_{enb}(t)) \leftrightarrow E_{2.1}$$

The authors present the specification of the proposed case study requirements using the presented approach. We move to the next step which is the modeling, simulation and the implementation of the system.

3.5 Discussion

We proposed a new expressive and optimized modelling approach compared to UML [Bondavalli 1999], [Shousha 2012] and [Bernardi 2007] Petri net formalisms, and statecharts used in symbolic model checking. The proposed approach has ability to cope with reconfigurable systems and timed constraints which was not possible in statecharts and other approaches. It can also express various constraints related to timed systems and real time process and enables to describe systems that could change their behavior. It is also possible to select different firing modes for the transition states: i.e., we can opt for AND/OR mode according to the system requirements. ("AND" if all the input transition are required and "OR" is used if one of them could activate the transition). In addition, it is possible to check the availability of resources before starting such a reconfiguration process, i.e., to guarantee the non-resources violation once the system executes its tasks. Unpredictable behaviors are also considered here since the specification approach is able to describe the probabilistic behaviour. Tab.1 represents a detailed comparison between different modelling formalisms.

Table 1 Comparison Table

Formalisms	**Advantages**	**Drawbacks**
UML-RT [Selic 1998]	✓ Complete modeling language allowing to model complex and event-driven RT systems	- No support for time constraints - Limited modeling capabilities for performance and architecture.
Embedded UML [Gogolla 2001]	✓ Specification, design and verification of embedded RT systems	- Concurrency, communication and implementation issues.
Petri Nets [Genter 2007]	✓ Graphical notation for choice, iteration and concurrent execution. ✓ Well-developed mathematical theory for process analysis.	- Shortcoming when handling time and reconfiguration.

NCES [Rausch 1995]	✓ Condition/event signals and the possibility of firing several transitions simultaneously.	- No time intervals for output flow arcs.
TNCES [Hanisch 1997]	✓ Possibility to assign time intervals to each output flow arcs.	- Only relevant for static systems.
R-TNCES [Zhang 2013]	✓ Introduces reconfiguration in TNCES.	- Lack of a tool for modeling and verification.
GR-TNCES [Khlifi 2015]	✓ Model probabilistic, energy and memory features. ✓ ZIZO tool enables the modeling and simulation using this formalism.	- Cannot deal with complex probabilistic theories.
Statecharts [Chan 2001]	✓ Model hierarchical reactive timed systems.	- Difficult to provide formal semantics and probabilistic behaviors. - Cannot model energy and memory resources.

3.6 Summary

We introduced GR-TNCES which is a new extended formalism for the modeling of adaptive probabilistic systems. It enables the specification of probabilistic reconfigurations, energy and memory resources, real-time constraints and distributed architectures of such a system. A new specification approach dealing with unpredictable flexible control systems running under memory and energy resources constraints is also presented; It is an expressive method that could specify limited memory and energy reserves, probabilistic behaviors, and reconfigurable processes which was not discussed in the previous work. The proposed approach is based on GR-TNCES formalism which enables us to express the probabilistic reconfiguration scenario of such a system. An automotive transport system is the considered case study to concretize the contribution.

Chapter 4
Simulation and Formal Verification

Contents

4.1 Introduction……...43
4.2 Simulation……..43
4.2.1 Probabilistic Simulation……44
4.2.2 Energy Simulation…….....45
4.2.3 Memory Simulation…….....47
4.3 Formal Verification……,,,……...48
4.3.1 Formal verification: Export to PRISM……48
4.3.2 New Verification Approach…….....53
4.3.3 Incomplete Labeled Transition System…….....56
4.3.4 XCTL Model Checker…….....57
4.3.5 Marking Algorithms…….....59
4.3.6 Degraded Verification Mode……...59
4.3.7 Discussion……...61
4.4 Summary…….....61

SIMULATION AND FORMAL VERIFICATION

4.1 Introduction

Simulation is the creation of a model that can be manipulated logically to decide how the physical model works. Simulation and formal verification are complementary techniques; both are required for the development of complex and safety-critical systems. We present here our contributions related to simulation of probabilistic processes running under resources constraints and formal verification of reconfigurable systems. The simulation algorithm is implemented and published in [Khlifi 2018a] and the verification approach is published in [Khlifi 2017]. This chapter is detailed in the recently cited papers for more details.

4.2 Simulation

Simulation is a primary certification step realized for the Petri net models. We simulate the model through the selection of a token game animation. Once the simulation is finished, the token reaches the final state of the model. Petri nets simulators offer features of standard simulation functionality; it can also generate a report showing a textual description of the simulated process. If a deadlock is detected [Li 2012], then the token cannot reach the end state and it is stuck at this deadlock state, i.e., it cannot progress to the next states. Thus, it is helpful to detect the failure at an earlier stage and revise the specified model at this state. We refer to the specification to change the model again as shown in Figure 14 and reiterate the certification process. Simulation is applied prior to model checking to get a first functional model of the system´s behavior and to find out the eventual design errors early. If the latter exists, formal verification techniques may be time-consuming once it is performed on a very large state-space model.

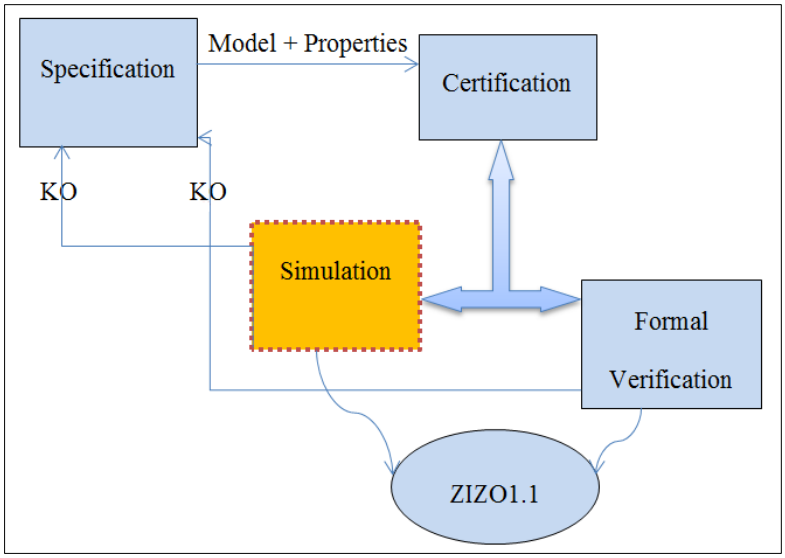


Figure 14 Simulation step

In this chapter we present a new simulation algorithm dealing with simple Petri net models and with more complex systems [Bruyninckx 2013] such as probabilistic reconfigurable systems running under limited memory and energy resources. The presented simulation algorithm depends on various inputs, not just on the presence of the token at the precedent places. Here we deal with condition input, event input, a valid time interval and a firing mode. In addition to the pre-mentioned artifacts, the approach distinguishes between three probabilistic simulation scenarios.

4.2.1 Probabilistic Simulation

Simulation-based approaches ensure if a finite number of user-defined system trajectories meet the desired project goal. Probabilistic systems have various sequences of tasks that could be executed at each state. Thus, we assigned the probabilities to the branches to indicate the chance of choosing such a path. We are interested in three simulation strategies according to the probability level: (i) High: Visiting successively just the branches with the highest probability level, see Figure 15. The model shows that only branches with highest probabilities are executed, i.e., the token will visit just these branches with the highest chance of execution. (ii) Average: Visiting the branches with the average probability level and (iii) Low: Visiting the paths with the lowest probability level, see Figure 16 which presents an example of the visited model branches. The simulation process consumes memory and energy resources depending on the chosen simulation. The aim is to control the resources' availability before starting and even at run-time process to guarantee the non-violation, i.e., make sure that the system will not face any situation where there is no energy and memory resources.

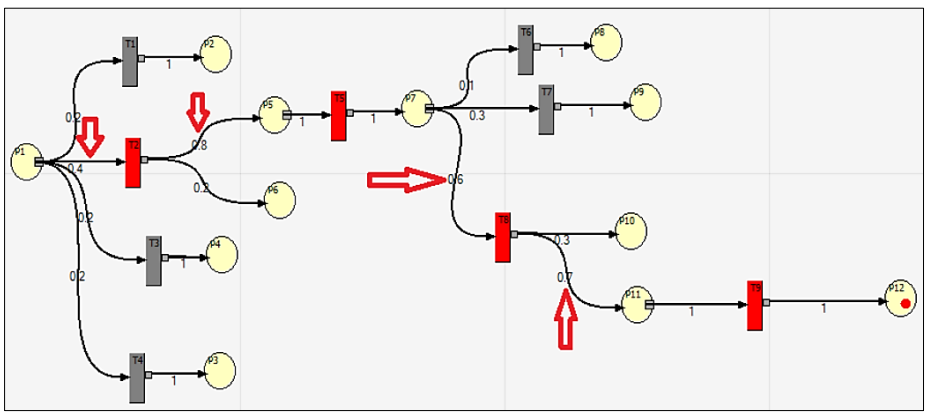


Figure 15 High Probabilistic Simulation

The simulation algorithm is based on these rules:

(i). The simulation cannot start if the memory reserve is lower than the chosen path's consumption and a deny message is displayed.

(ii). If the energy reserve reaches the minimal capacity (chosen to be three tokens in the proposed case study), then the tool displays a warning message at run-time asking the designer to recharge the battery.

(iii). The designer must set a recharge period: periodically, the energy reserve will be recharged to the full value; the memory resources will be free at the end of each reconfiguration scenario since we cannot charge the memory.

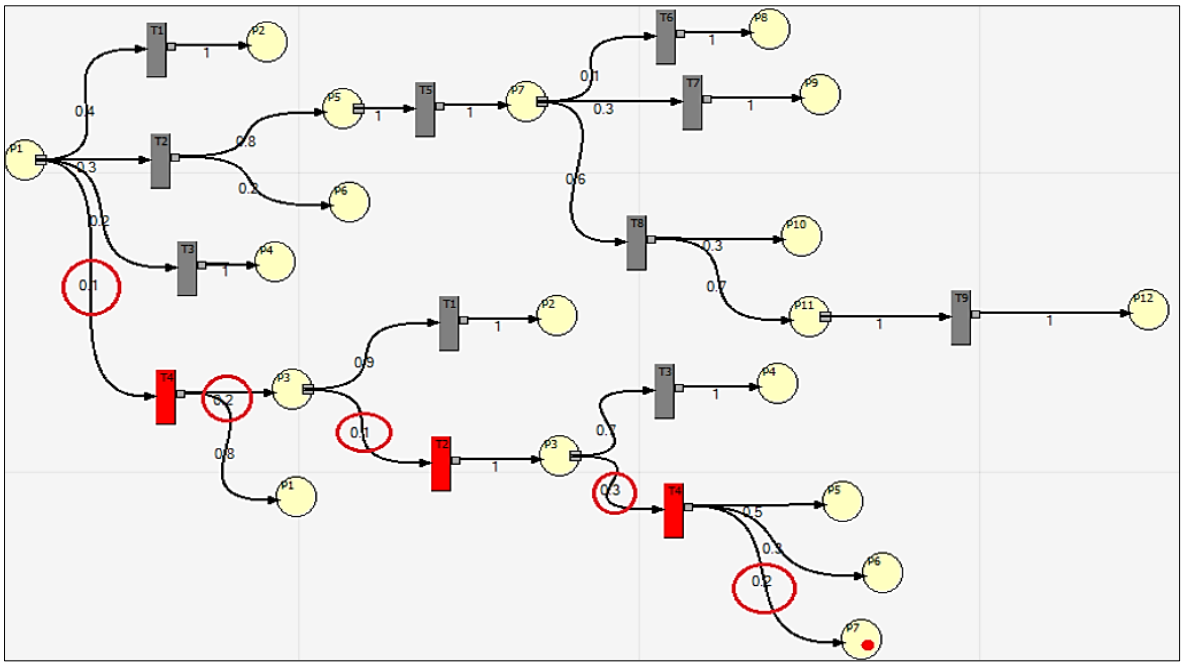


Figure 16 Low Probability Simulation

4.2.2 Energy Simulation

Many systems and protocols run using components with limited energy resources, e.g., limited battery in wireless sensor network nodes [Shareef 2010]. Thus, the system could violate its resources at run-time or an adaptation process. In our thesis, the energy unit is not the Watt, but it is an abstract unit. We suppose that each fired transition corresponds to the execution of such a task and consumes one unit or one token. The presented simulation process consumes an amount of energy resources that is estimated depending on the chosen simulation type and the size of the played process. The aim of our work is to develop a control strategy to supervise the resources consumption. This supervision and simulation process could detect and predict if there is a scenario in which the system crashes due to the lack of energy. Moreover, the designer could manage the battery's capacity of each system. Then, the resources will be supervised during the chosen simulation scenario. Figure 17 presents the implemented algorithm to choose the average probability branches. Once the branch is chosen, the proposed energy simulator has to prove various rules to ensure the energy control.

- **Rule-1**: the simulation cannot start if the current energy capacity is lower than the number of places in the chosen path. In fact, once the simulation is chosen, the controller compares the current capacity to the size of the selected path. More precisely, if the designer chooses the high probabilistic simulation, then only the size of this branch will be evaluated and compared to the available resources at that moment. If the energy capacity does not fulfil the required resources, the simulation will not be started and an error message is displayed.
- **Rule-2**: if the energy reserve reaches three units (rest of energy=3), a warning message will be sent at runtime asking the designer to recharge the battery. Then, the simulation will be continued again just once the battery is recharged, otherwise, it is stopped, and the Endpoint could not be reached.
- **Rule-3**: The designer must set a recharge period measured in seconds. Then, periodically, the energy reserves will be initialized to this initial value.

```
private NodeGraphLink GetLinkAverageProba(List<NodeGraphLink> liste)
{
    float[] tab = new float[10];
    NodeGraphLink resultat = new NodeGraphLink();
    float tmp = 0;
    float proba_choosed = 0;
    int size = 0;
    int midle = 0;
    for (int i = 0; i < liste.Count(); i++)
    { if (liste.ElementAt(i) != null)
        {tab[i] = liste.ElementAt(i).x;}
    }
    for (int k = 0; k < tab.Count(); k++)
    { if (tab[k] != 0)
        { size++; }
    }
    for (int i = 0; i < size - 1; i++)
    { if (tab[i] < tab[i + 1])
        {   tmp = tab[i];
            tab[i] = tab[i + 1];
            tab[i + 1] = tmp;
        }}
    midle = size / 2;
    proba_choosed = tab[midle];
    for (int i = 0; i < liste.Count(); i++)
    {
        if (liste.ElementAt(i).x == proba_choosed)
        { resultat = liste.ElementAt(i); }
    }
    return resultat;
}
```

Figure 17 Algorithm of Average Probability Simulation

4.2.3 Memory Simulation

Similar to the energy control strategy, we developed a memory control approach to guarantee and supervise the availability of the memory resources during simulation time.

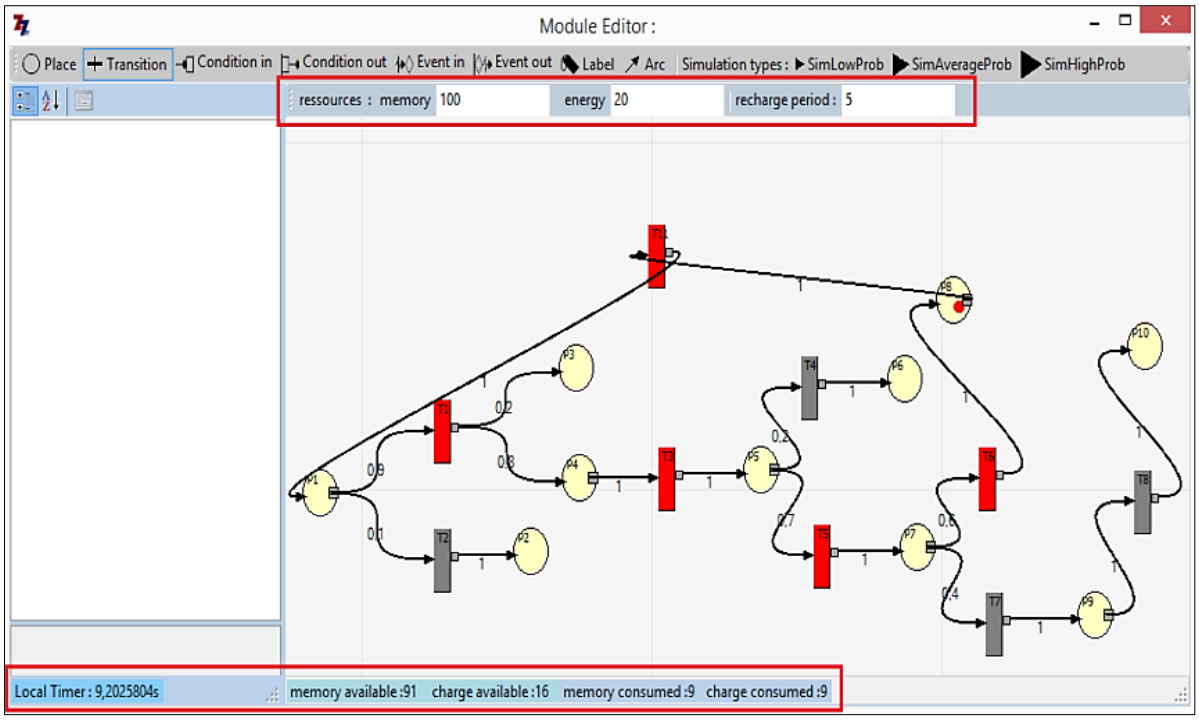


Figure 18 Energy and Memory simulation.

These memory reserves are also represented by an abstract unit or a token to be consumed at each firing transition. The precondition of starting such a simulation ensures that the initial memory reserves are the double of the size of the selected model. In addition, each fired transition consumes a single token of the memory resources. There will be no recharge period similar to the energy reserves because the memory of the system is a predefined limited capacity that is used by its resources and it's free only once the running processes are finished. Figure 18 shows how we can initialise the memory and energy resources and the recharge period of such a system. The bottom of the screenshot presents the available and consumed resources during the chosen simulation process. We implemented also the possibility to extract the curves of the resource's consumption during the simulation period, i.e., it is a curve that presents the total consumed number of tokens during the simulation time.

4.3 Formal Verification

Formal verification enables developers to deal with complexity using well-proven tools of logic and mathematics, providing strong assurance on compliance with requirements. We provide here two main contributions: the first is a mapping algorithm that enables to transform a GR-TNCES model to a PRISM code for the formal verification. The second is a new formal verification approach dealing with incomplete and reconfigurable systems.

4.3.1 Formal verification: Export to PRISM

Once the model is simulated using the environment ZIZO, we need to go further in the certification process described in Figure 14. The goal is to formally prove that the proposed model is safe, correct, and does not present any time or resources violation. We developed the possibility to export the created model to PRISM model checker for the formal verification. It is an automatic generation of the PRISM code from the graphical model. We do not need to re-describe the system model in PRISM especially for complex system which saves us an important time. This operation is guaranteed through the generation of a "*.pm*" file. The export operation is done using a transformation protocol that matches each element of the GR-TNCES model with a corresponding element from the PRISM Language. To understand this mapping protocol, we need to present the GR-TNCES formalism components and the PRISM language basics. Then, we can describe the mapping protocol.

4.3.1.1 ZIZO Components

ZIZO enables us to model adaptive probabilistic distributed discrete event condition systems. These systems are represented by various distributed modules connected using condition/event signals see Figure 19. Orange link are condition links and red links are event links that connect the modules. This model defines the dependence between modules, i.e., "*module_b*" needs an input event and an input condition from "*module_a*" to execute some tasks. It is similar to "*module_c*" that has a dependency to "*module_b*". Each module has its own structure/architecture that describes its corresponding behavior. Figure 20 describes the content of such module and the various components presented by ZIZO. More precisely, it connects a set of identified components:

- Places: Which are the basic elements of our Petri net model and it could be marked by a token to represent the current state of the system.
- Transitions: Enable moving the system from one state to second one if it is fired. It has two firing modes and a predefined firing time period.
- Condition_Out: Enables to check the existence of a token in the current place, it has a Boolean value.
- Condition_In: Connects the condition_out of the previous module and contains its value. It is an input for the corresponding module.
- Event_Out: It is an output event that transfers an event signal to the next module.
- Event_In: It is an input event that receives an event from the previous module.
- Normal_Link: used to connect places to transitions, it could transfer a token and contains the probability on the arc.
- Condition_Link: used to evaluate the presence of a token in the source place.
- Event_Link: used to stimulate a transition once the parent transition is fired.

- Inhibitir_Link: it is the opposite link of Condition_Link. The link selection offered by ZIZO is shown in Figure 20.

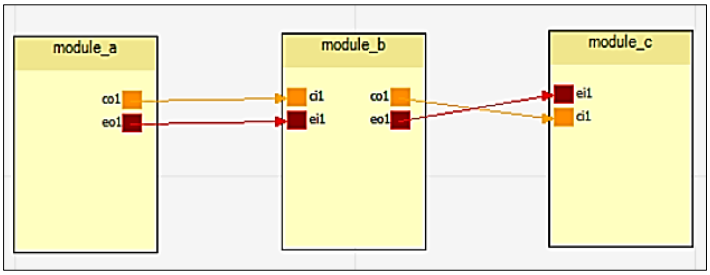


Figure 19 Example of a System Model.

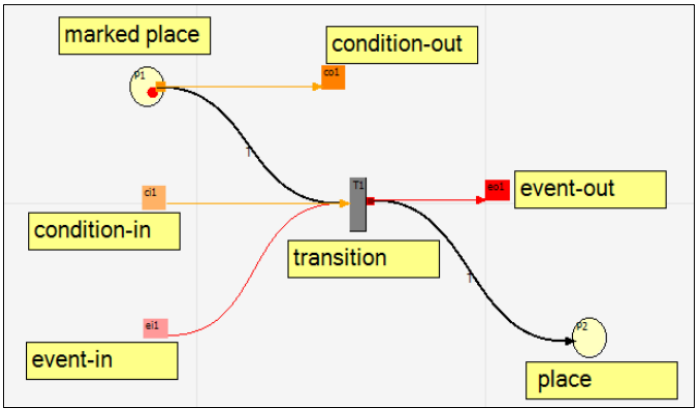


Figure 20 Component of a Module.

4.3.1.2 Program in PRISM

Table 2 presents the main symbols used for the PRISM syntax and semantics. For defining a path, the usual operators known in computational tree logics can be used such as: X (next), U (until), F (eventually), G (always), W (weak until), R (release). Probabilistic Timed Automata "PTA" is on the first hand an extension of Markov Decision Processes "MDP" with clocks and constraints on clocks. On the other hand, it is an extension of timed automata with discrete probabilistic choice [Jeremy 2008]. GR-TNCES is based on a probability on arcs and timed Petri net. It can offer the characteristics of PTA and then, of "MDP" since it is one of its extensions. Thus, to export ZIZO model to PRISM, we can translate the GR-TNCES model into an MDP model (because PTA is modeled by MDP). The first line of the generated file is based on MDP model indicates the model type and the remaining lines describe the PRISM modules that represent the behavior of the system. In each module, there are variables used to specify the state of the system. Each line starts with a state number and indicates the probability to reach other states.

Table 2 Syntax of PRISM symbols.

−	unary minus
*, /	multiplication, division
+, −	addition, subtraction
<, <=, >=, >	relational operators (less than, less or equal than...)
=, ! =	equality operators
!	Negation
&	Conjunction
|	Disjunction
<=>	if-and-only-if
=>	Implication
−>	actualization operator (placed after guard expression)
[ ]	transition label (label can be written between the brackets)
?	condition evaluation: condition? x : y translates to "if condition is true then x else y"
P	probabilistic operator
S	steady-state operator
R	reward operator
A	for-all operator
E	there-exists operator

Table 3 ZIZO-PRISM Correspondence

ZIZO Model	***PRISM Model***
Place	System State
Transition	Arrow (→)
Condition_Out	System State
Condition_In	System State
Event_Out	System State
Event_In	System State
Prob. Normal Link	State Probability
Event link	&: Conjunction
Condition link	&: Conjunction
Inhibitor link	None

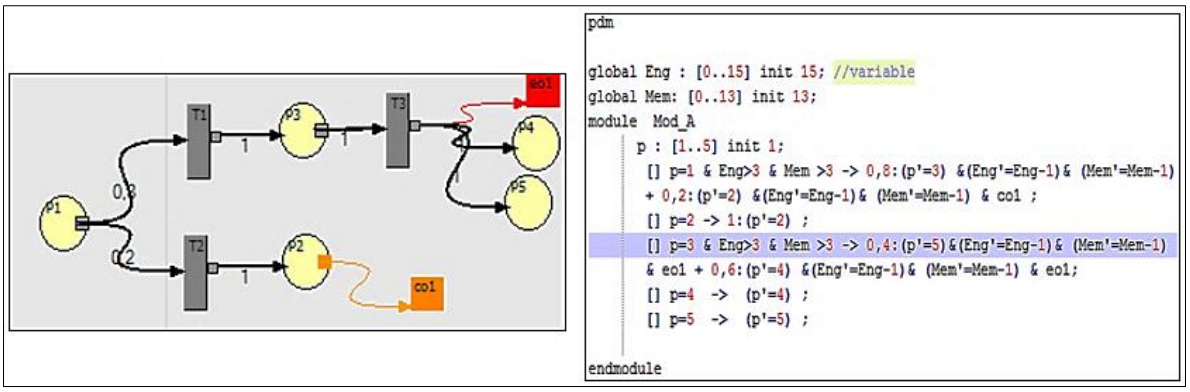


Figure 21 Example of ZIZO-PRISM Transformation

The transformation protocol is described in Table 3, i.e., it presents the correspondence between the components of each model. We use a simple example to clarify the related protocol, i.e., the model's diagram is shown in the lift part of Figure 21. The considered simple example is composed of five places, three transitions, an *Event_Out* and a *Condition_Out*. The right part of the same screenshot represents its corresponding PRISM code. The code starts with the declaration of the memory and energy resources (*Eng* and *Mem*). Here, we present the generated Prism code using our contribution. The module name and its state's number are transferred to the PRISM file.

```
foreach (NodeGraphNode module in modules)
{   int nombre = 0;
    Components.Module module_a_sauvegarder = (module as Components.Module);
    savefilecontent +="module  "+ module_a_sauvegarder.Name+"\r\n";
    NodeGraphPanel graphe_du_module_a_sauvegarder = module_a_sauvegarder.Localpanel;
    List<NodeGraphNode> elements_du_module = graphe_du_module_a_sauvegarder.View.NodeCollection;
    foreach (NodeGraphNode node in elements_du_module)
    {   if (node is Components.Place)
        { nombre++; }}
    savefilecontent += "      p : [0.." + nombre + "] init 0;\r\n";
    foreach (NodeGraphNode element in elements_du_module)
    {
        if (element is Components.Place)
        {
            if (element.Name[1] =='0' ||element.Name[1] =='1'|| element.Name[1] == '2' || element.Name[1] == '3' || element
            { savefilecontent += "       [] p=" + element.Name[1] + " -> "; }
            else if ((element.Name[1] == '0' || element.Name[1] == '1' || element.Name[1] == '2' || element.Name[1] == '3'
            { savefilecontent += "       [] p=" + element.Name[1] + element.Name[2] + " -> "; }

            List<NodeGraphNode> successeurs = nodes_successor_of(element);
            List<NodeGraphLink> liste_lien_normal = new List<NodeGraphLink>();
            foreach (NodeGraphNode succ in successeurs)
            {   if (link_Connector(element, succ).NodeGraphDataType is NormalLink)
                {liste_lien_normal.Add(link_Connector(element, succ));  }
            }
            if (liste_lien_normal.Count() > 0)
            {
                foreach (NodeGraphLink lnk in liste_lien_normal)
                {   float proba_place = 1;
                    proba_place = lnk.x;
                    NodeGraphNode transition = new NodeGraphNode();
                    transition = lnk.GetNextNode();
```

Figure 22 ZIZO-PRISM Transformation Algorithm

The transformation code shows that *P1* reaches *P2* with the probability 0.2 and *P3* with the probability 0.8. *P4* and *P5* are end nodes: thus, they remain at the same state. The management of the memory and energy resources is described during the progress of the simulation. "*Eng*>3" and "*Mem*>3" are the conditions to control the resources before firing each transition. It is possible to simulate the model only if there are more than three token units. "*Eng*'=*Eng*-1" and "*Mem*'=*Mem*-1" describe the variation of the resources at each firing transition. The transformation algorithm is presented in Figure 22. It presents the export process of each GR-TNCES component to PRISM model.

4.3.2 New Verification Approach

Formal verification has been widely used to guarantee that a system specification satisfies a set of properties [Kalita 2002]. We present in this part the current verification methods and our proposed approach dealing with incomplete and reconfigurable systems.

4.3.2.1 Overview

The existing methods to certify reconfigurable systems mainly focus on the specification and verification of adaptation process: These approaches are based on a complete knowledge of the system and the environment behavior at design time, so they are able to reason about the properties of the whole interaction model [Bortolussi 2015]. However, this is not the case in many realistic examples in which the information about the behaviour of some components and the environment are obtained only at runtime. Therefore, run-time verification techniques come into play to monitor and check that the running system does not violate the specification and the properties [Bortolussi 2015]. Although it is less expensive than model checking but it still not complete, and do not guarantee the satisfaction of the properties. Nevertheless, we find some limits in the temporal logic CTL for the optimal verification of adaptive properties.

To avoid any requirement violation, we must guarantee that all the properties will be satisfied in case of applying any reconfiguration scenario [Sharifloo 2013]. This could be guaranteed by formally verifying the new system specification, which is obtained by integrating the specification of the new configuration, against the properties. Intuitively, it is an extra work and overhead because the major part of the specification does not change. Moreover, model checking a large specification at run-time at each reconfiguration is difficult because of the time and resource limitations [Sharifloo 2013]. Thus, once it is possible to refer to the verification results of the invariant part for the future verification, this would significantly save the time and resource usage. This is why verification techniques to be proposed here should verify all behaviors of the reconfigurable systems. We address run-time model checking of reconfigurable systems which are seen as systems with changing or unstable specifications. We focus on components based reconfigurable systems represented by an extension of Labelled Transition System and a model checking approach based on

reconfigurable Computation Tree Logic [Khlifi 2017]. More specifically, this approach allows the designer to verify the system at design time, even if some components are not fixed (unstable, can be replaced). The proposed model checking approach verifies if the requirements hold and produce a set of constraints for the unspecified components.

4.3.2.2 Motivation and Example

Rail transport is a means of conveyance of passengers and goods on wheeled vehicles running on rails. It is referred to a train transport system which is a complex and critical system because it deals with millions of human every day. In addition, it is faced to different challenges: safety from collisions and derailments and providing the maximum line capacity to support trains on the same line within the safety constraints [The Metro 2017]. These systems are considered to be reconfigurable distributed systems since the railway structure is not static: it is usually the subject of various extensions on different lines. It is also faced to numerous accident, structures breaking and natural disasters. Moreover, the number of trains is always changeable; it is possible to add extra trains to cover the increased need and to maintain the quality of service. Similarly, rapidly increasing the capacity is the biggest challenge facing all mass transit operators today. As major cities expand, so this leads to a demand for high capacity and efficient railway network. In addition, the speed of trains is not constant for almost of the lines. Each change can be considered as an adaptation process that affects the characteristics of the system. As a real case study, the Paris Metro is a safety critical reconfigurable system [The Metro 2017]. It is a large railway network with 14 main lines that cover 303 stations in the Paris area. It is mostly underground and it has 205 km of tracks. This system carried 1.5 billion passengers in 2014 [The Metro 2017].

The Metro system is an example of component-based systems that their safety properties depend on the dynamic components which are variable and change at run-time. Such systems require a continuous verification process to certify the safety of the system at any new adaptation process. This verification step should be as light as possible to avoid intolerable overheads. The system is highly critical, and its safety has an incredible value. Moreover, the formal verification of the whole system at each adaptation process is considered to be unfeasible because of the resources and time limitation at run-time. We present the system as a modular connected structure. It is a reconfigurable distributed system that can change its characteristics at run-time operations. Figure 23 presents an abstract model of the system which is a 14 module system that represents the different lines of the railway network. Each module represents one metro line with its trains and characteristics. It describes its capacity, structure and its connection to other lines. We assume that modules links represent the connections points between different lines of the railway network. The white rectangles are the stable parts of the system and the red rectangles represent the unstable parts/lines: their behaviour is not fixe at run-time. They are the object of new configurations to cope with the environment requirements.

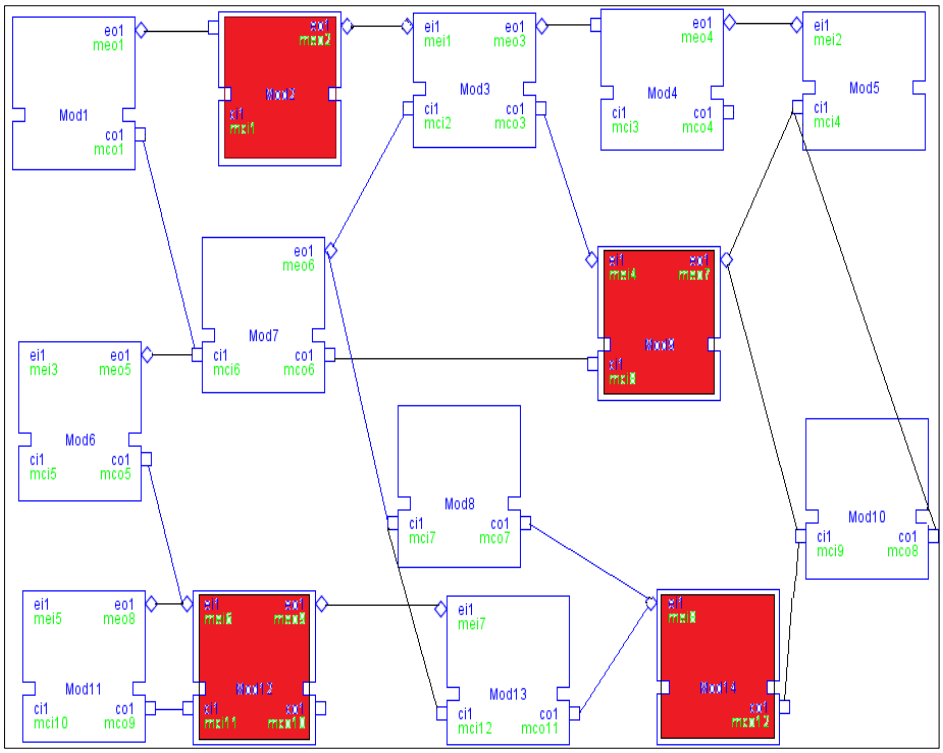


Figure 23 Reconfigurable railway network structure

These reconfigurations are due to an increased demand to enlarge the line capacity, the quality of service or to extend the line to new parts of the urban area of the city.

4.3.2.3 Contribution

The proposed model checking approach deals with distributed reconfigurable models, where a set of components or modules are considered to be unstable (change their behavior at run-time process) and could be also unspecified at design time and are known only at run-time. Moreover, the classical techniques enable to check the system every time the unspecified components are resolved or modified at design time. Indeed, the time and space required for the verification could be considerable and since many configurations are resolved only while the system is operating, the total overhead in resolving them has to be as small as possible. To get over this problem, we propose a two-phase verification approach that enables the designer to:

(i) Deal with reconfigurable scenarios and incomplete specification at design time and,

(ii) Generate a set of constraints to be checked for the unstable or unspecified parts of the system. Those constraints are verified at run-time against the new configuration of the component once it is available.

A complete over view is given in Figure 24. It presents two verification levels: at design time, the incomplete specification system is represented by a particular labelled transition system.

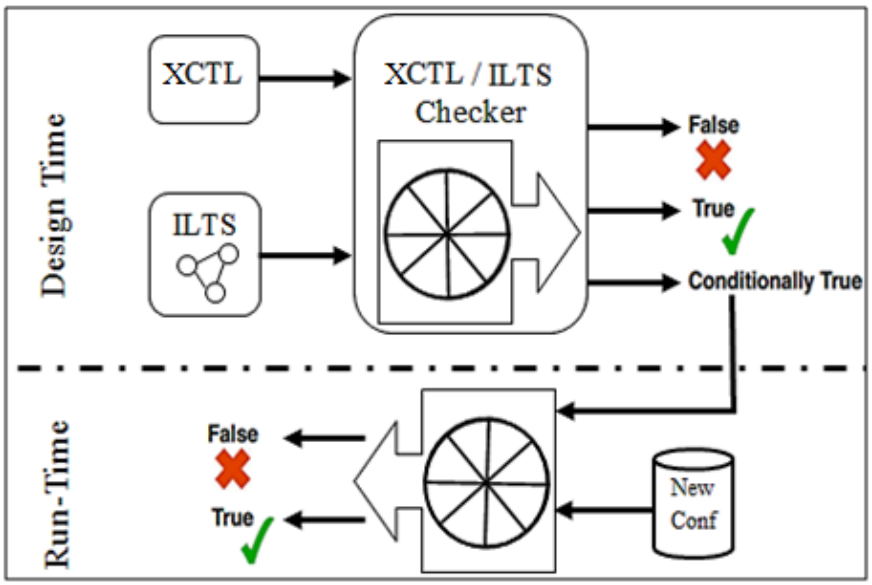


Figure 24 New Verification Approach

It is an Incomplete Labelled Transition System dealing with specified and unspecified states. It contains two different states categories: the first are known as stable states which describe a predefined fixed part or task of the system. Here we mean that all the state properties are known at the specification step. The second are defined as unstable states to describe the reconfigurable scenarios of the system which are unknown at design time or could change at run-time. The inputs of the system are the ILTS model and the property to be checked. It will be checked against the desired Reconfigurable CTL properties "XCTL". The proposed model checker approach has three possible results: "False", "true" and "Conditionally True". The results of the proposed verification process differ from the traditional model checker by an extra output namely "Conditionally True". This option generates a set of constraints that will be checked against the reconfigurable module later. During any reconfiguration process, we reduce the verification process from chickening the whole system specification to the check of the generated constrains lists, i.e., only these constraints are checked against the new configuration and not the complete system specification as used before in the standard model checking. Here is the importance of our contribution of the presented approach, i.e., we reduce the verification of the whole process to the verification of the new adaptation scenario.

4.3.3 Incomplete Labeled Transition System

An incompletely labelled transition system (ILTS) is a labelled transition system in which there are two sets of states: stable and unstable states. It can describe the unknown characteristics of the reconfigurable system at the specification step. Formally, it is a tuple (S, s_0, R, L) where:

- S is the set of stable states T_s and unstable states I_s, i.e., $S = T_s \cup I_s$ and $T_s \cap I_s = \emptyset$;

- s_0 is the initial state, the unique entering state, and it is a stable state,
- $R \subseteq S \times S$ represents the transitions between states,
- L is a labelling function that associates a subset of propositions to each stable state.

ILTS is used to specify any incomplete system later. The proposed verification approach is based on this formalism. Here, we present the ILTS of the motivating example showed in Figure 25. It is derived from the abstract net structure model presented in Figure 23. It is an LTS with some special unknown states. The white places represent the predefined (stable) states/structure of the system. The red states represent the reconfigurable states of the system: its characteristics change at run-time. They are the object of new configurations to cope with the environment requirements at the current state of the system. These reconfigurations are due to an increased demand to enlarge the line capacity, the quality of service or extending the line to new parts of the area. (R_2, R_7, R_{11} and R_{14}) are respectively new simple structures of the reconfigurable modules (2, 7, 11 and 14) at the adaptation phase. Then, once the structure is known or could be defined, the generated constraints are applied to be checked against these new specifications. R_2 is checked against the matrix generated to satisfy the desired XCTL formula in the second module.

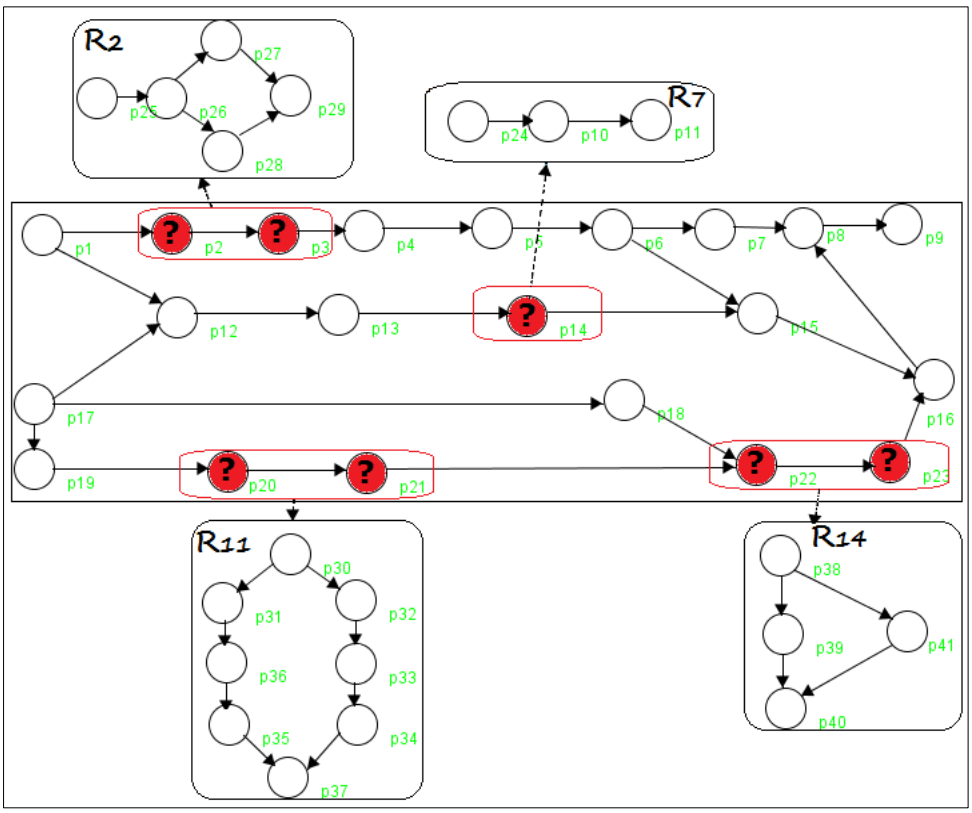


Figure 25 ILTS of the railway model

4.3.4 XCTL Model Checker

Reconfigurable CTL (XCTL) model checking is an extended version of CTL applied to adaptive systems. It has the same semantics as the standard CTL model checking for the "*True*" and "*False*" outputs with an extra definition related to the third possible output namely "*Conditionally True*". We will not recall the standard definition of CTL semantics here; we will just add the new semantics related to unstable states and undefined paths. CTL is basically defined on a state of *LTS*. XCTL should be defined on states of ILTS, $M=(S, s_0, R, L)$. $M, s \models \phi$ means that ϕ could hold in a state s of the ILTS M. The formula ϕ is checked against the whole system with its stable and unstable states. For the stable states, we will get a standard verification process, whereas, in the unstable cases, it will be different. There will be a generation of sub-constraints to be fulfilled by the incomplete sub-system for the aim of satisfying the formula ϕ after the definition of the reconfiguration scenario. The set of constraints that are needed to satisfy the formula ϕ in an unstable state s are saved in a matrix *constr*. Each element $constr(\phi, s)$ is a set of constraints in the form $[(\phi_1, state_1), \ldots, (\phi_n, state_n)]$, meaning that the formula ϕ holds in s if the path XCTL formula ϕ_1 holds in $state_1, \ldots, state_{n-1}$ and the path XCTL formula ϕ_n holds in $state_n$. We present here the semantics of XCTL and we recall the definition of the ILTS system (S, s_0, R, L) to clarify the proposed semantics:

- S is the set of stable states T_s and unstable states I_s, i.e., $S = T_s \cup I_s$ and $T_s \cap I_s = \emptyset$;
- s_0 is the initial state, the unique entering state, and it is a stable state,
- $R \subseteq S \times S$ represents the transitions between states,
- L is a labelling function that associates a subset of propositions to each stable state

The semantics should be defined as follows for the stable states T_s and unstable states I_s:

- $M, s \models \phi \Leftrightarrow \phi \in L(s)$ if $s \in T_s$ and $s \models constr(\phi, s)$ if $s \in I_s$;
- $M, s \models \neg\phi \Leftrightarrow M, s \not\models \phi$ if $s \in T_s$ and $s \not\models constr(\phi, s)$ if $s \in I_s$;
- $M, s \models \phi_1 \wedge \phi_2 \Leftrightarrow M, s \models \phi_1$ and $M, s \models \phi_2$ if $s \in T_s$; and $s \models constr(\phi_1, s)$ and $s \models constr(\phi_2, s)$ if $s \in I_s$;
- $M, s \models \phi_1 \vee \phi_2 \Leftrightarrow M, s \models \phi_1$ or $s \models \phi_2$ if $s \in T_s$; and $s \models constr(\phi_1, s)$ or $s \models constr(\phi_2, s)$ if $s \in I_s$;
- $M, s \models AX\phi \Leftrightarrow (\forall \pi$ such that $\pi_0 = s, M, \pi_1 \models \phi)$ for all paths starting at s, next time ϕ if $s \in T_s$ or next time $constr(\phi, s)$ if $s \in I_s$;
- $M, s \models AF\phi) \Leftrightarrow (\forall \pi$ such that $\pi_0 = s, \exists i$ such that $M, \pi_i \models \phi)$ for all paths starting at s, *eventually* ϕ if $s \in T_s$ or *eventually* $constr(\phi, s)$ if $s \in I_s$;
- $M, s \models AG\phi \Leftrightarrow (\forall \pi$ such that $\pi_0 = s, \forall i\ M, \pi_i \models \phi)$ for all paths starting at s, *always* ϕ or *always* $constr(\phi, s)$ if $s \in I_s$;
- $M, s \models \phi_1 A U \phi_2 \Leftrightarrow (\forall \pi$ such that $\pi_0 = s, \exists i$ such that $(\forall j < i\ (M, \pi_j \models \phi_1)) \wedge (M, \pi_i \models \phi_2))$, for all paths starting at s, ϕ_1 *until* ϕ_2 if $s \in T_s$ or $constr(\phi_1, s)$ until $constr(\phi_2, s)$ if $s \in I_s$;
- $M, s \models EX\phi \Leftrightarrow (\exists \pi$ such that $\pi_0 = s, M, \pi_1 \models \phi)$ there exists a path such that *next time* ϕ if $s \in T_s$ or *next time* $constr(\phi, s)$ if $s \in I_s$;
- $M, s \models EF\phi \Leftrightarrow (\exists \pi$ such that $\pi_0 = s, \exists i$ such that $M, \pi_i \models \phi)$ there exists a path such that *eventually* ϕ if $s \in T_s$ or *eventually* $constr(\phi, s)$ if $s \in I_s$;
- $M, s \models E\ \phi_1 \cup \phi_2 \Leftrightarrow$ if there exists a path π starting from s such that $\exists s_k \in \pi \mid M, s_k \models \phi_2$ if $s \in T_s$ or $s \models constr(\phi_2, s)$ if $s \in I_s$ and $\forall s_i \in \pi$ with $i < k$, $M, s_i \models \phi_1$ if $s \in T_s$ or $s \models constr(\phi_1, s)$ if $s \in I_s$;

- $M, s \models EG\phi \Leftrightarrow$ if there exists an infinite path π starting from s such that $\forall s_i \in \pi$, $M, s_i \models \phi$ if $s \in T_s$ and $s \models constr(\phi, s)$ if $s \in I_s$.

The core of the presented approach is an XCTL model checking algorithm for incomplete models, described using the ILTS formalism. It is based on the CTL model checking [Clarke 1986] and manipulated in order to deal with unstable and incomplete states. The inputs of the algorithm are an XCTL property and an ILTS model. If the ILTS is a stable LTS, i.e., the system specification is already predefined, it behaves as the traditional CTL model checker approach on predefined LTS. On the other hand, if the ILTS contains unknown states or as mentioned before that they are not specified at the specification time, it computes the presented XCTL formulae that shall be guaranteed by the unspecified components later at run-time. More precisely, the presented approach follows these algorithm steps. First, the XCTL formula is parsed and its parsing tree is derived. Usually, the leaves are propositions and the inner nodes are boolean and temporal operators. As CTL model checking, a bottom-up approach is applied to the tree to check if each sub-formula holds. For each node of the tree, the set of the states in which the sub-formula holds is evaluated by parsing the tree, starting from the leaves. The algorithm takes as inputs a subtree S_T of the parsing tree, the formula ϕ, and the ILTS M on which the original formula is evaluated. The tree S_T is a binary tree, where a node representing a unary operator has a single son, while a node representing a binary operator has two sons. We use $S_{T.S}$ to refer to the set of states in M that satisfy the formula represented by the current subtree, $S_{T.left}$ and $S_{T.right}$ to refer to the left and the right subtrees of the current tree (when the root is a binary operator), and $S_{T.son}$ to refer to the subtree of the current tree (when the root is a unary operator). The algorithm can store the elements that satisfy ϕ in a local set X_ϕ. Moreover, the set of constraints that are needed to satisfy the formula ϕ in an unstable state s are saved in the matrix *constr*.

4.3.5 Marking Algorithms

We present here the marking algorithm [Ma 2017b] of the proposed XCTL temporal logic. The inputs are: A model structure M, an XCTL formula ϕ and a subtree t. The constraint matrix is initiated (line 3). *Mark* (φ, s) is a standard CTL marking function dependent on the formula φ. This function is applied once the visited state is a stable one (line 4). Let's assume that *Mark* $(\varphi, s) \in \{$ *Mark*(ϕ, s), *Mark*$(\neg\phi, s)$,*Mark*$(\phi1 \wedge \phi2, s)$, *Mark*$(\phi1 \vee \phi2, s)$, *Mark*$(AX\phi, s)$, *Mark*$(AF\phi, s)$, *Mark*$(AG\phi, s)$, *Mark*$(\phi1AU\phi2, s)$, *Mark*$(EX\phi,s)$, *Mark*$(EF\phi,s)$, *Mark*$(E\ \phi1U\phi2, s)$, *Mark*$(EG\phi,s)\}$. It the same logic used in the standard model checking, i.e., checks if a formula is satisfied by the model. On the other case (line 5), a constraint is generated to be investigated at the adaptation phases. The constraint is a sub-formula that should be verified by this instable state. This constraint is added to the list of the existent constraints (line 6).

```
1: Marking (φ, t, M) {
2: for all (s ∈M.S) {
3: constr (φ, s) = φ ;
4: if (s ∈ M.T_s) {mark (φ, s) }
5: elseif (s ∈ M.I_s) {
6: constr(φ, s) = constr(φ, s) ∪ {s};}}}
```

4.3.6 Degraded Verification Mode

Reconfiguration enables a system to operate in different modes, thus, to be flexible as possible and adapted according the characteristics and requirements of the environment. Openness is also an inherent property, as agents may join or leave the system throughout its lifetime. The proposed verification approach is based on the generation of the constraints to be checked at each reconfiguration scenario.

- In case, we opt to check the $AG\phi$ formula (line 3), i.e., this property has to be satisfied by the whole system model. We generate the corresponding constraints to be respected during any reconfiguration scenario. Before applying the new adaptation, the proposed algorithm checks that the new configuration satisfies the requirements of the generated constraints (line 7).
- If it is true (line 8), the system will operate safely and complete its running task. In various cases, the properties are not respected and the system has to go forward with respect to its safety.
- In this case, the algorithm chooses to degrade the running mode to the second level, i.e., instead of verifying the satisfaction of the property in all the paths, we try to find a possible combination of paths that could be executed by the system (line 9). Then, we move to check the validity of following formula: EGϕ (line 10) that presents the existence of a possible solution for the occurred deadlock state.

```
1: Verif_output R;
2:  While ( R ≠ false) do
3:  if (φ= AGp)
4:   {   R="Conditionally True";
5:      constr(φ, s);
6:      Execute_Reconfiguration();
7:      Verif_constr();
8:      if (R= True) then end;
9:      else {φ:= EGp ;
10:           Verif_constr();} }
11:   if (φ= AXp) OR (φ= AFp)  OR (φ=pAUq) {
12:    if (R= "True") then end;
13:     if (R= "Conditionally True")
14:       {  constr(φ, s);
15:          Execute_Reconfiguration();
16:          Verif_constr();
17:        if (R= True) then end;
18:          else φ:= SUBSTITUE (φ; "A"; "E"); }
19:          Verif_constr();}
20:   end while
```

- For the following three formulas: "AXϕ, AFϕ, pAUq" (line 11), it is possible that the properties are satisfied at the stable part of the system (line 12), i.e., the reconfiguration scenario will not affect the requirement of the system. Then, the verification results should be "True".
- Otherwise the corresponding constraints are generated and should be checked against the updated parts of the system (line 14).
- In case of the non-satisfaction of the desired constraints, we can opt to the degradation (line 18). Then, we check respectively the following constraints formulas "EXϕ, EFϕ, pEUq" (line 19).

The degradation strategy is presented in a summarized view in Figure 26. Safety is a crucial element in critical systems. Here, the railway network is always the subject of different addition/removing of trains to various lines. As a solution for the increased demand to enlarge the system structure and the quality of service respecting its safety, we can think about the existence of a possibility to apply the desired property in the possible lines instead of the entire network. We opt to check the validity and existence of paths that satisfy to desired target. For example, if we aim to double the speed of some trains: then, it will affect the safety distance between the components of the network. Let's consider the property *p*="double the speed", then we check: *EFp* instead of *AFp.* Similarly, if we hope to add two extra trains in the network from certain stations to cover the large demand: ϕ= "add two extra trains", then we check $EX\phi$ instead of $AX\phi$. We look for proving the existence of safe options to improve the quality of service of the system. Thus, we can guarantee the service continuously with its safety to satisfy the user requirements.

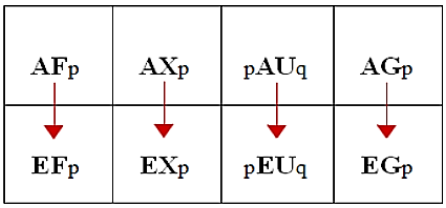


Figure 26 Degradation approach

4.3.7 Discussion

This chapter highlights a double-phase approach to efficiently verify reconfigurable distributed systems, in which some components may dynamically change at run time. The idea aims to introduce an optimized formal certification approach for reconfigurable systems: much more useful to save time and memory resources. The purpose is to:

- Optimize the verification process, i.e., the needed time and space resources after any modification of the system behavior.

- Based on the use of a separated modular verification approach and the results of the previous verification, we avoid the repetition of many extra unnecessary tasks during the certification of a reconfigurable scenario.
- To support the methodology, a new semantics of the temporal logic CTL is proposed to deal with the incomplete labelled transition systems of an adaptive system.
- A new marking algorithm to concretize the approach is presented. A new degraded verification algorithm is proposed as a solution for the deadlock states after applying any adaptation process.
- To support this built framework, correctness tests will be evaluated, we will check the validity of the results of the proposed XCTL model checking compared to the standard model checking. Scalability of the approach will be considered in our future work.

4.4 Summary

We highlight in this chapter two contributions: the first is a simulation method of probabilistic reconfigurable system with energy and memory control. We presented a probabilistic simulation algorithm that can supervise the energy and memory consumption of such a reconfigurable system. This algorithm offers the possibility to detect resources violation before applying any adaptation scenario, i.e. save the system specification from running without resources. It can also simulate the proposed model using different probabilistic strategies. The second is related to formal verification, i.e., we presented an export method of GR-TNCES model to PRISM model and a new online formal verification approach of reconfigurable systems. It can avoid repetitive useless tasks that slow down the formal certification at any adaptation scenario. It reduces the formal verification of the whole system specification to a certain number of constraints that could be verified later at the reconfiguration time, i.e. reducing the verification time and the varication resources. This approach is proposed to cover the limits of traditional model checking method coping with large reconfigurable systems. This work could be extended in many directions. At the moment, we are working on the implementation of the algorithm and to explore a new symbolic approach.

Chapter 5
ZIZO1.1: New Environment for Modeling, Simulation and Verification of APDECS

Contents

5.1 Introduction……..……....64
5.2 New Environment ZIZO1.1 ……....64
5.2.1 ZIZO1.0 ……64
5.2.2 New Version of ZIZO: Architecture……..…65
5.2.3 Implementation…….…….....67
5.2.4 System Modeling and Simulation…….....68
5.3 Case Study 1: Skid Conveyor System……..……..69
5.3.1 Structure…….....69
5.3.2 System Modeling…..……70
5.3.3 Simulation and Optimization…….……73
5.3.4 Discussion…….……73
5.4 Case Study 2: IPV4 ZeroConf…….....74
5.4.1 Principles and Challegnes……..…….....74
5.4.2 IPV4 ZeroConf: Modeling……..……..…74
5.4.3 IPV4 ZeroConf: Simulation…….……76
5.4.4 IPV4 ZeroConf: Formal Verification…….……..77
5.5 Discussion……..…….....80
5.6 Summary…….....81

ZIZO1.1: NEW ENVIRONMENT FOR MODELING, SIMULATION AND VERIFICATION OF APDECS

5.1 Introduction

We present here the new version ZIZO1.1* [Khlifi 2018a] and its usefulness in modeling, simulation and certifying adaptive probabilistic discreet event control systems [Kouskoulas 2013]. The tool is used to model and simulate two case studies: the IPV4 ZeroConf protocol and an automotive transport system. This chapter is published in [Khlifi 2018a].

5.2 New Environment ZIZO1.1

5.2.1 ZIZO1.0

ZIZO1.0 is an R-TNCES modeling and random simulating tool written in C# programming language for the Windows platform and developed in LISI laboratory of INSAT (Tunisia) presented in [Salem 2015]. It is the first version of ZIZO: Its originality consists in featuring the simulation of a built R-TNCES and highlighting the reconfiguration aspect of a DRCS, which are not offered in any other Petri nets editor. The main window of ZiZo GUI shown in Figure 27 comprises five dockable frames: Menu Bar, Model Arborescence, Place Properties, the Document Explorer and the Debug Window.

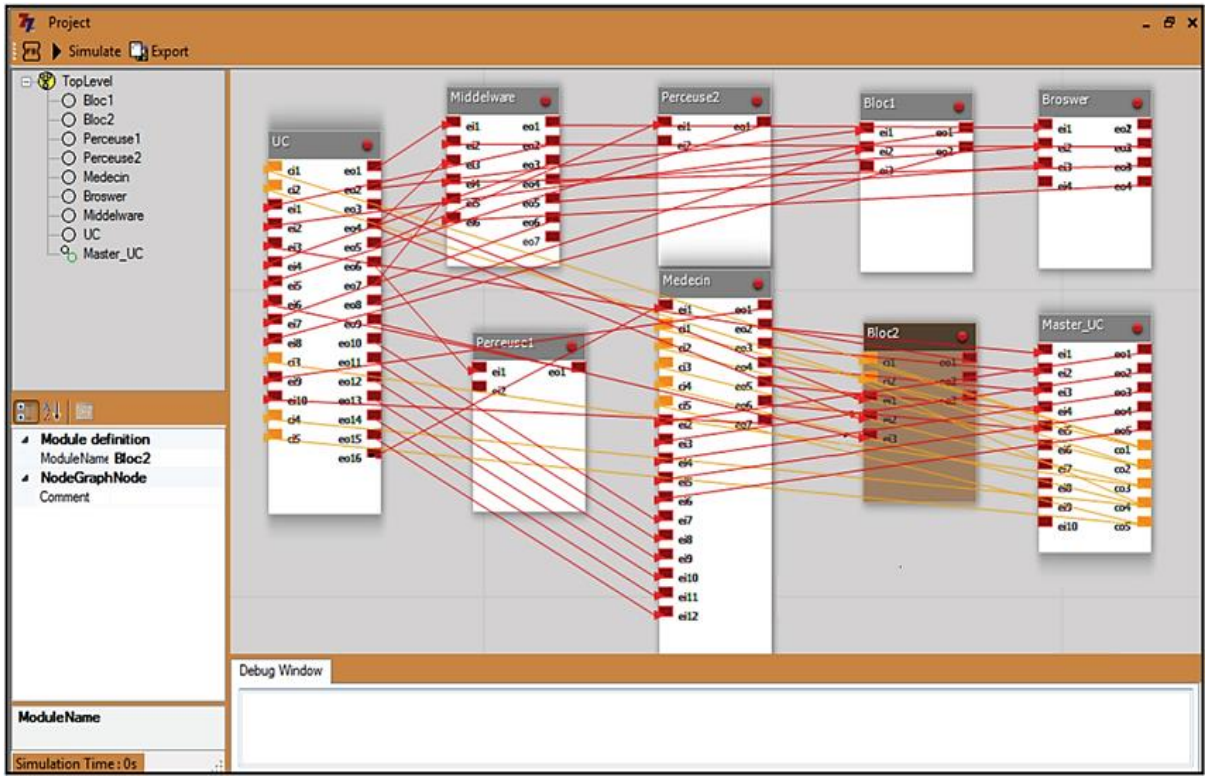


Figure 27 Main window of ZIZO1.0.

ZIZO1.0 is capable of:

- creating several modules within the same model;
- interconnecting modules by input/output condition and event signals;
- randomly simulating the created model to detect any eventual deadlock;
- storing the created model in a special file format (*.pnt);
- loading a created model to edit it and/or simulate it;
- exporting the model to the model-checker SESA.

5.2.2 New Version of ZIZO: Architecture

In the conception part of the new version of the tool ZIZO1.1 [Khlifi 2018a], we used StarUML [StarUML, 2018] which offers the world application structure, behavior, and architecture, business process and data structure. It unifies every step of development and integration from business modeling, through architectural and application modeling, to development, deployment, maintenance, and evolution. It helps to specify, visualize, and document models of software systems, including their structure and design, in a way that meets all of these requirements. It defines thirteen types of diagrams, divided into three categories: structure diagrams, behavior diagrams, and interaction diagrams. We present here the architecture of ZIZO1.1. In the following picture Figure 28, we present the global class diagram of the tool which contains four packages: Components, Link Edition, Project Handler and Simulation.

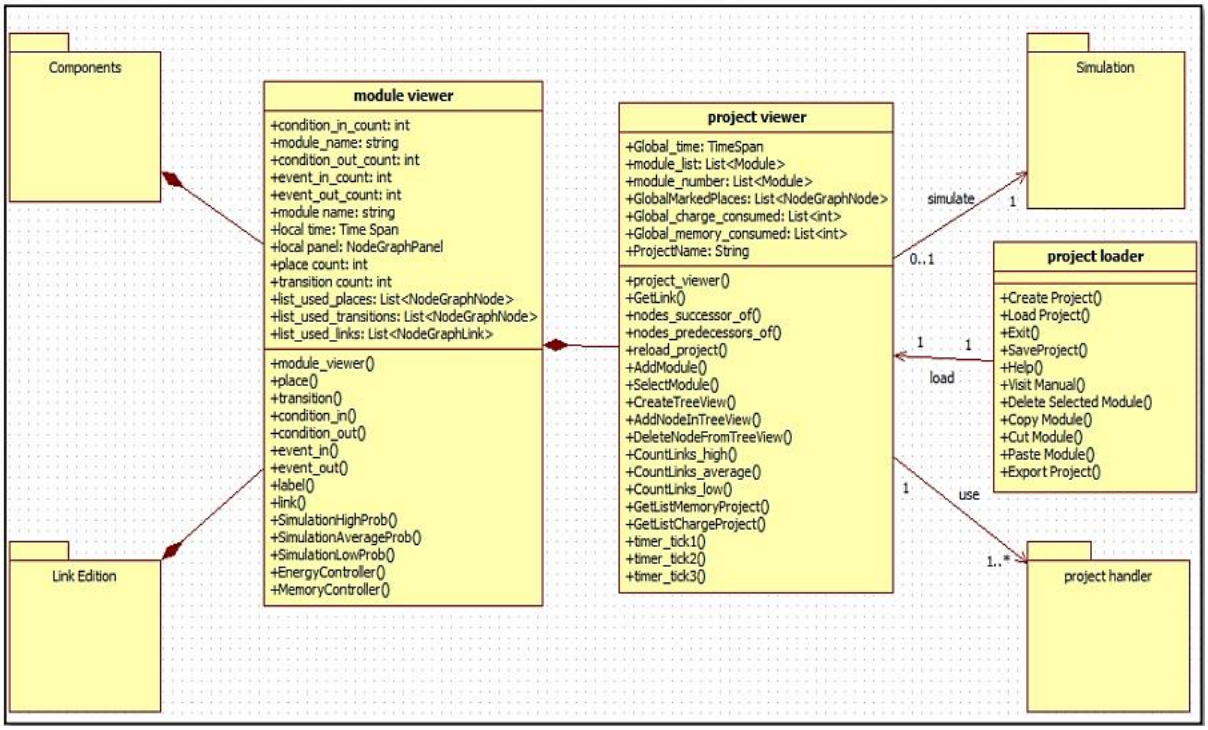


Figure 28 Packages of ZIZO1.1.

Table 4 Links Description

	Source	Destination	Role
Normal Link	Place Transition	Transition Place	Enable token crossing.
Condition Link	Place Place Condition_In	Transition Condition_out Transition	Transfer the state of the parent place (If it is marked or not).
Event Link	Transition Transition Event_In	Transition Event_out Transition	Stimulate a transition when the previous is fired.
Inhibitor Link	Place	Transition	Enable firing a transition only if parent place is not marked.

These packages are connected and interact using three more classes which are: module viewer, project viewer, and project loader.

- Package Link Edition: It presents various types of arcs used to connect the project elements. Each arc is used to link specific element. The following table Tab. 4 presents how the links are affected (source, destination) and why they are used.
- Package Simulation: This package contains three simulation classes according to the probability strategies: High, average or low. If the designer chooses the high probabilistic simulation strategy, then only the highest probabilistic paths will be executed, i.e., the token will visit only the branches with high probability. The same rules are applied for average and low simulation strategies. A warning message is displayed once the energy reserves reach the minimal allowed capacity. It is a parameter chosen by the designer.
- Package Project Handler: It enables to perform certain operations on the project: (i) Saving the project, (ii) Loading the created project for editing and simulation, (iii) Exporting a project to PRISM model checker.
- Package Components: This package showed in Figure 29 describes all different elements of ZIZO1.1 and its relation to each other. They can be places, transitions, modules, Condition_In, Condition_Out, Event_In, Event_Out and labels.

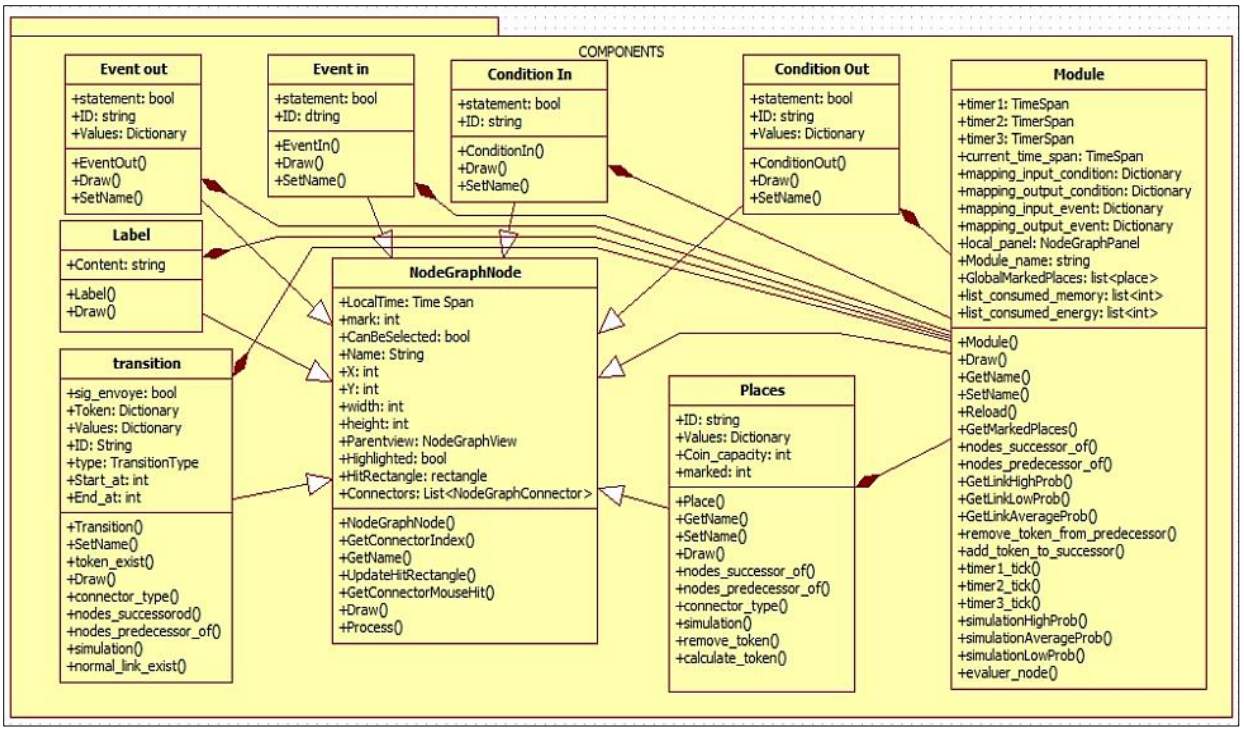


Figure 29 Package Components

5.2.3 Implementation

ZIZO1.1 is developed for the modeling and simulation of APDECS. It helps to deal with distributed architectures using condition/event signals. We present briefly the utilities which we have used to develop our project. For the Hardware, we disposed a computer which has these technical specifications:

- Processor: Intel (R) Core (TM) i5-3230M CPU @2.60GHz
- RAM: 4,00Go
- System type: OS 64 bit
- Operating System: Windows 8.1 Professional 2013 Microsoft Corporation.

For the software part, we used Visual Studio Ultimate 2012 [Visual Studio Ultimate 2012] which offers the possibility to use flexible agile planning tools to enable incremental development techniques and agile methodologies. It offers also advances modeling, discovery, and architecture tools to describe your system and helps to ensure that your architecture vision is preserved in the implementation. We used the development Platform: Microsoft.Net Framework 4.5 that provides a comprehensive and consistent programming model for building applications that have visually stunning user experiences and seamless and secure communication. As a programming language, Microsoft C# is an object-oriented programming language designed for building a wide range of enterprise applications that run on the .NET Framework. C# code is compiled as managed code, which means it benefits

from the services of the common language run-time. These services include language interoperability, garbage collection, enhanced security, and improved versioning support. Finally, using this work environment, we come out to build ZIZO1.1 which enables the following features:

- Modeling of probabilistic distributed system based on the GR-TNCES formalism, i.e. extending the tool from R-TNCES (ZIZO1.0) to GR-TNCES.
- Editing and connecting modules to each other through condition and event signals.
- Simulation of the global model with a token game animation, the control of the simulation depending on the memory and energy reserves and showing the evolution of the reserve at run-time: the consumed as well as the memory and energy reserves, i.e. adding the various probabilistic features to the simulation algorithm compared to the last version of the tool.
- Choice of the simulation type according to the probability level: (High, Average, Low). It was not offered by ZIZO1.0, i.e. dealing with probabilistic functionality.
- Extraction of curves for the memory and energy consumption during the simulation time, this is also a new functionality of the presented ZIZO version.
- Export of the model to the PRISM model checker by the generation of the model's PRISM code, it is also a new feature to enable the formal verification of the probabilistic model and save time for writing the complete system specification job.
- Loading and saving a model.

5.2.4 System Modeling and Simulation

ZIZO1.1 enables the modeling of distributed probabilistic systems using GR-TNCES formalism. The content of each module can be designed using different components. Figure 30 represents an example of a GR-TNCES module. It contains places, transitions, condition output, condition input, event input, event output, normal link, condition link, event link, probabilistic parameter on the links, memory and energy reserves and the recharge period for the energy resources. We develop the simulation algorithm of the built model; it is shown through a token game animation. The color of each fired transition changes to red and one token from the memory and energy resources is consumed. There are three simulation strategies according to the probability level: High, Average and Low. The simulation process consumes memory and energy resources depending on the chosen simulation. We have to control resources availability before the starting and even at run-time process. The idea is to check the resources' availability before starting the system simulation to make sure that no energy violation will be faced during the running process.

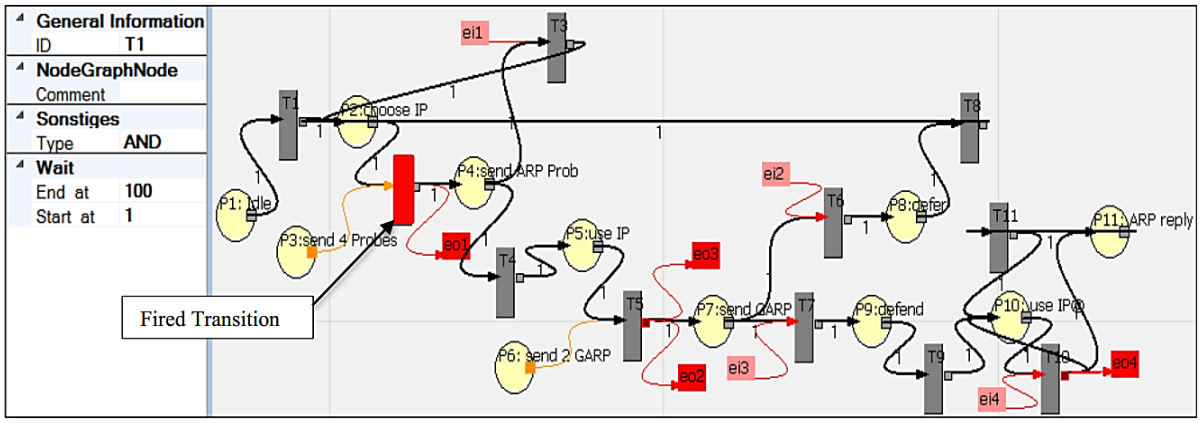


Figure 30 Module Example.

5.3 Case Study 1: Skid Conveyor System

In this Section, we present the automotive transport system model based on GR-TNCES formalism using the environment ZIZO1.1. For the purpose of optimizing the energy consumption of the old system, we present the requirements of the proposed transport system model.

5.3.1 Structure

The existent automotive transport system at ZeMA is based on three conveyor parts, each one equipped with one motor to move the chassis on it from one position to the next one. The user could only activate/deactivate the three motors simultaneously. Even if there is just one car in the transport system, all the motors are active continuously. Our contribution in this point aims to save the energy consumed by the motors if there is no car at such a position of the skid. In order to realize energy-efficient operations, the system is extended by a control unit and six inductive sensors. The first sensor is placed 2.62m and the second one 4.69m from the start point of each conveyor part. Using these sensors, it is possible to detect the skid position on the conveyor. Inactive components are switched into an energy efficient state. For this purpose, the system is extended by a control unit and six inductive sensors. For example, once the chassis is in the third conveyor part, the motor of the second and first one should be switched to *Standby* mode. The activation/deactivation of the motors is controlled based on the car position. Since we have a fixed chassis position on the skid, the inductive sensors enable us to determine the chassis position. This information can be used in the assembly task for example and then, we differentiate three different cases:

(i). If a rising edge is detected by the first sensor, then the skid reaches the conveyor and the associated motor must be turned *ON*,
(ii). If there is a rising edge at the second sensor, then the skid is in the middle of the conveyor. The motor is switched to *Standby* mode for an exemplary cycle time of 10 seconds,

(iii). If a falling edge is detected by the second sensor, then the skid leaves the conveyor part and the first motor must be switched *Off*. Monitoring the skid position has a further advantage.

In order to provide any energy efficient operation of the system, we have to install a control system that allows switching *ON* and *Off* all components at the right time. The central unit of the system forms a programmable logic controller (PLC) which connects all the sensors, the motors and the conveyor part of the system. The PLC communicates via PROFINET with the drives and a mobile panel. The Siemens PROFIenergy [PROFIBUS 2010] profile is based on PROFINET and allows *Active* and *Standby* modes for the non-used loads during non-productive periods. The drive system is a modular component that ranges from the control unit and the power modules to the motors. The user-handling and control were realized with a mobile panel. It is easy to command the system via touchscreen and buttons. Figure 31 shows the layout within the control components. It represents the control system and the connection among its modules.

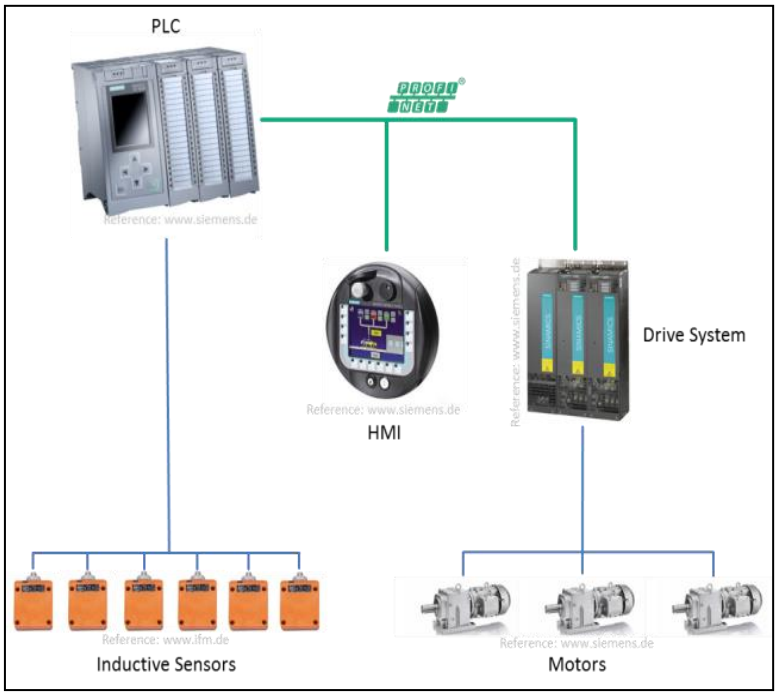


Figure 31 Control system of the skid conveyor.

5.3.2 System Modeling

We model the proposed system model and also the model of the existent system (without control and inductive sensors) using the environment ZIZO1.1. To evaluate the energy optimization of the proposed plant model, it should be compared with the energy consumed by the old production line model. Figure 32 describes the proposed model which is a distributed discrete event system composed of four modules: The car in the conveyor, the

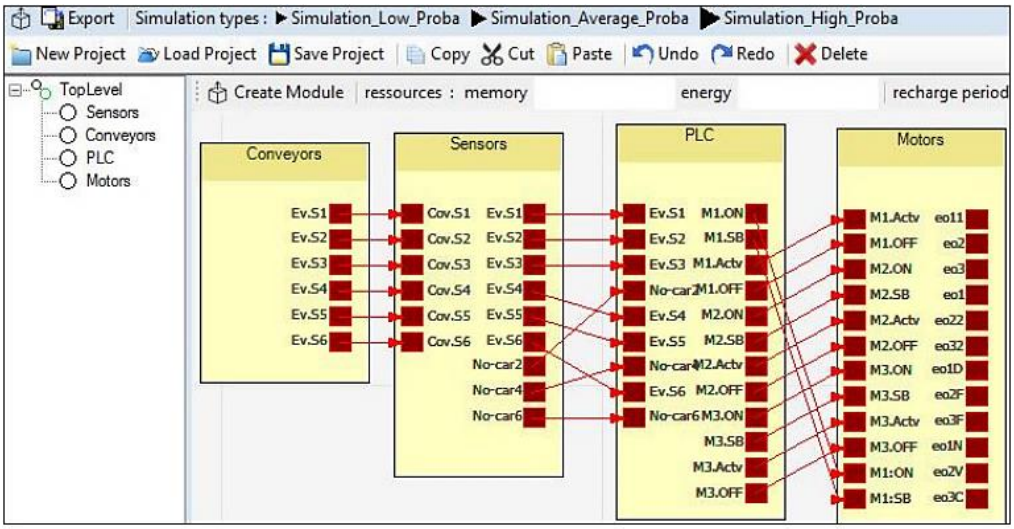


Figure 32 Proposed Transport Model.

sensors, PLC and the three motors. If the sensitive sensor detects the existence of a car in the conveyor, then it sends an event signal to the PLC. It activates and deactivates the corresponding motors according to the car position in the conveyors. The first module contains six events which correspond to the six sensors installed in the skid conveyor (two sensors at each conveyor part). For the sensors module, it receives the events sent by the conveyor and transfers them to the PLC module. It has three extra-events denoted by "*No-Car2*", "*No-Car4*", and "*No-Car6*" which correspond respectively to the events received from sensors number two, four and six to notify the PLC that the car has left the conveyor part. The third module corresponds to the PLC module which controls the whole system.

The PLC receives signals from the sensors to control the state of the motors (*Active, Standby, Off*). The *Event_In* "*M1.ON*", "*M1.SB*", "*M1.Act*", and "*M1.Off*" see Figure 33, correspond respectively to control the states of the motors "*Idle, Standby, Active* and *Off*". Figure 33 introduces the model of the first and second motors, i.e., it describes the transition between the different states of the motors. The pink rectangles correspond to input events received from the PLC to stimulate the firing of the corresponding transitions. The motor keeps the running mode till it receives another PLC signal. The event-in "M1.SB" pushes the first motor to switch from *Active* to *Standby* mode; "*M1.Act*" is used to reactivate the motor after the energy efficient mode. Figure 34 shows the PLC model that corresponds to the transition between different states of the system. The pink rectangles correspond to *Event-In* signals received from the inductive sensors. The red rectangles correspond to *Event-Out* signals that control the motors' states. The PLC model represents the logical and the temporal control unit to manage the entire system.

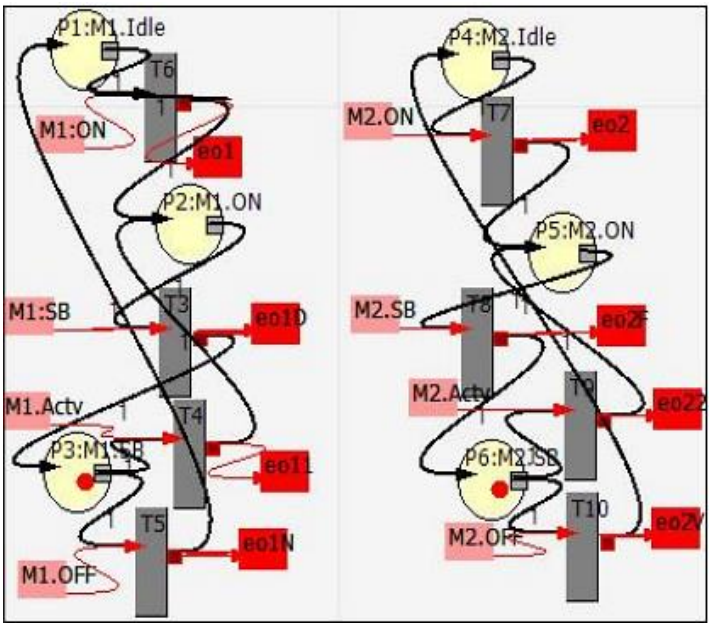


Figure 33 Model on motors.

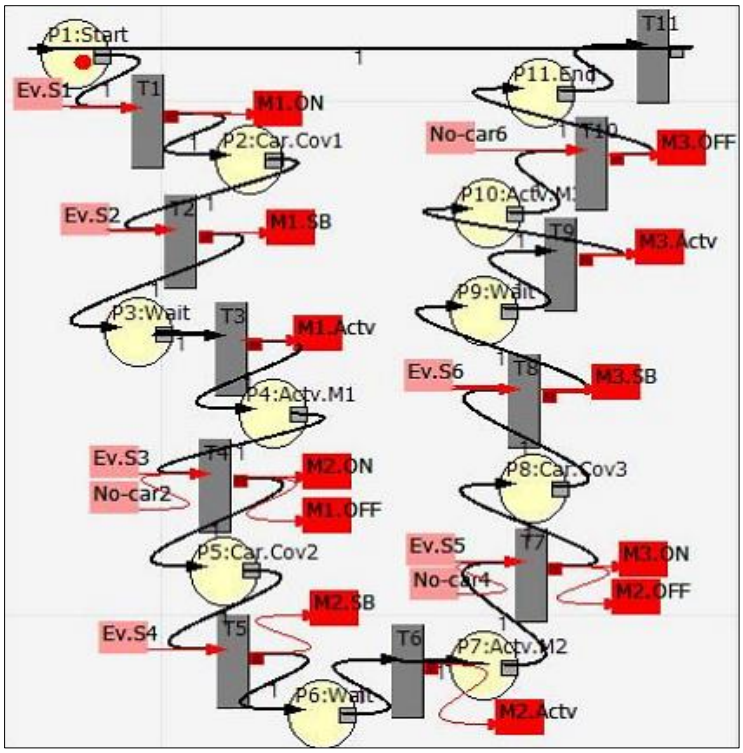


Figure 34 Model of the PLC.

Basically, it has to ensure the following states: “Start”, “Activate Motor 1”, “Car in conveyor 1”, “Wait 10 seconds”, “Activate Motor 1”, “Car in conveyor 2”, “Wait 10 seconds”, “Activate Motor 2”, “Car in conveyor 3”, “Wait 10 seconds”, “Activate Motor 3”, “End”. During the waiting time, the motor is in the *Standby* mode while there is another robot working on the car’s chassis. Then, the motors move the car to the next part of the skid

conveyor. There are additional sensors to detect the workpiece's position on the conveyor. The control strategy is based on the sensors' optimal position to reduce the period in which it is essential to activate two motors for the task of moving the car. We detect exactly the suitable time for deactivating the current motor and activating the next one.

5.3.3 Simulation and Optimization

There are two system/model variants: An old one where all the motors could be switched only simultaneously and manually between *Active*/*Stop* modes, and another system model where each motor could be monitored and switched independently. The new model also features additional sensors to detect the position of the workpiece on the conveyor. Accordingly, the motors need to be into operation mode, are automatized by means of the PLC. To evaluate the energy optimization of the proposed model, we refer to the consumption of the old system. The standard plant model contains only touch screen for the control of the three motors. It is used to activate and deactivate all the motors which are continuously in a running mode except the delay to work on the chassis by another robot. As shown in Figure 35, the basic model presents only two modules: "Control Panel" and "Motors". The red signals between these modules correspond to the activation and deactivation control events of the motors. For the simulation of the systems' resources, we suppose that each motor consumes four energy units (tokens) per second in the running mode, one token in the *Standby* mode and zero unit if it is *Off*. We execute the simulation using the environment ZIZO to evaluate its consumption. The energy consumption curves are shown in Figure 36 during the simulation time (40 seconds). This figure illustrates the evolution of the token number needed by the system in that period. The curves present three horizontal parts. It corresponds to the period in which the motors are deactivated in the old model and the *Standby* mode in the proposed model. The other portions correspond to the motors' activation period and the energy consumed by the three motors to move the car from one position to the next one.

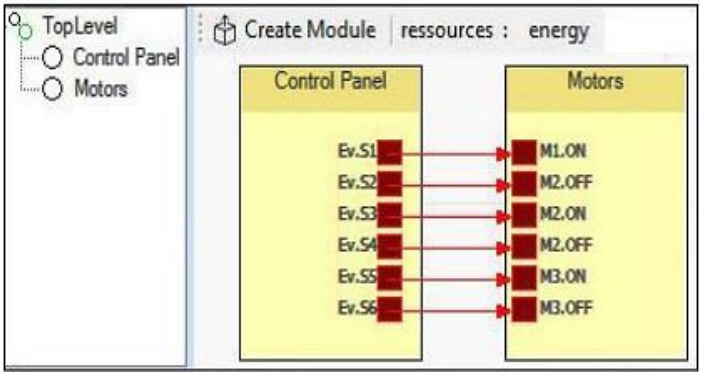


Figure 35 Standard system's model.

5.3.4 Discussion

Figure 36 shows the consumption curves of the proposed energy efficient mode in the right graph and the curves of the energy consumption of the old model in the left one. In the energy efficient mode, usually there is only one motor which is active. This idea is based on the detection of the car position to activate and deactivate the corresponding motors. We aim to reduce the period in which we need two motors to move the car from one skid to the next one through the optimal position of the sensors. The curves describe the energy needed by each system during the simulation time. We notice that there is an important reduction of the energy consumed in the proposed model. For the first part (2-5 seconds), the consumption is highly reduced (from 22 to 9 tokens) since only one motor is activated instead of three motors compared to the old model. To move the car to the second position (13-16seconds), the proposed system model consumed 22 tokens. On the other hand, the basic model needs 44 energy units for the same task which is a valuable optimization. In fact, the sensors detect the car position and the PLC controls the activation and deactivation of the motors: It deactivates the first motor and turns on the second one. For the third skid part of the system, this strategy enables us to save 24 energy units compared to the basic plant model presented in the left part of Figure 36.

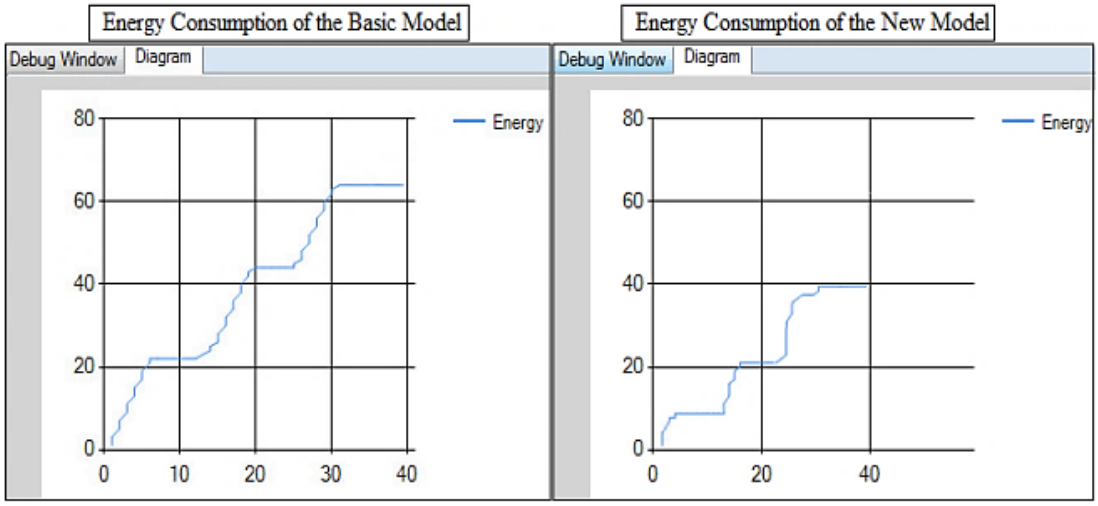


Figure 36 Optimization of energy consumption.

5.4 Case Study 2: IPV4 ZeroConf

5.4.1 Principles and Challegnes

IPV4 ZeroConf is a probabilistic protocol executing many devices with limited memory and energy resources. In large scale of wireless networks, significant packets loss is inevitable which could consume the memory and energy of the devices. The communication and the storage of the messages need memory and energy reserves: it needs one unit of memory and

energy resources to transfer a message from one device to another. The selection of a valid address is not guaranteed because it is an uncertain process. Reconfiguration processes are probabilistic, i.e., they are necessary in case of an address conflict. In some cases, the protocol has to change its address as a solution for the address conflict. In this work, the authors try to deal with these questions: (i) How can we model the unpredictable behaviors? (ii) How can we simulate and check probabilistic adaptive discrete event systems? (iii) How can we certify the system's correctness and safety? i.e., to verify the violation of memory, energy and probabilistic real-time constraints, (iv) and how to prove that there are no deadlock states due to the resources violation or real-time constraints?

5.4.2 IPV4 ZeroConf: Modeling

The model of the network is shown in Figure 37: each module corresponds to a device model. The whole model connects the devices using condition/event signals and the tokens represent the memory and energy resources to be consumed by the system. The battery is recharged after a desired time period chosen by the user. The probabilities on the arcs are used to describe the nondeterministic behaviors of the protocol. Realtime constraints are assigned on the transition parameter ("start_at": the desired time to fire the transition and "end_at": the last possible instant to fire the transition, i.e., out of this period, the transition is not fired even if there is a token in the predecessor state).

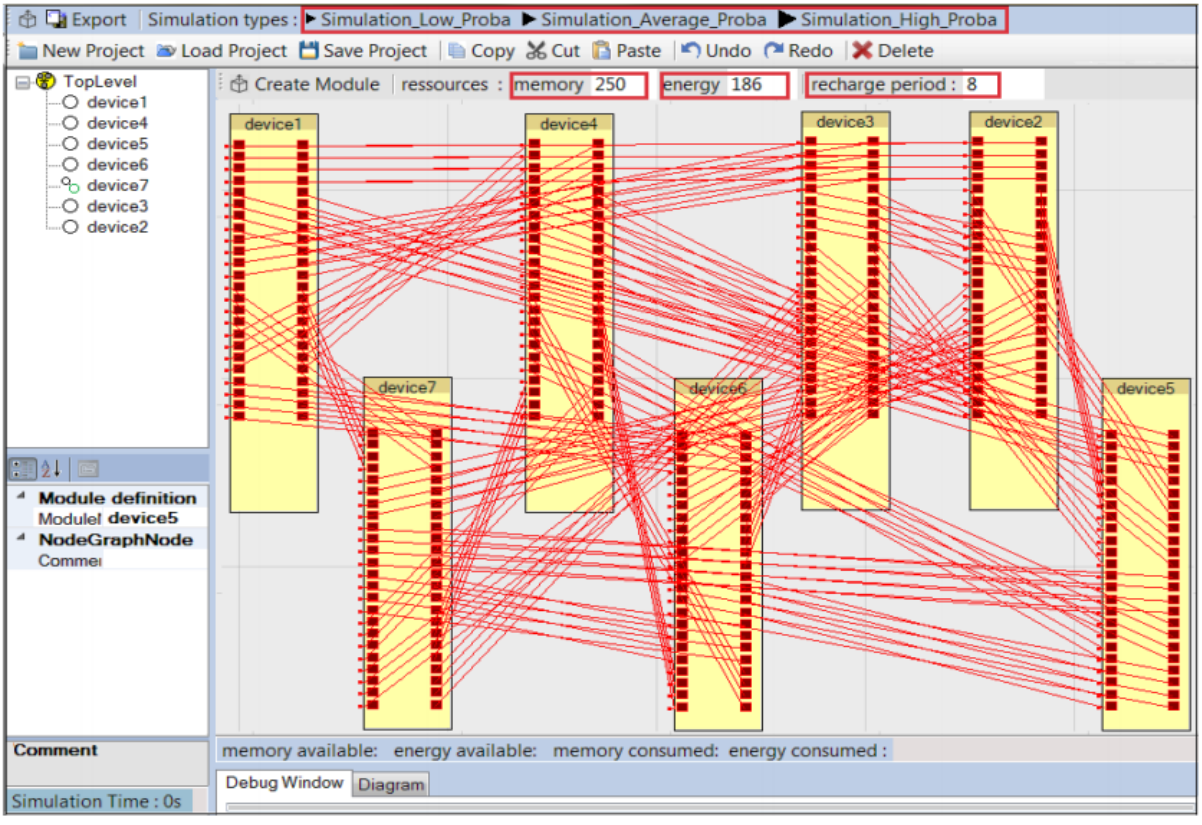


Figure 37 Devices network

The description of the GR-TNCES model of one device using ZIZO1.1 is presented in Figure 30. The devices have the same model description while they have the same behavior. Each device starts at the idle state ("P1: idle"), then chooses an IP address ("P2"), and sends four probes to the rest of the network. It is a repetitive task modeled with a condition to evaluate the probe's emission ("P3: send four probes"): T2 will be fired only if P3 is marked. An Event_Out is created to send the ARP probes. Once the four ARP probes are sent ("P4: Send ARP probes") and the device receives an ARP reply (ei1 in T3), the protocol is restarted ("P2: choose IP address"). If the device does not receive any message, it fires T4 to reach "P5: Use IP". Then, the device confirms its ownership of the address to the rest of the network by sending two gratuitous ARP. The place (P6) models the condition of sending two GARP packets, i.e., it is modelled by two events *eo2* and *eo3*. If it receives a GARP packet during the first second of the use of the address, it must defer the address ("P8: Defer") and restart the protocol through "P2: Choose IP". Then, it can continue to use the chosen address ("P10: use IP") and consequently, the device becomes a network member.

5.4.3 IPV4 ZeroConf: Simulation

Once the network model is built using ZIZO1.1, we have to certify the marking properties and the time constraints: the messages synchronization and the respect of the response delay (2 and 10 seconds). In addition, we have to check that the memory and energy resources will not be violated at runtime operation or during a reconfiguration process. To run the system, (i) all initial states ("P1: idle") should be marked with a token, and (ii) a transition could be fired only if all its inputs are valid: the time constraints, Event_In and Condition_In. We simulate two scenarios: a seven-device model and a 14-device model, i.e., we have realized many experiments to optimize and supervise the needed resources for the probabilistic protocol. For the 14 devices, we obtain a GR-TNCES model with 154 places, 213 transitions and 1484 Event_In/Out to exchange the messages. Upon the definition of the model, the simulation can be tracked by selecting a token game animation. Once the simulation is finished, a trace file is displayed in the debug window. We simulate the protocol through the high probability strategy and the average strategy. During the simulation, the color of each fired transition is changed to red and the simulator shows the variation of the consumed and the available memory and energy resources. For the 14 devices network, if we simulate the system with only 100 units of memory and only 20 units of energy, we obtain a warning message because the resources are insufficient see Figure 38. Thus, using only these resources, the system will violate its reserves and will reach a deadlock state. According to the simulation scenarios, we conclude that the minimal resources for this system are 250 units of energy, 186 units of memory tokens and the recharge period for the battery is eight seconds. We simulate the system model without any violation problem or any blocking states. The debug window shows the evolution of the simulation through a trace file description.

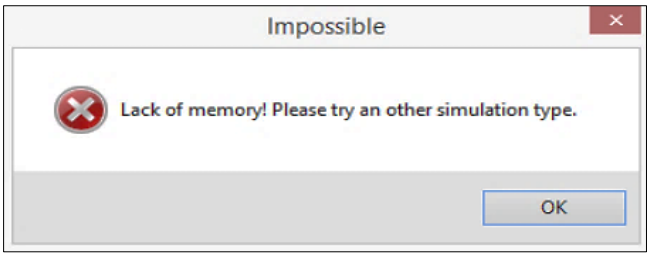


Figure 38 Memory warning message

Different simulation scenarios are established to obtain the optimal model with the minimal memory and energy resources. Figure 39 shows the evolution of the consumed energy and memory resources during the simulation time (94 seconds). The resources consumption increases during the time due to the increase in the probabilistic network traffic. The system consumes 250 of energy tokens and 186 of memory tokens that is the optimal consumption of the simulated model.

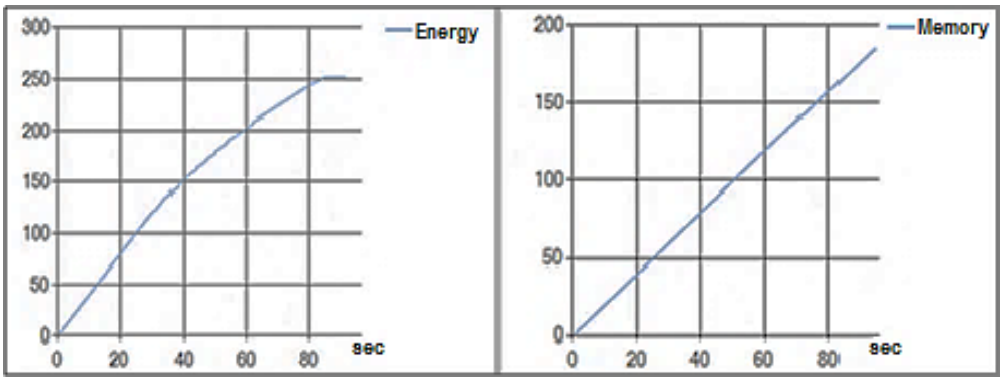


Figure 39 Curves for memory and energy consumption

5.4.4 IPV4 ZeroConf: Formal Verification

The authors aim to formally certify the safety of the IPV4 ZeroConf model, i.e., verifying if the model respects the required resources and if any reconfiguration scenario dealing with adding/removing devices lead to a blocking situation. PRISM is a probabilistic model checker, a tool for formal modeling and analysis of the systems that exhibit random or probabilistic behavior. It is based on the model's reachability graph to analyze different systems and answer many typical questions, e.g., what is the probability of a failure? Typical properties which can be verified are boundedness of places, liveness of transitions [Uzam 2016], and reachability of states. The tool's input is a description of a probabilistic system written in the PRISM language. Using PRISM, we can check the correctness of a model, i.e., the existence of a deadlock state in which the system could not progress or violate the memory and energy resources or real-time constraints. PRISM introduces the model type as a discrete time Markov chains, continuous time Markov chains, Markov decision processes,

probabilistic automata and probabilistic timed automata [PRISM 2015]. Then, the names of the modules and their variables are declared. Finally, the model states and their probabilities to reach the successor states are described. We aim to check the probability that the system violates its resources during a reconfiguration process. For that purpose, the GR-TNCES model is exported to the PRISM model using ZIZO1.1. Figure 40 describes a part of the ZeroConf protocol's PRISM model. The temporal/functional properties specified by the users based on CTL could be checked manually.

```
global Eng : [0..15] init 15; //variable
global Mem: [0..15] init 15;
module  device1
      p : [0..11] init 1; // number of place in the graph
//consumption: if energy reserve contain more than 3 unit then we continue consumption
      [] p=1   & Eng>3 & Mem >3   -> 1:(p'=2)&(Eng'=Eng-1) & (Mem'=Mem-1);
      [] p=2   & Eng>3 & Mem >3 -> 1:(p'=4)&(Eng'=Eng-1) & (Mem'=Mem-1)  ;
      [] p=3    -> 1:(p'=3) ;
      [] p=4   & Eng>3 & Mem >3 -> 0.2:(p'=2)&(Eng'=Eng-1)& (Mem'=Mem-1)   +  0.8:(p'=5)&(Eng'=Eng-1)& (Mem'=Mem-1);
      [] p=5   & Eng>3 & Mem >3 -> 1:(p'=7)&(Eng'=Eng-1)& (Mem'=Mem-1)   ;
      [] p=6     -> 1:(p'=6) ;
      [] p=7   & Eng>3 & Mem >3 -> 0.2:(p'=8)&(Eng'=Eng-1)& (Mem'=Mem-1)   +  0.8:(p'=9) &(Eng'=Eng-1) & (Mem'=Mem-1);
      [] p=8   & Eng>3 & Mem >3 -> 1:(p'=2) &(Eng'=Eng-1)& (Mem'=Mem-1)  ;
      [] p=9   & Eng>3 & Mem >3 -> 1:(p'=10) &(Eng'=Eng-1)& (Mem'=Mem-1)  ;
      [] p=10  & Eng>3 & Mem >3 -> 0.2:(p'=11) &(Eng'=Eng-1)& (Mem'=Mem-1) + 0.8:(p'=10) &(Eng'=Eng-1)& (Mem'=Mem-1);
      [] p=11  & Eng>3 & Mem >3 -> 1:(p'=10) &(Eng'=Eng-1)& (Mem'=Mem-1)  ;

//control: if low battery(Eng<=3), we periodecally charge the battery , for example here we choose as period of recharge: 4
      [] p=2    & (Eng=3 | Eng=2 | Eng=1) & (Mem=3 | Mem=2 | Mem=1)-> 1:(p'=4)&(Eng'=Eng+4) &(Mem'=Mem+4);
      [] p=4    & (Eng=3 | Eng=2 | Eng=1) & (Mem=3 | Mem=2 | Mem=1)-> 0.2:(p'=2)&(Eng'=Eng+4) &(Mem'=Mem+4)+0.8:(p'=5)&(Eng'=Eng+4)&(Mem'=Mem+4);
      [] p=5    & (Eng=3 | Eng=2 | Eng=1) & (Mem=3 | Mem=2 | Mem=1)-> 1:(p'=7)&(Eng'=Eng+4)&(Mem'=Mem+4);
      [] p=7    & (Eng=3 | Eng=2 | Eng=1) & (Mem=3 | Mem=2 | Mem=1)-> 0.2:(p'=8)&(Eng'=Eng+4) &(Mem'=Mem+4)+0.8:(p'=9) &(Eng'=Eng+4) &(Mem'=Mem+4);
      [] p=8    & (Eng=3 | Eng=2 | Eng=1) & (Mem=3 | Mem=2 | Mem=1)-> 1:(p'=2) &(Eng'=Eng+4)&(Mem'=Mem+4);
      [] p=9    & (Eng=3 | Eng=2 | Eng=1) & (Mem=3 | Mem=2 | Mem=1)-> 1:(p'=10) &(Eng'=Eng+4) &(Mem'=Mem+4) ;
      [] p=10   & (Eng=3 | Eng=2 | Eng=1) & (Mem=3 | Mem=2 | Mem=1)-> 0.2:(p'=11) &(Eng'=Eng+4)&(Mem'=Mem+4)+0.8:(p'=10) &(Eng'=Eng+4)&(Mem'=Mem+4);
      [] p=11   & (Eng=3 | Eng=2 | Eng=1) & (Mem=3 | Mem=2 | Mem=1)-> 1:(p'=10) &(Eng'=Eng+4)&(Mem'=Mem+4);
endmodule

module  device2
      p1 : [0..11] init 1; // number of place in the graph
```

Figure 40 PRISM code of ZeroConf protocol mode.

E[F " deadlock "]

We aim to check the resources control strategy: the optimal control of the memory and energy resources during run-time processes. We have to prove that there is no resource violation during a probabilistic reconfiguration process. The following CTL formula is applied to check if the system is deadlock free. This formula is proven to be false, i.e., we do not have any blocking situation during this running process with the activation of the memory and energy controllers. The energy resources are guaranteed by the optimal recharge period, i.e., the model established by ZIZO1.1 and exported to PRISM is well-controlled and does not lead to a blocking situation at a run-time process. We use the probabilistic temporal logic PCTL which is an extension of computation tree logic for the probabilistic verification of the described properties. We formally certify that the memory resources are sufficient at any reconfiguration process. The memory is needed for data storing, the synchronization and transfer of the messages in the network. For each new

device connection, the network has to exchange a number of messages request and confirmation for the validity and availability of such an IP address. One has to prove that the system's memory is continuously available to exchange messages in the network. The idea is to instantaneously check if there are the needed resources for all the system scenarios. We have proven that it is not possible to reach the end states without memory resources. PRISM certifies that the probability to reach a state with a problem of memory resources is null. The following PCTL formula is an example of the checked properties.

```
Pmin=? [p=11& Mem=0]
```

This formula is evaluated and proven by PRISM, i.e., the probability that the system reaches the last state of the device module with no memory resources is null as shown in Figure 41. Thus, GR-TNCES can specify and supervise the model's resources at run-time operation. Similarly, the network devices need energy resources to keep their running mode. It is a critical resource for the connectivity of the devices to the network. We certify that there is no lack of energy resources during a reconfiguration scenario.

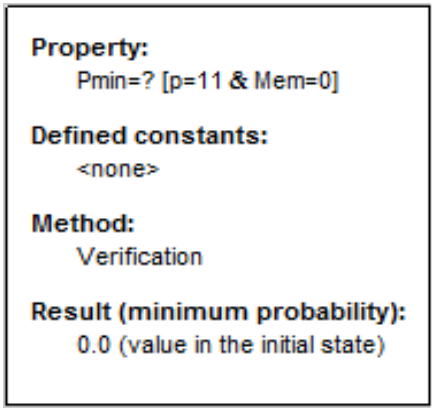


Figure 41 Probabilistic Property.

According to PRISM, the probability to reach an end state without energy reserve is null. The following PCTL formula is an example of the checked formulas and it is proven to be null.

```
Pmin=? [p=11 & Eng=0]
```

We verify that the model meets users' requirements, i.e., any adaptation process do not lead to a deadlock state as proven by the Red Cross in PRISM in Figure 42.

Even after 334 steps, there is no blocking situation faced by the protocol. Each step refers to an operation done by the protocol: Sending a message, adding a device, and removing a device. Thanks to GR-TNCES, ZIZO1.1 and PRISM, we have modelled, simulated and

certified the safety of the proposed case study. We evaluated the resources consumption and supervised their availability before any adaptation process.

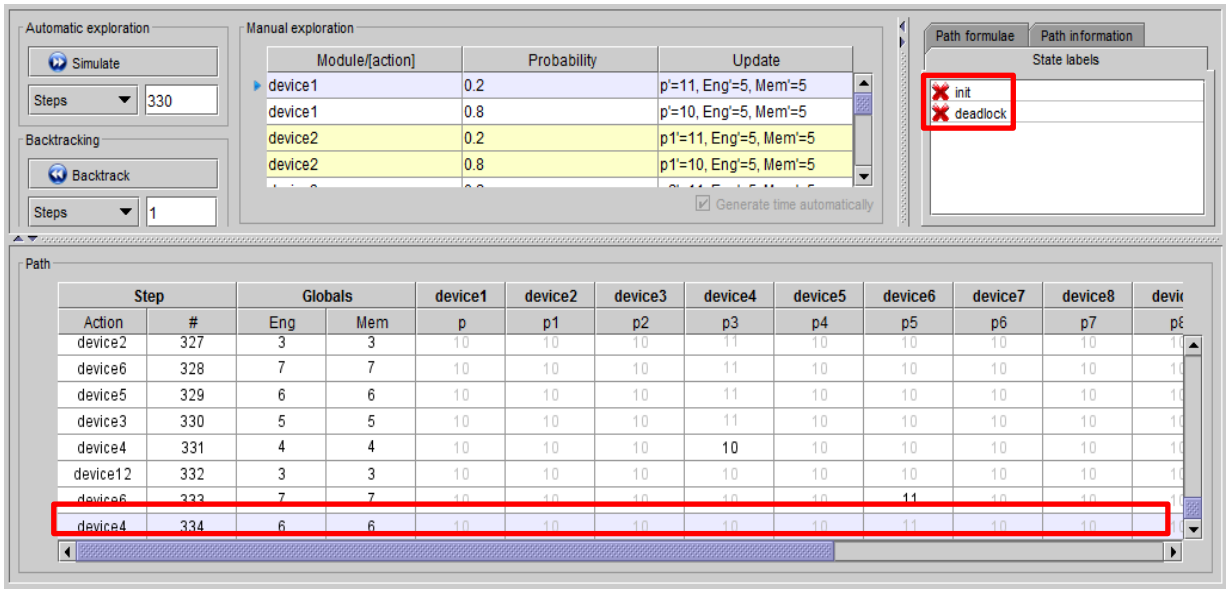


Figure 42 Simulation at step 334.

5.5 Summary

This chapter presents a new original visual environment for the modeling, simulation and verification of APDECS using the formalism GR-TNCES. The new version of ZIZO is used to model probabilistic distributed reconfigurable architectures and connect modules using event and condition links. It enables the monitoring and supervision of the consumed resources to avoid their violation. A mapping algorithm used to export any ZIZO model for a formal verification using PRISM is also presented. Compared with the previous studies on formal methods, we are able to model unpredictable reconfigurable discrete event system running under limited resources. A new model of the automotive transport system is developed and simulated using ZIZO to evaluate its energy consumption compared to old system model. The presented model offers a significant energy optimization compared to the standard model. The reported results of the improved system show an important reduction of the consumed energy, i.e., more than 60% in the first part of the conveyor and more than 40% in the second part. Moreover, IPV4 ZeroConf protocol is also a considered case study to concretize these contributions. This protocol is modeled and simulated using ZIZO1.1. The resources consumption is analysed and we guarantee their availability. The IPV4 PRISM model is also generated and checked to certify the correctness and the safety of the GR-TNCES models.

Chapter 6
Conclusion and Perspectives

Contents

6.1 Context……..…… 82
6.2 Problems.....…… 82
6.1 Output and Contributions..…… 83
6.2 Tool ……..…… 83
6.1 Perspectives ....…..…… 84

CONCLUSION AND PERSPECTIVES

6.1 Context

The development of probabilistic critical distributed reconfigurable control systems is an open issue due to the complex dependencies, the artifacts, the real-time constraints, the limited energy and memory resources and the reconfigurable behavior of these systems. Such a system could violate its real-time constraints, it could also violate its battery or its memory. Therefore, an expressive and optimal specification could improve and help to identify and describe all the features and the behavior of the system. In our thesis, we focus on the modeling, simulation and formal verification part since the current based Petri nets formalisms are not able to specify all the aforementioned features and characteristics. Moreover, these systems should be easily modified and reconfigured after any evolution of the environment within the system behaves. Therefore, each reconfiguration scenario should meet energy, memory and real time constraints since the system could violate its resources during run-time process which leads to a deadlock state and dangerous effects. This thesis tries to extend the formalism R-TNCES with new solutions for the optimal specification and control of predictable as well as unpredictable behaviors and tries to cover the problem of time, memory and energy violation. The second problem deals with the formal verification of reconfigurable properties i.e., how can we formally verify such a reconfigurable property or an uncompleted specification. On the other hand, a complete visual environment named ZIZO was developed for the modeling, simulation and formal verification of adaptive probabilistic systems. This environment was applied to an IPV4 ZeroConf network protocol and an automotive transport system. We presented in this work a complete approach ranging from specification, modeling, and simulation to the real implementation of the proposed automotive transport system.

6.2 Problems

This research work focusses on three problems related to specification, modeling, simulation and formal verification of APDECS:

- The first theoretical modeling problem is to extend the formalism R-TNCES since it is not able to specify probabilistic systems running under memory and energy resources constraints. More precisely, how to enable the modeling and simulation of probabilistic reconfigurable behavior, energy and memory resources using R-TNCES formalism.
- The second theoretical problem reveals the formal verification of reconfigurable properties; i.e., how can we formally certify such a system with the properties that

could change during run-time operation since the current verification approach cannot deal with reconfigurable properties. How to optimize the verification time? How to reduce the certification process from verifying the whole system to just checking the affected states by the reconfiguration scenario.

- The third focuses on the technical part of the previous theoretical issues; i.e., we need a complete environment to model, simulate and formally verify probabilistic reconfigurable systems running under limited energy and memory resources.

6.3 Output and Originalities

This thesis focuses on the modeling, simulation and formal verification of probabilistic adaptive systems running under resources and real-time constraints. The contributions consist on presenting a complete approach running from system specification, simulation to formal verification. The following points were introduced in this research work:

(i) For the modeling part, a new extension of the formalism R-TNCES and a new based specification approach are proposed, it enables the specification and supervision of probabilistic systems under resources constraints. This contribution is applied later to an automotive transport system.

(ii) For the verification part: a new algorithm for the probabilistic simulation of system behavior and the energy and memory resources is implemented to guarantee its availability. Moreover, a new model checking approach dealing with uncompleted and adaptive systems is also presented. It enable to reduce verification time and optimize the verification process from verifying the complete specification at each adaptation process to just checking and investigating the affected part. Moreover, an export module connecting GR-TNCES models to PRISM model checker is implemented here.

(iii) Here, these previous theoretical contributions were integrated to a new software version ZIZO1.1 that offers all the cited features to make the modeling and verification tasks more simple.

6.4 Tool

In this thesis, a new version ZIZO1.1 is introduced: it is a new visual environment for the modeling, simulation and formal verification of APDECS using the new formalism GR-TNCES. The new version of ZIZO presents three new features related to modeling, simulation and formal verification.

(i) For the modeling, it implements GR-TNCES and it is used to model probabilistic distributed reconfigurable architectures and connect modules using event and condition links. It enables also to model the memory and energy resources.

(ii) For the simulation, it offers a probabilistic simulation according to the selected probability level, monitoring and supervision of the consumed energy and memory resources to avoid such a resources violation which is an undesirable blocking state.

(iii) For the formal verification, the tool offers also an automatic transformation from GR-TNCES model to PRISM model used to formally certify the system model. It generates the Prism model to make the verification easy and save time.

6.5 Perspectives

This thesis focuses on the modeling, simulation and formal verification of probabilistic adaptive systems running under resources and real-time constraints. The contributions consist on presenting a complete approach running from system specification, simulation to formal verification. As a future work, we suggest:

- The validation of the proposed skid conveyor model through a real energy data measurement of the transport system will be done in a next project.
- Regarding the formal verification part, our contribution states the preliminary steps to address the runtime model checking of adaptive distributed systems. This work could be extended in many directions. The implementation of the algorithm and exploring new symbolic approach should be done.
- We presented only XCTL logic, thus, in the future work the full CTL logics extensions should be supported to deal with security issues in intelligent systems.

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my supervisors Prof. Mohamed Khalgui, Prof. Georg Frey and Dr. Olfa Mosbahi. I thank Prof. Khalgui for his thoughtful guidance, instructive advice and constant encouragement. He patiently taught me to pay attention to details in doing research and writing papers. I could not have imagined having a better adviser and mentor for my Ph.D study. Prof. Georg Frey is my supervisor in Saarland University. I was lucky to have him as one of my supervisors. His speeches are always succinct and explicitly, which never hidden his enthusiasm. I appreciate him for his professional guidance on my research work and warm help. I would like to specially express my gratitude to Dr. Olfa Mosbahi, who has done a great contribution to various pieces of my research work during the past 3 years. She is always ready to revise and comment my research work. I can hardly make progress in doing research without her guidance, criticism, and help. Besides my supervisors, I would like to express my gratitude to Prof. Zhiwu Li, who has done a great contribution to my research work. His criticism was really constructive and improved the quality of our research work. My sincere thanks also to all my colleagues and friends at University of Carthage and Saarland University. Finally, I would like to thank the members of jury for accepted to review my thesis.

REFERENCES

[Koren 1999] Koren Y., et al.: "Reconfigurable Manufacturing Systems", CIRP Annals, Volume 48, Issue 2, Pages 527-540, ISSN 0007-8506, 1999.

[Ben Salem 2016] Ben Salem M. O, Mosbahi O, Khalgui M, Jlalia Z, Frey G, Smida M: "BROMETH: Methodology to design safe reconfigurable medical robotic systems", International Journal of Medical Robotics and Computer Assisted Surgery, DOI: 10.1002/rcs.1786, 2016

[Khlifi 2017] Khlifi O, Mosbahi O, Khalgui M, Frey G: "New Verification Approach for Reconfigurable Distributed Systems", The 12th International Conference on Software Engineering and Applications (ICSOFT), Madrid, Spain, 2017.

[Genter 2007] Genter G, Bogdan S, Kovacic Z and Grubisic I: "Software tool for modeling, simulation and real-time implementation of Petri net-based supervisors", In 2007 IEEE International Conference on Control Applications, pages 664–669, 2007.

[Kopetz 2003] Kopetz H: "Time-triggered real-time computing", Annual Reviews in Control, vol. 27, no. 1, pp. 3–13, 2003

[Gherbi 2006] Gherbi A. and Khendek F: "UML Profiles for Real-Time Systems and their Applications", Journal of Object Technology, vol. 5, no. 4, pages 149–169, 2006. (Cited on page 23)

[Hamid 2010] Hamid B. and Krichen F: "Model-based engineering for dynamic reconfiguration in DRTES", In Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, pages 269–276. ACM, (Cited on page 23) (2010)

[Starke 2002] Starke Peter H, and Roch S: "Analysing signal-net systems", Professoren des Inst. für Informatik, 2002.

[Baier 1998] Baier Christel: "On algorithmic verification methods for probabilistic systems", Universität Mannheim, 1998.

[Clarke 2008] Edmund M. Clarke and Allen E. Emerson: "Design and synthesis of synchronization skeletons using branching time temporal logic", In 25 Years of Model Checking, pages 196–215. Springer, 2008.

[Baier 2008] Baier Christel, Joost-Pieter Katoen et al: "Principles of model checking", volume 26202649. MIT Press Cambridge, 2008. (Cited on page 19)

[Koh 1991] Koh I. and DiCesare F.: "Checking liveness in Petri nets using synchronic concepts". [Online] Available: http://www.koreascience.or.kr/article/CFKO199111919674120.page

[Khlifi 2018a], Khlifi Oussama, Mosbahi Olfa, Khalgui Mohamed, Frey Georg, Li Zhiwu: "Modeling, Simulation and Verification of Probabilistic Reconfigurable Discrete-Event Systems under Energy and Memory Constraints", Iranian Journal of Science and Technology, Transactions of Electrical Engineering. 2018

[Kouskoulas 2013] Yanni Kouskoulas, David Renshaw, Andre Platzer and Peter Kazanzides: "Certifying the safe design of a virtual fixture control algorithm for a surgical robot", In Proceedings of the 16th international conference on Hybrid systems: computation and control, pages 263–272. ACM, 2013.

[Klotzbücher 2012] M Klotzbücher and H Bruyninckx: "Coordinating robotic tasks and systems with rFSM statecharts", JOSER: Journal of Software Engineering for Robotics, vol. 3, no. 1, pages 28–56, 2012.

[Ye 2015] J Ye, Z Li and A Giua: "Decentralized supervision of Petri nets with a coordinator", IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 45, no. 6, pp. 955-966, (2015)

[Ma 2017b] Ma Z, Li Z. W, Giua A: "Characterization of admissible marking sets in Petri nets with conflicts and synchronizations," IEEE Transactions on Automatic Control, vol. 62, no. 3, pp. 1329-1341, (2017).

[Salem 2014] Salem M. O. B, Mosbahi O, and Khalgui M: "PCP-based solution for resource sharing in reconfigurable timed net condition/event systems", in Proc. of Adaptive Discrete Event Control Systems, ADECS, Tunisia, pp. 52-67. (2014)

[Wu 2012a] Wu N. Q and Zhou M. C: "Modeling, analysis and control of dual-arm cluster tools with residency time constraint and activity time variation based on Petri nets", IEEE Transactions on Automation Science and Engineering, vol. 9, no. 2, pp. 446-454, (2012)

[Wu 2015] Wu N.Q, Zhou M.C. and Li Z.W "Short-term scheduling of crude-oil operations: Petri net-based control-theoretic approach", IEEE Robotics and Automation Magazine, vol. 22, no. 2, pp. 64-76. (2015)

[Kumar 2015] Pranav Srinivas Kumar, et al, ROSMOD: "A toolsuite for modeling, generating, deploying, and managing distributed real-time component-based software using ROS", In 2015 International Symposium on Rapid System Prototyping (RSP), pages 39–45, 2015.

[Murata 2002] Satoshi Murata, Eiichi Yoshida, Akiya Kamimura, Haruhisa Kurokawa, Kohji Tomita and Shigeru Kokaji: "M-TRAN: Self-reconfigurable modular robotic system", IEEE/ASME transactions on mechatronics, vol. 7, no. 4, pages 431–441, 2002.

[Roch 2000a] Stephan Roch: "Extended computation tree logic", In Workshop Concurrency, Speci & Programming, number 140 in Informatik-Bericht. Citeseer, 2000.

[Roch 2000b] Stephan Roch: Extended computation tree logic: Implementation and application, 2000. (Cited on pages 20 and 21)

[Rouff 2012] Christopher Rouff, Richard Buskens, Laura Pullum, Xiaohui Cui and Mike Hinchey: "The AdaptiV approach to verification of adaptive systems", In Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering, pages 118–122. ACM, 2012. (Cited on page 19)

[Schlegel 2004] Christian Schlegel: "Navigation and execution for mobile robots in dynamic environments: an integrated approach", PhD thesis, University of Ulm, 2004. (Cited on page 74)

[Li 2012] Li Z. W, Liu G. Y, Hanisch M-H, and Zhou M. C: "Deadlock prevention based on structure reuse of Petri net supervisors for flexible manufacturing systems", IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 42, no.1, pp.178-191. (2012)

[Shousha 2012] Marwa Shousha, Lionel Briand and Yvan Labiche: "A uml/marte model analysis method for uncovering scenarios leading to starvation and deadlocks in concurrent systems", Software Engineering, IEEE Transactions on, vol. 38, no. 2, pages 354–374, 2012. (Cited on page 23)

[Khlifi 2015] Khlifi O, Mosbahi O, Khalgui M, and Frey G: "GR-TNCES: New extensions of R-TNCES for modelling and verification of flexible systems under energy and memory constraints," in Proc. of Int. Conf. on Soft. Eng. and App, ICSOFT-EA, Colmar, France, pp. 373-380. (2015)

[PRISM 2015] PRISM model checker, [Online]. Available: http://www.prismmodelchecker.org/. (2015).

[Bohnenkamp 2003] Bohnenkamp H, Van der Stok P, Hermanns H, and Vaandrager F: "Cost-optimization of the IPV4 zeroconf protocol", in Proc. of the Int. Conf. on Dependable Systems and Networks, pp. 531-540. (2003)

[Clarke 1986] Clarke E. M, Emerson E. A, and Sistla A. P: "Automatic verification of finite-state concurrent systems using temporal logic specifications", ACM Trans. Program. Lang. Syst., vol. 8, no. 2, pp. 244-263. (1986)

[Axelsson 2010] Axelsson R, Hague M, Kreutzer S, Lange M, and Latte M: "Extended computation tree logic", in Logic for Programming, Artificial Intelligence, and Reasoning, Berlin Heidelberg: Springer, pp. 67-81. (2010)

[Christel 1997] Christel Baier, Edmund M. Clarke, Vassili Hartonas-Garmhausen, Marta Z. Kwiatkowska, and Mark Ryan: "Symbolic model checking for probabilistic processes", In ICALP 1997, 24th International Colloquium on Automata, Languages and Programming, volume 1256 of Lecture Notes in Computer Science, pages 430–440. Springer-Verlag, 1997.

[Brázdil 2008] Brázdil T, Forejt V, Kretínský J, and Kucera A. "The satisfiability problem for probabilistic CTL," in Proc. of the 23rd Annual IEEE Symp. on Logic in Computer Science, LICS, Pittsburgh, PA, pp. 391-402, (2008)

[Bouyer 2007] Bouyer P.: "Model-checking timed temporal logics", in Proc. of the 5th Workshop on Methods for Modalities, France, pp. 323-341. (2007)

[Sérgio 1995] Sérgio Vale Aguiar Campos, Edmund M. Clarke, Wilfredo R. Marrero, and Marius Minea. Verus: "A tool for quantitative analysis of finite-state real-time systems", In Workshop on Languages, Compilers, & Tools for Real-Time Systems, pages 70–78. ACM, 1995

[Preuße 2012] Preuße S, Lapp H. C, and Hanisch H. M.: "Closed-loop system modeling, validation, and verification", in Proc. of Emerging Technologies & Factory Automation, ETFA, Poland, pp.1-8. (2012)

[Sharifloo 2013] Shrifloo A.M, and Spoletini P: "LOVER: Light-weight fOrmal Verification of adaptivE systems at Run time," Formal Aspects of Component Software, Berlin Heidelberg: Springer, pp. 170-177. (2013)

[Sustainable 2014] Sustainable Glasgow: "Energy and carbon masterplan", Technical report, Glasgow City Council, 2014, [Online]. Available: https://www.glasgow.gov.uk/CHttpHandler.ashx?id=32441&p=0

[Forejt 2011] Forejt V, Kwiatkowska M, Norman G, and Parker D: "Automated verification techniques for probabilistic system", in *Formal Methods for Eternal Networked Software Systems,* Berlin Heidelberg: Springer, pp. 53-113. (2011)

[Chen 2015] Chen Y. F., Li, Z. W., Barkaoui, K, Giua, A.: "On the enforcement of a class of nonlinear constraints on Petri nets", Automatica, vol. 55, pp. 116-124. (2015)

[Andrade 2009] Andrade E, Maciel P, Callou G, and Nogueira B.: "A methodology for mapping SysML activity diagram to time Petri net for requirement validation of embedded real-time systems with energy constraints," in Proc. of the 3rd Int. Conf. on Digital Society, Cancun, Mexico, pp. 266-271. (2009)

[Bai 2016] Bai L. P, Wu N. Q, Li Z. W, and Zhou M.C: "Optimal one-wafer cyclic scheduling and buffer space configuration for single-arm multicluster tools with linear topology", IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 46, no. 10, pp. 1456-1467. (2016)

[Shareef 2010] Shareef A, and Zhu Y: "Energy modeling of wireless sensor nodes based on Petri nets", in Proc. of the 39th Int. Conf. on Parallel Processing, ICPP, San Diego, CA, pp. 101-110. (2010)

[Khlifi 2018b] Khlifi O, Siegwart C, Mosbahi O, Khalgui M, Frey G: "From Specification to Implementation of An Automotive Transport System", submitted in "Communications in Computer and Information Science (CCIS)" published by Springer, 2018

[Ma 2008] Ma F, and Wang J: "Modeling and simulation method of enterprise energy consumption process based on fuzzy timed Petri nets", in Proc. of the 7th world congress on intelligent control and automation Chongqing, China, pp. 4148-4153. (2008)

[Dumitrache 2000] Dumitrache I, Caramihai S. I, and Stanescu A. M: "Intelligent agent based control systems in manufacturing", in Proc. of IEEE Int. Symp. Intell. Control, Rio Patras, pp. 369-374. (2000)

[Wu 2016] Wu N.Q, Zhou M.C, Bai L.P, and Li Z.W: "Short-term scheduling of crude oil operations in refinery with high-fusion-point oil and two transportation pipelines", Enterprise Information Systems, vol. 10, no. 6, pp. 581-610. (2016)

[Kalita 2002] Kalita D. and Khargonekar P. P: "Formal verification for analysis and design of logic controllers for reconfigurable machining systems", IEEE Trans. Robot. Autom, vol. 18, no. 4, pp. 463-474. (2002)

[Gasmi 2016] Gasmi M, Mosbahi O, Khalgui M, Gomes L, and Li Z. W: "R-Node: New pipelined approach for an effective reconfigurable wireless sensor node", IEEE Trans. Systems, Man, and Cybernetics: Systems, no. 99, pp. 1-14. (2016)

[Ratzer 2003] Ratzer A. V, et al,: "CPN tools for editing, simulating, and analyzing coloured Petri nets," in Applications and Theory of Petri Nets, Berlin Heidelberg: Springer, pp. 450-462. (2003)

[Model 2007] Model - Checkers for Net Condition/Event Systems, Available: http://www.vyatkin.org/tools/modelchekers.html. (2007)

[Wu 2012b] Wu N. Q and Zhou M. C: "Schedulability analysis and optimal scheduling of dual-arm cluster tools with residency time constraint and activity time variation", IEEE Transactions on Automation Science and Engineering, vol. 9, no. 1, pp. 203-209. (2012)

[Salem 2015] Salem M. O. B, et al, "ZiZo: Modeling, simulation and verification of reconfigurable real-time control tasks sharing adaptive resources- application to the medical project BROS", in Proc. of 8th Int. Conf. on Health Informatics, Portugal, pp. 20-31. (2015)

[Leslie 1983] Leslie Lamport: "What good is temporal logic?" In IFIP Congress, pages 657–668, 1983.

[Wang 2015] Wang X, Khemaissia, I, Khalgui, M, Li, ZW, Mosbahi, O, and Zhou, MC: "Dynamic low-power reconfiguration of real-time systems with periodic and probabilistic tasks", IEEE Transactions on Automation Science and Engineering, vol. 12, no. 1, pp. 258-271. (2015)

[Norman 2013] Norman G, Parker D, and Sproston J: "Model checking for probabilistic timed automata," Formal Methods in System Design, vol. 43, no.2, pp. 164-190. (2013)

[Suender 2011] Suender C, Vyatkin V, and Zoitl A: "Formal validation of downtime less system evolution in embedded automation controllers", ACM Transactions on Embedded Control Systems, (2011)

[Tong 2016] Tong Y, Li Z, and Giua A: "On the equivalence of observation structures for Petri net generators", IEEE Trans. on Automatic Control, vol. 61, no. 9, pp. 2448-2462. (2016)

[Tong 2017] Tong Y, Li Z.W., Seatzu C, and Giua A: "Verification of state-based opacity using Petri nets", IEEE Trans. on Automatic Control, vol. 62, no. 6, pp. 2823-2837. (2017)

[Wang 2016] Wang X., Li Z. W, and Wonham W.M: "Dynamic multiple-period reconfiguration of real-time scheduling based on timed DES supervisory control," IEEE Trans. on Industrial Informatics, vol. 12, no. 1, pp. 101-111. (2016).

[Uzam 2016] Uzam M., Li Z. W., Gelen G, and Zakariyya R. S: "A divideand-conquer-method for the synthesis of liveness enforcing supervisors for flexible manufacturing systems," Journal of Intelligent Manufacturing, vol. 27, no. 5, pp. 1111-1129. (2016)

[Cong 2017] Cong X., Fanti M. P, Mangini A. M, and Li Z. W: "Decentralized diagnosis by Petri nets and integer linear programming," IEEE Trans. on Systems, Man, and Cybernetics: Systems, DOI: 10.1109/TSMC.2017.2726108. (2017)

[Ma 2017a] Ma Z., Tong Y, Li L, Giua A: "Basis marking representation of Petri net reachability spaces and its application to the reachability problem," IEEE Transactions on Automatic Control 62 (3), 1078-1093. (2017)

[Zhang 2017] Zhang S, Wu N, Li Z, Qu T, and Li C: "Petri net-based approach to short-term scheduling of crude oil operations with less tank requirement," Information Sciences, vol. 417, pp. 247-261. (2017)

[Zhang 2018] Zhang H., Feng L, Wu N, and Li Z: "Integration of learning-based testing and supervisory control for requirements conformance of black-box reactive systems," IEEE Transactions on Automation Science and Engineering, vol. 15, no. 1, pp. 2-15. (2018)

[Bortolussi 2015] Bortolussi L, et al.: "Verification of Complex Adaptive Systems", [Online]: http://homepage.lnu.se/staff/daweaa/papers/2015CASVerification.pdf. (2015)

[Harel 1990] Harel D, et al.: "STATEMATE: A Working Environment for the Development of Complex Reactive Systems", IEEE Trans. on Software Engineering, vol. 16, no. 4, pp. 403-414, (1990).

[Leveson 1994] Leveson, N.G, Heimdahl, M.P.E, Hildreth, H, and Reese, J.D: Requirements Specification for Process-Control Systems, IEEE Trans. Software Eng., vol. 20, no. 9, pp. 684-707, (1994).

[Ross 1997] Ross D., Structured Analysis (SA): "A language for communicating ideas", IEEE Trans. Software Engineering, pp.16-34, (1997).

[Zedan 1999] Zedan H, Cau, A, Chen, Z, and Yang. H.: "ATOM: An object-based formal method for real-time systems", Annals of Software Engineering 7, pp. 235-256, (1999).

[El-kustaban 2012] El-kustaban A, Moszkowski, B, and Cau. A: "Specification Analysis of Transactional Memory using ITL and Ana Tempura", Lecture Notes in Engineering and Computer Science, pp. 176-181, (2012).

[Khlifi 2016] Khlifi O, Siegwart C, Mosbahi O, Khalgui M, Frey G.: "Modeling and Simulation of an Energy Efficient Skid Conveyor using ZIZO", 13th International Conference on Informatics in Control, Automation and Robotics (ICINCO), ISBN: 978-989-758-198-4, pp. 551-558, Lisbon, Portugal, (2016).

[PROFIBUS 2011] PROFIBUS Nutzerorganisation e.V., "Pi White Paper: The PROFIenergy Profile," Karlsruhe, Germany, pp. 10-11, (2010)

[International 2015] "International Organization for Standardization", Energy efficiency and renewable energy sources - Common international terminology - Part 1: Energy efficiency (ISO/IEC 13273-1:2015). (2015)

[Rausch 1995] Rausch M, and Hanisch H.M: "Net condition/event systems with multiple condition outputs", In Emerging Technologies and Factory Automation, ETFA'95, Proceedings., INRIA/IEEE Symposium on, volume 1, pages 592–600. IEEE, (1995)

[Hanisch 1999] Hanisch HM and Lüder A., Modular modelling of closed-loop systems. In Colloquium on Petri Net Technologies for Modelling Communication Based Systems, Berlin, Germany, pages 103–126, (1999)

[Aziz 1996] Adnan A, Kumud S, Vigyan S, and Robert B: "Verifying continuous time markov chains". In International Conference on Computer Aided Verification, pages 269–276. Springer, (1996)

[Bastide 1998] Bastide R., and Buchs, D: Models, "Formalisms and Methods for Object-Oriented Distributed Computing", In: Proceedings of Object-Oriented Technologys, vol. 1357, pp 221-255, Springer, Heidelberg (1998).

[Chan 2001] Chan W., et al: "Optimizing Symbolic Model Checking for Statecharts", IEEE Trans. on Software Engineering, vol. 27, no. 2, pp. 170-190, (2001).

[Khlifi 2017a], Khlifi O, Siegwart C, Mosbahi O, Khalgui, M., Frey, G: "Specification Approach Using GR-TNCES -Application to an Automotive Transport System-", In: 12th Int. Conf. on Soft. Tech, Madrid, Spain, (2017).

[Chen 2014] Chen Y. F., Li, Z. W., and Zhou, M. C: "Optimal supervisory control of flexible manufacturing systems by Petri nets: A set classification approach", IEEE Trans. Autom. Sci. Eng., vol. 11, no. 2, pp. 549-563, (2014).

[Harel 1987] Harel D: "Statecharts: a visual formalism for complex systems", Science of Computer Programming, vol. 8, no. 3, pp. 231-274, (1987).

[Wasserman 1985] Wasserman A: "Extending state transition diagrams for the specification of human-computer interaction", IEEE Trans. Soft. Engineering, vol. 11, no. 8, pp. 699-713, (1985).

[Bernardi 2007] Bernardi S. and Merseguer J: "Performance evaluation of UML design with Stochastic Well-formed Nets", Journal of Systems and Software, vol. 80, no. 11, pages 1843–1865, (2007)

[Nicollin 1990] Nicollin X., et al., "ATP: an algebra for timed processes". In Proceedings of the IFIP TC, vol. 2, 1990.

[Bondavalli 1999] Bondavalli A., Majzik I, and Mura I: "Automated dependability analysis of UML designs". In Object-Oriented Real-Time Distributed Computing, 1999, Proceedings. 2nd IEEE International Symposium on, pages 139–144. (1999)

[Bruyninckx 2001] Bruyninckx H: "Open robot control software: the OROCOS project". In Robotics and Automation, 2001. Proceedings ICRA. IEEE International Conference on, volume 3, pages 2523–2528. (2001)

[Bruyninckx 2013] Bruyninckx H, et al., "The BRICS component model: a model-based development paradigm for complex robotics software systems", In Proceedings of the 28th Annual ACM Symposium on Applied Computing, pages 1758–1764. ACM, (2013).

[Zhang 2013] Zhang J, Khalgui M, Li Z, Mosbahi O, and Al-Ahmari M: "R-TNCES: A novel formalism for reconfigurable discrete event control systems". In Systems, Man, and Cybernetics: Systems, IEEE Transactions on, vol. 43, no. 4, pages 757–772, (2013)

[Khalgui 2011] Khalgui M, Mosbahi O, Li Z. and Hanisch H. M: "Reconfiguration of distributed embedded-control systems", IEEE/ASME Transactions on Mechatronics, vol. 16, no. 4, pages 684–694, (2011)

[Hanisch 1997] Hanisch H. M, Thieme J, Luder A, and Wienhold O: "Modeling of plc behavior by means of timed net condition/event systems", In Emerging Technologies and Factory Automation Proceedings, ETFA, (1997)

[Steinberg 2005] Steinberg M: "Historical overview of research in reconfigurable flight control", Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, 219(4): 263-275, (2005)

[Zhang 2008] Zhang Y, Jiang J. "Bibliographical review on reconfigurable fault-tolerant control systems", Annual reviews in control, 32(2): 229–252, (2008)

[Chandler 1984] Chandler P. R: "Self-repairing flight control system reliability and maintainability program executive overview" in Proc. Nat. Aero. & Electr. Conf. 1984 pp: 586–590, (1984)

[John 1985] John S. E. et al.: "Design issues for fault tolerant-restructurable aircraft control" in 24th IEEE Conference on Decision and Control, DOI: 10.1109/CDC.1985.268630, (1985)

[Gustavo 2017] Gustavo G, Flávio R. W: "Managing Cache Memory Resources in Adaptive Many-Core Systems", in System Level Design from HW/SW to Memory for Embedded Systems, 172–182, DOI: 10.1007/978-3-319-90023-0_14, (2017)

[Daniel 2016] Daniel B. et al., "Configurable memory systems for embedded many-core processors", Available from: http://arxiv.org/abs/1601.00894, (2016)

[Davies 1992] Davies J. and Schneider S: "A brief history of Timed CSP", Oxford University. Computing Laboratory. Programming Research Group, (1992)

[Bianco 1995] Bianco A. and De Alfaro L: "Model checking of probabilistic and nondeterministic systems", In International Conference on Foundations of Software Technology and Theoretical Computer Science, pages 499–513. Springer, (1995)

[Qianchuan 2006] Qianchuan Z., Bruce H. K.; "Formal verification of statecharts using finite-state model checkers", IEEE Transactions on Control Systems Technology, DOI:10.1109/TCST.2006.876921, (2006)

[Boussahel 2016] Boussahel W., "Improving energy efficiency of manufacturing systems through formal analysis of alternative strategies", https://www.shaker.de/de/content/catalogue/index.asp?lang=&ID=8&ISBN=978-3-8440-5154-4, (2016)

[Zhang 2015] Zhang J: "Modeling and Verification of Reconfigurable Discrete Event Control Systems", https://publikationen.sulb.uni-saarland.de/handle/20.500.11880/23165, (2015)

[Pedro 1996] Pedro R. D and Brinksma E: "A calculus for timed automata". In International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems pages 110–129. Springer, (1996)

[StarUML, 2018] [online]. [Accessed 05 Mars 2018]. Available from: http://staruml.io/

[Visual Studio Ultimate 2012] [online]. [Accessed 05 Mars 2015]. Available from: https://www.microsoft.com/en-us/download/details.aspx?id=30678

[The Metro 2017] the metro: a Parisian institution [online]. [Accessed 20 February 2017]. Available from: http://www.ratp.fr/en/ratp/r_108503/the-metro-a-parisian-institution

[Dubinin 2006] Building of reachability graph extractions using a graph rewriting system, [online]. [Accessed 15 February 2019]. Available from: https://www.researchgate.net/publication/235763013_Building_of_Reachability_Graph_Extractions_using_a_Graph_Rewriting_System.

[Sifakis 1980] Sifakis J., "Use of Petri nets for performance evaluation". Acta Cybernetica, 4 (1978):185–202, (1980)

[Hansson 1994] Hansson Hans A. and Bengt J: "A logic for reasoning about time and reliability". Formal aspects of computing, 6(5) pp:512-535, (1994).

[Berthomieu 1991] Berthomieu B. and Diaz M: "Modeling and verification of time-dependent systems using time Petri nets", IEEE transactions on software engineering, 17(3), (1991)

[Alur 1991] Alur R. and Thomas A. Henzinger. "Logics and models of real-time: A survey", In Workshop/School/Symposium of the REX Project (Research and Education in Concurrent Systems), pages 74-106. Springer, (1991).

[Chaochen 1991] Chaochen Z. et al: "A calculus of durations". Information processing letters, 40(5), pp:269-276, 1991.

[Alur 1990] Alur R. and Dill D: "Automata for modeling real-time systems". In International Colloquium on Automata, Languages, and Programming, pages 322-335. Springer, (1990)

[Alur 1993] Alur R., et al., "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems". In Hybrid systems, pages 209-229. Springer, (1993)

[Abdeddaim 2006] Abdeddaim Y., Asarin E., and Maler O: "Scheduling with timed automata". Theoretical Computer Science, 354(2), pp:272-300, (2006)

[Alur 2001] Alur R., et al: "Optimal paths in weighted timed automata". In International Workshop on Hybrid Systems: Computation and Control, pages 49–62. Springer, (2001)

[Behrmann 2001] Behrmann G., et al: “Minimum-cost reachability for priced timed automata”, In International Workshop on Hybrid Systems: Computation and Control, pages 147–161. Springer, (2001)

[Christel 2008] Christel B, Joost-Pieter K, and K G. Larsen, “Principles of model checking” MIT press, (2008)

[Selic 1998] Selic B : “Using UML for Modeling Complex Real-Time Systems”, in Proceeding LCTES '98 Proceedings of the ACM SIGPLAN Workshop on Languages, Compilers, and Tools for Embedded Systems, ISBN:3-540-65075-X (1998)

[Gogolla 2001] Gogolla M, Kobryn C: “UML 2001: the unified modeling language: modeling languages, concepts, and tools”, in Lecture Notes in Computer Science, DOI:10.1007/3-540-45441-1s, (2001)