

BOSTON UNIVERSITY  
COLLEGE OF ENGINEERING

Dissertation

**A FORMAL FRAMEWORK FOR ANALYSIS AND  
DESIGN OF SYNTHETIC GENE NETWORKS**

by

**BOYAN YORDANOV YORDANOV**

B.A., Clark University, 2005  
M.S., Boston University, 2009

Submitted in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy

2011

UMI Number: 3445758

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

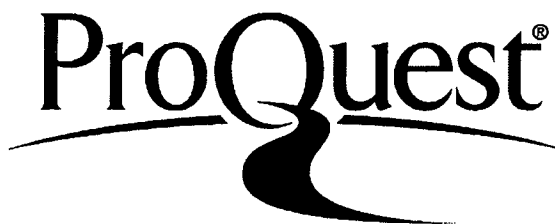
In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 3445758

Copyright 2011 by ProQuest LLC.

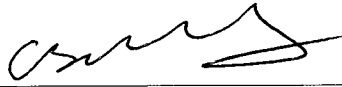
All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106-1346

Approved by

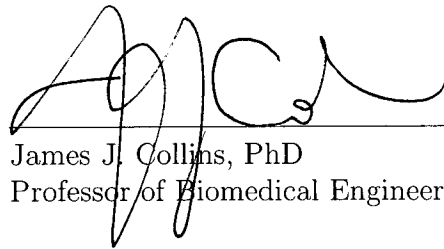
First Reader



---

Calin Belta, PhD  
Assistant Professor of Mechanical Engineering, Systems Engineering, and Bioinformatics

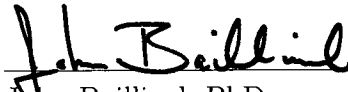
Second Reader



---

James J. Collins, PhD  
Professor of Biomedical Engineering

Third Reader



---

John Baillieul, PhD  
Professor of Mechanical Engineering, Electrical and Computer Engineering, and Manufacturing Engineering

Fourth Reader



---

Daniel Segrè, PhD  
Assistant Professor of Bioinformatics, Biology, and Biomedical Engineering

## Acknowledgments

There are quite a few people who have contributed directly towards the completion of this project. All work described in this dissertation is the result of a collaboration with my research advisor Professor Calin Belta. His patience and insightful advice helped me overcome a number of hurdles during my time under his supervision and attempting to describe how much I have learned from him should be the goal of a separate project. Some results from Chapters 3 and 4 are based on work with Gregory Batt, who was willing to listen and provide feedback during my first steps on this project. The results presented in Section 4.2, as well the results on the control of nondeterministic systems from Chapter 6, are based on work with Jana Tůmová, Ivana Černá, and Jiří Barnat. The implementation of the methods described in this dissertation uses model checking code developed by Marius Kloetzer. The experimental data used for the case studies presented in Chapter 7 was provided by Professor Ron Weiss. I thank Professor James Collins and the members of his group for sharing their lab space and resources and for their help. I also thank the members of my dissertation committee for their patience, guidance, and time.

Besides those who have directly helped this project, there are many people whose contribution has been in the form of moral support, time for discussions, and advice. Throughout this work I have heavily depended on my family to help get me through tough times and I am deeply grateful that they have always made themselves available. I thank my friends who have been around to answer a random question, provide the occasional distraction or engage in equally enjoyable though provoking and mind-numbing conversations. The countless discussions I have had with former and current members of the HyNeSs group have contributed immensely towards this project and have made both my work and lunch breaks more enjoyable.

# A FORMAL FRAMEWORK FOR ANALYSIS AND DESIGN OF SYNTHETIC GENE NETWORKS

(Order No.                                 )

**BOYAN YORDANOV YORDANOV**

Boston University, College of Engineering, 2011

Major Professor: Calin Belta, Ph.D.,  
Assistant Professor of Mechanical Engineering,  
Systems Engineering, and Bioinformatics

## ABSTRACT

Synthetic biology has recently emerged as an attempt to study biological systems by construction rather than through observation. Work in the field has demonstrated that synthetic gene networks with a specific function can be designed and constructed experimentally and is expected to lead to important application in bioremediation, biosensing, clean fuel and drug bioproduction, and therapeutics. However, designing biological systems that work as expected remains a challenge.

Mathematical modeling is often used to guide the design efforts in synthetic biology but the types of models are often complex and cannot be easily analyzed. In addition, only simple specifications such as the existence of equilibria, limit cycles, or invariance sets are usually considered. In contrast, methods for proving (or disproving) the correctness of software programs and digital circuits have been developed in the field of formal verification. Such systems can be modeled as simple finite transition graphs and algorithms for automatically deciding whether a model satisfies a specification, expressed in temporal logic, are available.

Temporal logics are rich enough to capture properties of biological systems relevant to synthetic biology. However, only simple, unrealistic models are directly amenable to formal verification. In this work, we bridge the gap and develop a theoretical framework and a set of computational tools allowing the analysis of realistic models from rich temporal logic specifications. We consider discrete time, piecewise affine (PWA) systems, which evolve along different affine dynamics in different regions of the continuous state space. This structure results in models that are globally complex and can approximate nonlinear systems with arbitrary accuracy, but are also locally simple, which allows us to construct finite abstractions. Based on this, we develop formal methods for the analysis, parameter synthesis and control of PWA systems from temporal logic specifications.

We apply our methods to analyze the synthetic gene networks that can be constructed from a set of available parts. We demonstrate how our tools can identify device designs that fail to meet the required specifications. Such an approach can be used to filter flawed designs before they are implemented experimentally, thereby decreasing the time and cost involved in synthetic biology projects.

# Contents

<b>Abstract</b>	<b>iv</b>
<b>Table of Contents</b>	<b>vi</b>
<b>List of Tables</b>	<b>ix</b>
<b>List of Figures</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Mathematical modeling of genetic regulatory networks . . . . .	6
1.2 Identification of PWA models . . . . .	9
1.3 Specification Formalisms . . . . .	11
1.4 Finite Abstractions of Infinite Systems . . . . .	15
1.5 Analysis, Parameter Synthesis, and Control of PWA Systems . . . . .	17
1.6 Tools for Synthetic Biology . . . . .	20
1.7 Contribution of the dissertation . . . . .	22
1.8 Organization of the dissertation . . . . .	26
<b>2 Mathematical Preliminaries</b>	<b>27</b>
2.1 Polytopes . . . . .	27
2.2 Operations on Polytopes . . . . .	29
2.3 Affine functions on Polytopes . . . . .	29
2.4 Transition Systems . . . . .	30
2.5 Simulation and Bisimulation Quotients . . . . .	31
2.6 Linear Temporal Logic . . . . .	33

2.7	Büchi and Rabin Automata . . . . .	36
2.8	Model Checking . . . . .	37
<b>3</b>	<b>Finite Abstractions of PWA systems</b>	<b>39</b>
3.1	PWA Systems with Uncertain Parameters . . . . .	40
3.2	Finite Quotients of PWA Systems . . . . .	42
<b>4</b>	<b>Formal Analysis of PWA systems</b>	<b>47</b>
4.1	Formal Analysis of Infinite Transition Systems . . . . .	49
4.1.1	Iterative model checking . . . . .	50
4.1.2	Quotient Refinement . . . . .	52
4.1.3	Conservatism . . . . .	55
4.2	Formula Guided Refinement . . . . .	57
4.2.1	$\phi$ -equivalence . . . . .	57
4.2.2	Constructing $\phi$ -equivalent quotients . . . . .	59
4.3	Formal Analysis of PWA Systems . . . . .	63
4.4	Complexity . . . . .	65
4.5	Implementation and Case Study . . . . .	66
<b>5</b>	<b>Parameter Synthesis for PWA systems</b>	<b>71</b>
5.1	Counterexample-guided Parameter Synthesis . . . . .	73
5.1.1	Construction of satisfying quotients . . . . .	74
5.1.2	Parameter synthesis . . . . .	75
5.2	Construction of Bisimulation quotients . . . . .	79
5.3	Implementation and Case Study . . . . .	80
<b>6</b>	<b>Formal Synthesis of Control Strategies for PWA systems</b>	<b>84</b>
6.1	Preliminaries . . . . .	85
6.2	Problem Formulation and Approach . . . . .	86



6.3	Control Transition System . . . . .	90
6.3.1	Construction . . . . .	90
6.3.2	Computation . . . . .	92
6.4	LTL Control for Finite Transition Systems as a Rabin Game . . . . .	95
6.5	LTL Control of PWA Systems . . . . .	99
6.6	Conservatism and Stuttering Behavior . . . . .	101
6.7	Complexity . . . . .	106
6.8	Implementation and Case Study . . . . .	107
<b>7</b>	<b>Applications to Synthetic Biology</b>	<b>111</b>
7.1	PWA Models of GRNs . . . . .	114
7.2	Analysis of a toggle switch and a repressilator . . . . .	117
7.3	Constructing Devices from Parts . . . . .	122
7.4	Formal Analysis of Device Models . . . . .	126
7.4.1	Cascade results . . . . .	126
7.4.2	Repressilator results . . . . .	127
7.4.3	Toggle switch results . . . . .	129
7.5	Discussion . . . . .	132
<b>8</b>	<b>Conclusions and Future Work</b>	<b>135</b>
	<b>References</b>	<b>139</b>
	<b>Curriculum Vitae</b>	<b>156</b>

# List of Tables

4.1	Parameters of the PWA system used as a case study. . . . .	67
7.1	Relative volumes and computation times of satisfying and violating regions for the toggle switch model with different amounts of parameter uncertainty. . . . .	119
7.2	Relative volumes and computation times of satisfying and violating regions for the cascade model. . . . .	127
7.3	Relative volumes and computation times of satisfying and violating regions for the <i>tetR/lacI/cI</i> repressilator model. . . . .	129
7.4	Relative volumes and computation times of satisfying and violating regions for the <i>tetR/cI/lacI</i> repressilator model. . . . .	129
7.5	Relative volumes and computation times of satisfying and violating regions for the three toggle switch models. . . . .	132

# List of Figures

4.1	An example of the application of $\text{REFINE}(T/\sim, X)$ to the quotient $T/\sim$ .	52
4.2	Formula equivalent quotient <b>(B)</b> of transition system <b>(A)</b> .	59
4.3	Simulated trajectories of the PWA system used as a case study. Initial conditions are denoted by red squares.	69
4.4	Analysis results. Regions satisfying the formula are shown in green (lighter gray), while regions satisfying the negation are shown in red (darker gray).	70
5.1	State partition for the PWA system used as a case study for parameter synthesis.	81
5.2	Parameter Synthesis Results. Transitions are represented by black arrows. Transitions removed during the execution of the algorithm are shown in red.	83
6.1	Illustration of our approach to formal synthesis of control strategies for PWA systems (Problem 4).	89
6.2	A trajectory remaining forever in state $l_2$ exists in the finite abstraction <b>(B)</b> , although such a behavior is not necessarily possible in the concrete system <b>(A)</b> .	102
6.3	Results from the formal synthesis of a control strategy for a PWA system	110

7.1	Gene regulatory function and approximations. In Figures 7.1(b) and 7.1(c) the dashed lines represent the Hill function and the solid lines its piecewise affine approximations. . . . .	115
7.2	Gene regulatory network topologies used in our case studies (square arrowheads indicate repression). . . . .	117
7.3	Numerical simulations of the deterministic and 10% parameter uncertainty toggle switch models. Trajectories in both state space and time are shown. . . . .	118
7.4	Results from the analysis of fixed parameter (deterministic), and uncertain parameter toggle switch model for specifications $\phi_1$ (eventually, the concentrations of $R_1$ and $R_2$ stabilize in a high and a low state, respectively) and $\phi_2$ (eventually the concentrations of $R_1$ and $R_2$ stabilize in a low and a high state, respectively). Satisfying regions are shown in green (lighter gray) and violating regions are shown in red (darker gray). . . . .	120
7.5	Numerical simulation of the deterministic repressilator model. . . . .	121
7.6	Results from the analysis of fixed parameter (deterministic), and uncertain parameter (1% noise) repressilator model for specification $\phi = \square(\diamond\pi_1 \wedge \diamond\pi_2)$ . The satisfaction of the specification can be guaranteed for trajectories of the system originating anywhere but the region shown in black. . . . .	122
7.7	The original transcriptional cascade used in this case study and the set of parts it is composed of. . . . .	123
7.8	A set of devices with a repressilator or a toggle switch topology, constructed from the parts from 7.7(b). . . . .	126
7.9	Numerical simulations of the transcriptional cascade model. . . . .	127

7·10 Numerical simulations of the two repressilator models. . . . .	128
7·11 Numerical simulations of the three toggle switch models. . . . .	130

## List of Abbreviations

$ A $	.....	the cardinality of a finite set $A$
$A^\omega$	.....	the set of all infinite strings over set $A$
$2^A$	.....	the power set of $A$ (the set of all its subsets)
$\mathbb{N}$	.....	the set of all Natural numbers
$\mathbb{R}$	.....	the set of all Real numbers
$\mathbb{R}^N$	.....	the $n$ -dimensional space with real coordinates
BDA	.....	Biodesign automation
CEGAR	.....	Counterexample guided abstraction refinement
CTL	.....	Computation tree logic
DES	.....	Discrete-event system
EDA	.....	Electronic design automation
$GRF$	.....	Gene regulation function
$GRN$	.....	Gene (genetic) regulatory network
LTL	.....	Linear temporal logic
ODE	.....	Ordinary differential equation
PLDE	.....	Piecewise linear differential equation
PWA	.....	Piecewise affine
SCC	.....	Strongly connected component

# Chapter 1

## Introduction

The “Central Dogma of Molecular Biology” describes the flow of information from DNA sequence to proteins and, ultimately, to biological function. Research in the field of genetic engineering has demonstrated that the phenotype of an organism can be modified by modifying its genetic code. Such biological “reprogramming” has become a valuable experimental tool, allowing organisms to be investigated through genetic perturbations. Pushing the methods of genetic engineering to the extreme, the field of synthetic biology has recently emerged as an attempt to engineer novel biological systems rather than merely study existing ones. Seminal work demonstrated that synthetic gene networks operating as a toggle switch [Gardner et al., 2000] and an oscillator [Elowitz and Leibler, 2000] can be experimentally constructed. Since then, biological systems implementing cascades, pulse generators, logic formulas, spatial patterning, and counters, have been built (see [Purnick and Weiss, 2009] for a review). More recently, a living organism containing a fully synthetic genome has been produced [Gibson et al., 2010], making the engineering of synthetic life a reasonable goal for the near future.

Synthetic biology offers the technology to test our understanding of natural systems through construction rather than observation. Mastering the design principles of nature can lead to the development of novel, drastically different devices, based on the use of “wet” biological components, and improved traditional devices inspired from biological systems. In addition, synthetic biology has the potential to affect

the human quality of life through applications including bioremediation [Cases and Lorenzo, 2005], biosensors [Antunes et al., 2006], clean fuel [Savage et al., 2008] and drug [Ro et al., 2006] bioproduction, and therapeutics [Anderson et al., 2006].

Two of the key enabling technologies for synthetic biology have been high throughput DNA sequencing and synthesis. The increasing speed and decreasing cost of DNA sequencing have resulted in the construction of databases containing an abundance of genetic information. This has provided a catalog of genetic “parts” that can be recombined to construct novel biological devices. Furthermore, it has led to a deeper understanding of how genetic sequence gives rise to biological function, what regulation mechanisms are involved in this process, and how regulation rather than genome size can be responsible for biological complexity [Levine and Tjian, 2003]. Although the cost of synthesis is still prohibitive for large DNA fragments, the methods are improving fast and hold the promise of making the implementation of a synthetic design into DNA sequence as easy as the compilation of a computer program from source code.

As technology allowing fast and cheap reading (sequencing) and writing (synthesis) of genetic code becomes increasingly available, the complexity of synthetic constructs that can be made surpasses our ability to design them manually at the level of sequence. Drawing inspiration from other engineering fields where similar challenges have been faced, standardization, decoupling, and abstraction of biological parts has been proposed as a solution [Endy, 2005]. One major effort in that direction has come from the development of the BioBricks standard [Knight, 2003] and the registry of standard biological parts [Peccoud et al., 2008]. The focus there, however, has been on defining parts in terms of their basic functionality as promoters, ribosome binding sites, and coding sequences with specifications to allow their standard assembly into more complicated devices. While this has simplified the design pro-



cess by allowing researchers to work with parts rather than sequence, constructing a biological device with a specific function remains a challenge and often involves trial and error effort. Therefore, it is not surprising that the number of newly constructed synthetic devices has steadily increased over the past years but their complexity has reached a peak [Purnick and Weiss, 2009].

In the past, a similar complexity peak has been reached and successfully overcome in the field of electronic circuit design. This has been achieved largely through the use of software tools for Electronic Design Automation (EDA), which have enabled the design of systems far more complicated than what has been possible through manual efforts. With the hope of bringing the same power to synthetic biology, frameworks for Bio-Design Automation (BDA) were recently proposed [Riedel, 2010, Densmore et al., 2009]. While current BDA platforms can make the physical assembly (*i.e.* putting parts together) of complicated biological devices easier, much of the design effort (*i.e.* which parts to use) is left to the user. Due to the time and cost involved in the process, designing devices that work as expected with minimal experimental work remains a major challenge.

Mathematical modeling is often used in order to select and tune potential device designs before they are implemented and tested experimentally. Both rational [Tuttle et al., 2005] and evolutionary [Francois and Hakim, 2004] model driven design has been successfully applied in synthetic biology. Various modeling formalisms, ranging from simple graphical representations to biologically accurate, stochastic models, have been proposed and are discussed in further detail in Section 1.1. A model that is realistic and has predictive power can guide the design efforts and lead to the construction of a device that works correctly, while saving expensive trial and error experiments. However, it is often the case that realistic models are also “complex”, making their analysis hard.

Besides selecting a model that is both realistic and analytically tractable, specifying the required behavior for devices is a separate challenge to the design efforts in synthetic biology. In principle, simple specifications such as bistability or oscillations can be captured formally using dynamical systems theory as the existence of stable equilibria or limit cycles, respectively. Although checking if such conditions are satisfied can be a non-trivial problem for dynamical models, it provides an approach for evaluating potential device designs. The problem becomes much harder if richer specifications, such as “*While the concentration of LacI remains below 50 copies per cell, there is at least 20% more of TetR than CI*”, are required.

The analysis of “complex” and realistic mathematical models from rich specifications is a challenging problem. Therefore, models are often analyzed manually using a variety of different tools and techniques in order to gain some confidence that the system under investigation behaves as expected. Providing formal guarantees of correctness is often beyond the power of currently available methods. In addition, such approaches are applicable only for simple specifications such as stability of equilibria or set invariance.

In contrast, methods for proving (or disproving) the correctness of software programs and digital circuits have been developed in the field of formal verification. The behavior of such systems is captured by finite transition graphs and automata models, specifications are expressed as rich temporal logic formulas, and off-the-shelf algorithms for deciding whether a model satisfies a specification are available [Clarke et al., 1999]. Temporal logic formulas are rich enough to capture properties of biological systems relevant to synthetic biology [Monteiro et al., 2008]. However, due to the simplicity of the models amenable to formal verification, such methods can be applied directly only to simple, qualitative models.

In this work, we focus on the design of (synthetic) genetic regulatory networks

(GRNs), which remains a major research direction in the field of synthetic biology. We bridge the gap between techniques from dynamical system theory, traditionally used to manually analyze “complex” models of GRNs for “simple” specifications, and formal verification methods that can automatically check the correctness of “simple” models against “complex” specifications. We develop a theoretical framework and a set of computational tools, based on the use of hybrid systems, which provide a realistic modeling formalisms for GRNs. Hybrid systems capture both the continuous dynamics of the various species (mRNAs, proteins, etc), as they are produced and degraded at certain rates, and the discrete transitions of the system between different regulation modes (*e.g.* basal versus full expression from a promoter). For a particular class of hybrid systems, we show that finite abstractions can be computed, which allows us to develop methods similar to the ones used in formal verification.

In our framework, the analysis of realistic hybrid system models of synthetic GRNs can be performed automatically, even when rich specifications are considered, while the correctness of the results is guaranteed for the model. This can simplify and speed up the construction of synthetic GRNs that work correctly, by allowing researchers to filter flawed designs through model analysis, rather than experiments. While our motivation comes from synthetic biology, the methods we develop are quite general and have the potential to impact other fields of engineering, where similar models are used.

In the following sections, we review related literature and motivate our choice of piecewise affine (PWA) systems as models of genetic regulatory networks and linear temporal logic (LTL) as a specification language. We outline the main contributions of this work in Section 1.7 and discuss both the theoretical implications of the developed methods and the practical significance of their implementation as tools for synthetic biology.

## 1.1 Mathematical modeling of genetic regulatory networks

In recent years, several formalisms for modeling genetic regulatory networks (GRNs) have been proposed (see [de Jong, 2002] for a review). They can be divided into deterministic and stochastic, continuous and discrete time, or finite and infinite (continuous) state. The most abstract models simply represent a regulatory network schematically, with symbols signifying genetic elements (promoters, RBSs, coding sequences), mRNAs, proteins, and the interactions between them. Although such models can provide an intuitive description of simple networks, they carry very limited information about the system and in this work are used only as visual aid. On the other extreme, stochastic models serve as an accurate description of biological systems, even when only a small number of molecules are involved. Such models are usually related to the Master Equation [Gillespie, 1977, Gillespie and Mangel, 1981] and are often analyzed through numerical simulation, which can be slow. In this work, we focus instead on deterministic models, where formal analysis methods can be developed.

Under the assumption of a large number of molecules present in the system, a deterministic model can be derived as an average of a stochastic one. If spatial dynamics are considered, such formalisms can lead to systems of partial differential equations. If a well-mixed environment is modeled, ordinary differential or difference equations are commonly used to describe the evolution of species' concentrations in continuous or discrete time, respectively. Following from chemical Michaelis-Menten kinetics, under the assumption that transcription factor binding and unbinding proceeds much faster than transcription and translation [McAdams and Shapiro, 1995], transcription regulation is modeled by Hill functions [Hill, 1913], which usually results in highly nonlinear systems. Then, as for the stochastic models, analysis is often limited to numerical simulations, which can sometimes lead to erroneous results and,

in general, does not provide any guarantees for correct behavior of the system.

The use of approximate models provides one way around the complexity of stochastic and nonlinear differential equation models and leads to formalisms allowing a more formal analysis. Approximating the nonlinear Hill regulation function by a discontinuous step function leads to decoupled piecewise linear differential equation (PLDE) models, which have been investigated in [Glass, 1975, Mestl et al., 1995, de Jong et al., 2003b, Ghosh and Tomlin, 2004]. Intuitively, in the PLDE framework the regulation function relating the transcription rate of a promoter to the concentration of transcription factors is quantized. This approximation is motivated by the switching behavior of many regulatory networks and the overall system transitions between finitely many regulation modes, where different dynamics (transcription rates) describe the evolution of the concentrations of the species. Other hybrid models with simplified continuous dynamics and discrete events are considered in [Alur et al., 2001, Belta et al., 2004, Belta et al., 2001, Belta et al., 2005].

As a more drastic idealization of GRNs, each gene can be described as being active or inactive by a single Boolean variable [Kauffmann, 1969]. Boolean regulation functions provide a deterministic update rule and the system evolves in discrete time. Such models have a finite number of states and can be conveniently analyzed even for very large networks but do not capture enough quantitative information of the system. Boolean networks have been further extended as generalized logical networks [Thomas, 1991] by allowing variables to have multiple states and transitions to occur asynchronously. Qualitative differential equations [Kuipers, 1981] provide an additional formalism for combining logical and continuous aspects. Rule-based formalisms [Brutlag et al., 1991, Blinov et al., 2004] use an even higher level of abstraction, such as graphs and graph rewrite rules, in order to model biological systems where a combinatorial number of states makes other methods intractable.

The methods outlined above are amenable to analysis but they are often based on oversimplifications and, in general, do not capture enough information to allow the design of synthetic GRNs. Some are purely discrete [Fages et al., 2004, Eker et al., 2002], or capture protein dynamics but cannot accommodate other chemical reactions [Kauffmann, 1969, Ghosh and Tomlin, 2004, de Jong et al., 2003b]. The more formal and expressive ones are based on previously defined specification formalisms such as CTL [Fages et al., 2004, Batt et al., 2005] and LTL [Eker et al., 2002] and inherit pre-existing methods and tools but can lead to complicated translations.

In this work, we use an approach based on hybrid systems [Alur et al., 1996, Henzinger and Sastry, 1998, Vaandrager and van Schuppen, 1999, Lynch and Krogh, 2000], where continuous dynamics are combined with discrete events. Specifically, to model genetic regulatory networks we use discrete time, piecewise affine (PWA) systems [Sontag, 1981]. Intuitively, PWA systems evolve in discrete time along different affine dynamics in different regions of the continuous state space. The dynamics can accommodate both chemical reactions and production and degradation of species such as mRNAs and proteins, while regulation is captured through the partition of the state space and arises naturally by considering piecewise affine regulation functions. The different regions in state correspond to different levels of activation of genes. As an extreme approximation, the gene regulation function can be captured by a step function, leading to a state partition with only two regions. The gene is expressed at a maximal rate in one of the regions and at a basal rate in the other. A more accurate approximation can be achieved by using a ramp function, which induces an intermediate regulation region, where expression is graded. By making the state partition finer, better approximations of the regulation functions can be achieved. Therefore, PWA systems can approximate nonlinear dynamics with arbitrary accuracy [Lin and Unbehauen, 1992]. Even though the local dynamics in each

mode are simple, the global behavior of the system can be complex, so the richness of the model (in terms of the class of allowed nonlinearities) is not sacrificed. However, the locally affine structure allows us to develop methods for the computation of finite abstractions of PWA systems (see subsection 1.4), which enables the formal analysis of such systems from temporal logic specification (see subsection 1.3).

Under mild additional assumptions, PWA systems are equivalent to several other classes of hybrid systems, including mixed logical dynamical (MLD), linear complementarity (LC), extended linear complementarity (ELC), and maxmin-plus-scaling (MMPS) systems [Heemels et al., 2001]. In particular, MLD systems are frequently used in practice and the conversion from MLD to PWA can be efficiently performed using a mode enumeration algorithm [Geyer et al., 2003].

The compromise between richness and analyzability motivates our choice of PWA systems with uncertain parameters as a realistic modeling formalism for synthetic gene networks.

## 1.2 Identification of PWA models

PWA models of GRNs can be obtained using several approaches. There exist computationally attractive techniques for the identification of such systems, which include Bayesian methods [Juloski et al., 2005], bounded-error procedures [Bemporad et al., 2003], clustering-based methods [Ferrari-Trecate et al., 2003], Mixed-Integer Programming [Roll et al., 2004], and algebraic geometric methods [Vidal et al., 2003]. Once a synthetic GRN has been constructed, input-output experimental data can be collected by varying the levels of external inducers (input) and measuring the concentrations of proteins or mRNA (output). Protein concentrations can be obtained using quantitative Western blot techniques, or more commonly by fluorometer, cytometer, or fluorescence microscopy measurements of the fluorescence of certain re-

porter proteins, in which case statistical and spatial information is also acquired. Concentrations of mRNA are often measured using quantitative Northern blot, Real time PCR, or microarray techniques.

A second approach to the construction of GRN models involves using quantitative information of the different parts (promoters, ribosome binding sites, proteins, mRNA) involved in the system in order to construct a composite PWA model that can be further analyzed. Predicting the behavior of genetic circuits from characterization information of regulatory elements has been successfully demonstrated in experiments [Rosenfeld et al., 2007], although issues of modularity and retroactivity [Del Vecchio et al., 2008] present a challenge to such compositional modeling. Kits for the characterization of promoters and ribosome binding sites have been recently proposed [Kelly et al., 2009], where all measurements are made against a reference standard in order to decrease variability. Prototypes of data sheet for biological parts, similar to the ones used for electronic components, have also been discussed [Canton et al., 2008] and databases of characterized parts are being developed [Biofab, 2009]. Even when parts are not well characterized, predictions of the degradation rates of proteins [Gasteiger et al., 2005], or the rates of translation initiation [Salis et al., 2009] and elongation [Zhang and Ignatova, 2009] can be made from sequence information.

Measurements of the gene regulation function (GRF), which relates the concentrations of a transcription factor to the transcription rate from a regulated promoter, are perhaps the most challenging to obtain. Indirect measurements can be acquired by varying the concentrations of external inducers and measuring downstream effects [Hooshangi et al., 2005]. A technique based on fluorescence reporter genes and the construction of fusion proteins allows gene regulation functions to be measured with single-cell resolution [Rosenfeld et al., 2005]. Experimental data collected using



this technique has suggested that the GRF fluctuates dynamically and limits the accuracy of information transfer that can be achieved in synthetic gene circuits. In order to capture such limitations and make analysis efforts relevant, the uncertainty inherent in biological systems must be represented in the models.

We consider PWA systems where parameters are not precisely known but allowed to vary in certain polyhedral ranges. By using this approach, very general GRFs can be captured, while methods for automatic and formal analysis can still be developed. In addition, our control methods can tolerate uncertainty in both state measurements and applied inputs. This accounts for the fact that even when the concentrations of an external inducer are controlled, they cannot be set precisely. We assume that no statistics of the uncertainty in a system are available and develop a nondeterministic rather than a probabilistic framework. While this makes the methods conservative, it is more general than assuming particular (normal) distributions.

### 1.3 Specification Formalisms

As in our choice of a modeling formalism, we have to compromise between expressivity and analyzability when choosing a specification language. Specifications considered as part of dynamical systems theory are qualitative and usually limited to expressing the existence of equilibria or oscillations. Specifications considered in optimization and classical control theory, on the other hand, are too quantitative and require precise mathematical conditions to be satisfied. Besides expressivity, the difficulty of formulating specification is another drawback of such methods.

In this work, we focus on temporal logics [Emerson, 1990, Clarke et al., 1999] as a specification formalism. Temporal logics were originally developed for specifying the correctness of digital circuits and computer programs. Due to their expressivity, resemblance to natural language, and the availability of algorithms for deciding if

a formula is satisfied [Clarke et al., 1999, Holzmann, 2004], temporal logics have gained popularity in many other areas. Analysis of systems with continuous dynamics based on qualitative simulations and temporal logic was proposed in [Shults and Kuipers, 1997, Brajnik and Clancy, 1998, Davoren et al., 2004]. Control of linear systems from temporal logic specifications has been considered in both discrete time [Tabuada and Pappas, 2003] and continuous time [Kloetzer and Belta, 2006a]. The use of temporal logic for task specification and controller synthesis in mobile robotics has been advocated as far back as [Antoniotti and Mishra, 1995], and more recent results include [Loizou and Kyriakopoulos, 2004, Quottrup et al., 2004, Fainekos et al., 2005, Kloetzer and Belta, 2006b]. In the area of systems biology, the qualitative behavior of genetic circuits can be expressed in temporal logic, and model checking can be used for analysis, as suggested in [Antoniotti et al., 2003, Batt et al., 2005, Batt et al., 2008]. Temporal logics are also used to specify properties of biomolecular networks in [Bernot et al., 2004, Chabrier-Rivier et al., 2004, Eker et al., 2002], where the aim is to check whether the system satisfies properties for given initial conditions.

Linear Temporal Logic (LTL), Computation Tree Logic (CTL) and their unifying CTL\* framework [Emerson, 1990, Clarke et al., 1999] are the most commonly used temporal logics. For this project, the computational expense involved in model checking CTL\* outweighs the gains in expressivity and therefore this logic is not considered. Both LTL and CTL have been used to describe properties of biological systems. The two logics are incomparable, in the sense that there exist LTL formulas that cannot be expressed in CTL and vice versa. CTL is a branching time logic and allows the quantification of specifications over the executions of the system. For example, in CTL a property can be satisfied by the system if it is satisfied over all paths or if there exists a path that satisfies it, and the distinction between the two is specified through additional path quantifiers. When expressing properties of

biological systems, CTL allows testing whether it is possible for the system to have some behavior, which can be useful when studying natural systems. The focus of this dissertation is on the design of synthetic gene networks, where requiring that the system always behaves a certain way seems more appropriate. In addition, in CTL each temporal operator must always be preceded by a path quantifier. This can make the translation of a specification from natural language to a formula prone to errors [Schlipf et al., 1997, Memon, 2003], as one must think about all possible executions of the system at the same time. Expressing specifications in LTL is more natural because executions are considered one at a time and we use this logic for the project. It might seem that CTL has the advantage of computationally cheaper, polynomial time model checking procedures over LTL, where algorithms are exponential time. However, based on empirical results, it has been suggested that for formulas expressible in both CTL and LTL model checkers perform similarly [Vardi, 2001], due to the fact that formulas in CTL can be larger than their equivalent LTL representation.

Other temporal logic formalisms have been developed to deal with continuous time systems (as in the timed automata model [Alur et al., 1990]) or probabilistic systems [Vardi, 1999], where the goal is to guarantee that the specification holds with certain probability. Since we consider discrete time, nondeterministic PWA systems with uncertain parameters, such extension are not required. A different extension of temporal logics allows a continuous degree of satisfaction to be established for formulas [Rizk et al., 2008], and applications of such methods to systems biology have been considered. While this can lead to interesting combination of heuristic search strategies and model checking, the methods are not directly related to the topics discussed here.

A rich spectrum of properties of gene networks are naturally expressed as LTL

formulas over linear predicates in the state variables, which represent the concentrations of different species in the system. In the most basic case, a linear predicate can specify a threshold in the concentrations of a protein or mRNA (*e.g. the concentration of LacI is above 30 copies per cell*). Multiple predicates can be used to specify ranges (*e.g. the concentration of LacI is between 30 and 50 copies per cell*). A linear predicate can also be used to relate concentrations of different species (*e.g. there is more of LacI than CI*). Once such predicates are formulated, they can be used to construct temporal logic formulas, allowing very rich specifications to be captured. Some examples of formulas useful for expressing biological properties include reachability (*e.g. the concentration of LacI eventually reaches values above 30 copies per cell*), safety (*the concentration of LacI never reaches values above 30 copies per cell*) and invariance (*the concentration of LacI always remains between 30 and 50 copies per cell*). Additional formula classes that capture interesting properties can express consequence (*if the concentration of LacI goes above 30 copies per cell then (eventually) the concentration of CI drops below 50 copies per cell*) and sequence properties (*eventually, the concentration of CI drops below 50 copies per cell, which is preceded by an increase in the concentration of LacI to levels above 30 copies per cell*). Furthermore, using the discrete nature of the systems we consider and the temporal LTL operators, specifications regarding absolute time can be formulated (*concentration of LacI goes above 30 copies per cell within 20 time steps after the concentration of CI drops below 50 copies per cell*). To make the formulation of specifications easier, the language can be restricted to sets of commonly used patterns expressible in temporal logic, as suggested in [Monteiro et al., 2008].

## 1.4 Finite Abstractions of Infinite Systems

In this work, we develop a theoretical framework for the specification and automatic verification of properties of PWA system from rich specifications. Our approach is based on the idea of abstraction, which allows an infinite state system, such as a PWA system, to be mapped to a finite state system, such as a transition system (see [Alur et al., 2000] for an earlier review of literature focused on the construction of finite abstractions of infinite systems). Intuitively, the states of the abstract model represent infinite sets of equivalent states in the concrete model. Analysis of the infinite PWA system can then be performed instead on its abstract model, which can either be equivalent with respect to the satisfaction of a specification, or provide an approximation with the guarantee that satisfaction in the abstract model implies satisfaction in the original system. Sufficient abstractions are constructed using simulation relations [Clarke et al., 1999] and equivalent abstractions are traditionally based on the notion of bisimulation [Milner, 1989]. The idea of constructing specification dependent equivalences, coarser than bisimulation, for finite system has been suggested in [Aziz et al., 2002]. In this work, we explore the construction of abstractions, which are equivalent only with respect to a given specification, for infinite systems.

It has been previously established that equivalent finite models do not exist even for some very simple hybrid systems [Henzinger et al., 1998]. However, classes of systems for which equivalent finite models exist can be identified by restricting the continuous behavior of the hybrid system, as in the case of timed automata [Alur and Dill, 1994], multirate automata [Alur et al., 1993], [Nicollin et al., 1993], and rectangular automata [Henzinger et al., 1998, Puri and Varaiya, 1994], or by restricting the discrete behavior, as in order-minimal hybrid systems [Lafferriere et al., 2000, Lafferriere et al., 1999a, Lafferriere et al., 1999b]. In more recent work, the existence of

equivalent abstractions for discrete-time, continuous-space linear systems has been shown [Tabuada and Pappas, 2006]. For more general systems, the relaxed notion of approximate bisimulation [Tabuada, 2006, Girard, 2007] can provide a measure of how accurate a finite abstraction is with respect to the original model. All system classes, for which the existence of equivalent abstractions has been established, are too weak to accurately represent the nonlinearities involved in gene regulation

We follow an approach similar to [Pappas, 2003, Tabuada and Pappas, 2003] and embed discrete-time systems into transition systems but we consider the more general class of PWA rather than linear systems. Since we cannot guarantee the existence of equivalent finite abstractions (bisimulation quotients) for this class of systems, we focus on the computation of sufficient abstractions (simulation quotients). In general, this leads to methods that are provably correct but conservative (*i.e.* we cannot guarantee finding a solution even when one exists, but if a solution is found, it is guaranteed to be correct). Other works related to the construction of simulation quotients has focused on partitioning using linear functions of the continuous variables, as in the method of predicate abstractions [Alur et al., 2002, Tiwari and Khanna, 2002], or using polynomial functions as in [Tiwari and Khanna, 2002, Ghosh et al., 2003]. However, the construction of simulation or bisimulation quotients (when they exist) using such methods is computationally expensive and involves the integration of the vector fields of the original system [Alur et al., 2002] or quantifier elimination for real closed fields and theorem proving [Tiwari and Khanna, 2002]. For continuous time piecewise affine systems with simplicial partitions and continuous time multi-affine systems with rectangular partitions, checking the existence of bisimulation quotients and constructing simulation quotients can be reduced to polyhedral operations only [Belta and Habetz, 2004].

## 1.5 Analysis, Parameter Synthesis, and Control of PWA Systems

Due to their ability to capture complex nonlinear dynamics and the existence of finite abstraction that can be constructed through polyhedral operations, we use PWA systems to model genetic networks. Because of their expressivity, resemblance to natural language, and the existence of model-checking algorithms we formalize specifications as LTL formulas. In the rest of this dissertation, we develop methods for the analysis, parameter synthesis and control of PWA systems from specifications expressed as LTL formulas, and apply our methods to the analysis of synthetic gene networks.

Analysis has been previously considered in literature focused on controlling PWA systems [Bemporad et al., 2000, Grieder, 2004], but only for very simple properties such as invariance and reachability. By considering arbitrary LTL specifications, we significantly expand this class of properties (invariance and reachability are particular examples of LTL properties). We use the concept of embedding an infinite PWA system into a transition system in order to reason about the satisfaction of temporal logic formulas by its trajectories. Our analysis method is an extension of model checking [Clarke et al., 1999] and bisimulation based refinement [Bouajjani et al., 1991], where both procedures are computed together as part of an iterative loop. In that respect, our work is related to [Chutinan and Krogh, 2001], where the verification of infinite state systems using approximation quotients is proposed. Since systems with continuous dynamics are considered, transitions in the quotient are computed using flow-pipe approximations [Chutinan and Krogh, 1998]. In this work, we consider discrete time dynamics and compute exact quotients, even when the PWA system is subjected to additive noise. In addition, in [Chutinan and Krogh, 2001] the goal is to verify the system and computation is terminated as soon as vio-

lating behavior is observed. Instead, we attempt to find the largest satisfying region (a region of initial conditions from which all trajectories satisfy the specification) and search for a similar violating region, which provides more quantitative results for analysis and leads to computational advantages. In addition, we characterize quotients coarser than bisimulation but equivalent to the original system with respect to a specification and develop algorithms for the construction of such quotients. In that respect, our approach is related to [Aziz et al., 2002], where similar quotients for finite systems are discussed.

Our strategy for parameter synthesis for PWA systems involves identifying violating behavior by model checking simulation quotients, and removing it by restricting system parameters. In that respect, our method resembles “debugging” and is related to literature focused on counterexample guided refinement (CEGAR) [Clarke et al., 2003]. Unlike classical CEGAR approaches, where violating trajectories of a quotient are checked against a concrete model and, if spurious, removed by refinement, we use counterexamples to find and remove sets of (possibly spurious) violating transitions from the quotient and restrict the parameters of the systems based on the transitions that were removed. Our approach is similar to the method proposed in [Frehse et al., 2008], where parameters for linear hybrid automata are synthesized through the use of counterexamples. The methods developed in both [Frehse et al., 2008] and this work are conservative and, in general, under-approximations of the satisfying parameters are obtained.

Designing (optimal) control strategies for PWA systems has been previously considered [Bemporad et al., 2000, Grieder, 2004] and efficient software tools are available [Kvasnica et al., 2004]. However, for such approaches specifications are expressed as polytopic and logical constraints, which makes them restrictive and less intuitive to use. Our method is also related to literature focused on controlling Mixed Logical



Dynamical (MLD) systems from LTL specifications, due to the equivalence between MLD and PWA systems [Heemels et al., 2001] (see Section 1.1). This problem has been considered in [Karaman et al., 2008], where a solution involving the translation of LTL specifications into mixed-integer linear constraints is proposed but a finite horizon assumption is imposed. Instead, we consider infinite executions of the system and our approach is orthogonal, since it relies on the construction of finite quotients. Decreasing the conservatism of our methods by identifying stuttering behavior in the abstraction (*i.e.* trajectories, which can only remain in a state for a finite number of steps) is related to [Batt et al., 2007a], where time-convergent trajectories were identified.

By constructing control abstractions of PWA systems, we reduce the problem of controlling an infinite transition system to generating a control strategy for a finite transition system from a temporal logic specification. Similar problems have been considered in literature focused on controlling discrete-event systems (DES) from specifications given as CTL\* [Jiang and Kumar, 2006] or  $\mu$ -calculus [Basu and Kumar, 2006] formulas. The problem of controlling a deterministic DES from specifications given as an LTL formula is considered as a particular case in [Jiang and Kumar, 2006] (LTL is a subset of CTL\*). Although similar, our approach to controlling nondeterministic transition systems can handle information about the stuttering behavior that arises during the construction of finite quotients. The problem of controlling a finite systems from LTL specifications has also been considered in [Kloetzer and Belta, 2008b] for deterministic systems and the proposed solution has been extended to nondeterministic systems in [Kloetzer and Belta, 2008a], but completeness is guaranteed only for the LTL fragment generated by deterministic Büchi automata.

## 1.6 Tools for Synthetic Biology

The framework developed in this dissertation leads to the construction of computational tools for the analysis of models of synthetic gene networks from temporal logic specifications. Therefore, it can be seen in the context of the many other software tools developed for the field of synthetic biology (<http://sbml.org/> currently lists 188 tools). One class of tools is based on the analysis and manipulation of sequence information, including DNA [Cai et al., 2007, Villalobos et al., 2006], RNA [Zuker, 2003], and protein [Fiser et al., 2000]. While not directly related to the methods developed as part of this work, where the focus is on the analysis and design of mathematical models, such tools can be used to predict model parameters. Some examples include tools for predicting protein degradation rates [Gasteiger et al., 2005] and rates of translation initiation [Salis et al., 2009] and elongation [Zhang and Ignatova, 2009] from sequence information.

A number of design tools with various functionality have also recently been developed to facilitate the effort of designing synthetic gene networks, by allowing the user to work with a set of standardized parts (promoters, ribosome binding sites, coding sequences, terminators). In general, such tools provide a graphical interface, where separate parts can be obtained from a database, linked together into devices, and translated into DNA sequence. Examples include BioJade [Goler, 2004], TinkerCell [Chandran et al., 2009], BioTapestry [Longabaugh et al., 2005], CellDesigner [Funahashi et al., 2003], and BioNetCAD [Rialle et al., 2010]. In some cases, such tools allow mathematical models (generally, in the form of ODEs) to be obtained once a device has been constructed *in silico*. Then, trajectories of this model can be simulated using built-in or external, deterministic or stochastic algorithms. While this provides a way to analyze and validate a design, simulation alone does not lead to correctness guarantees and exhaustively simulating larger dimensional

systems is infeasible. In addition, model parameters are usually not available and must be supplied by the user.

The BioBricks standard [Knight, 2003] and the registry of standard biological parts [Peccoud et al., 2008] were developed in an effort to introduce a modular design approach to synthetic biology. In a similar effort, the computational community has developed standards for specifying biological models [Le Novère et al., 2005] and model description languages including the SBML [Hucka et al., 2003] and CellML [Hedley et al., 2001] formats. This standardization has in turn led to the development of online repositories, collecting models of biological parts and systems. Currently, the BioModels Database [Le Novère et al., 2006] lists 249 curated and 224 non-curated models in both SBML and CellML formats and provides tools for the conversion between the two formats. The models can be simulated online through the JWS Online tool [Olivier and Snoep, 2004]. The CellML Model Repository [Lloyd et al., 2008, Cooling et al., 2010] lists about 480 models and model parts and provides a set of visualization, annotation, simulation, and validation<sup>1</sup> tools. Such databases are related to this project because PWA systems can be used as an approximation of other models, including the most commonly used ODE formalism, and therefore, available models can be recast into our framework and analyzed using the richer, temporal logic specification language it provides. In our case study, we use the idea of constructing device models from available information about the parts they are composed of. While this is related to an approach taken in [Rodrigo et al., 2007], here we specifically construct PWA models that can be analyzed formally in our framework.

Tools implementing more formal approaches to the verification and tuning of biological devices are also available. The **Biochemical Abstract Machine (BioCham)**

---

<sup>1</sup>here, models are validated in terms of well-posedness and not based on the satisfaction of a functional specification as in our approach.

[Fages et al., 2004] uses a rule-based modeling formalism compatible with SBML and includes several simulators for such system. It allows properties to be specified as temporal logic formulas and can validate the models or search for parameters based on the specifications. **Simpathica/XSSYS** [Antoniotti et al., 2003] can construct models that can be analyzed formally from observed traces. It can generate ODE representations and includes a temporal logic analysis tool with a “Natural Language Interrogation” interface, for specification formulation. The **Genetic Network Analyzer (GNA)** [de Jong et al., 2003a] constructs qualitative models of genetic regulatory networks as piecewise-linear differential equations (PLDEs), which can be specified using inequality constraints, rather than exact numerical values. A finite transition system can be generated from such information and simulated or analyzed using temporal logic specifications. **RoVerGeNe** [Batt et al., 2007b] uses piecewise-multiaffine differential equation models of synthetic gene networks and can test whether a temporal logic specification is satisfied over entire ranges of parameters. In addition, it can search for parameter sets for which a given property holds.

## 1.7 Contribution of the dissertation

The biggest theoretical contribution of this dissertation is to show that for discrete time piecewise affine (PWA) systems, a class of infinite hybrid systems, finite abstractions can be constructed and refined using computationally efficient procedures. This expands the known classes of hybrid systems for which such results exist (see Section 1.4). The computability of finite quotients leads to the development of methods for analysis, parameter synthesis and control of PWA systems from temporal logic specifications. In terms of applications, this allows us to develop a framework where properties of synthetic gene networks are captured formally, using rich specifications resembling natural language, and system models can be automatically analyzed or

tuned.

Our computability results allow us to extend efficient formal verification methods, developed for finite transition systems to the class of discrete time PWA systems. Those results are quite general for two main reasons. First, PWA systems can approximate nonlinear dynamics with arbitrary accuracy [Lin and Unbehauen, 1992] and our approach can be used for deciding whether an arbitrarily fine approximation of a general nonlinear system satisfies a temporal logic formula. Second, as discussed in Section 1.1, PWA systems are equivalent with several other classes of hybrid systems [Heemels et al., 2001] and the results we establish can be extended to those equivalent classes.

We show that for discrete time PWA systems, simulation quotients can be constructed efficiently using polyhedral operations only [Yordanov et al., 2007]. In addition, we show that quotients can be constructed efficiently, even when the parameters of the system are uncertain [Yordanov and Belta, 2008a]. For cases when only additive noise is considered, this results in the computation of exact quotients and, when all system parameters are uncertain, (conservative) over-approximation quotients are constructed. Our results on the construction of finite quotients of PWA systems are presented in Chapter 3.

We show that for PWA systems, all steps of the classical “bisimulation algorithm” [Bouajjani et al., 1991], a procedure for the construction of the coarsest bisimulation relation, are computable using polyhedral operations [Yordanov et al., 2007], even when there is additive noise in the system [Yordanov and Belta, 2008a]. We develop methods for the construction of abstractions, coarser than bisimulation, that can be used equivalently instead of the original (infinite) system to decide the satisfaction of a specification [Yordanov et al., 2010]. By combining iterative refinement and model checking, we are able to develop efficient computational procedures for the analysis

of PWA systems from temporal logic specifications [Yordanov and Belta, 2010]. Our results on the analysis of PWA systems are presented in Chapter 4.

We use a counterexample guided procedure to synthesize parameters for PWA systems. Due to the use of abstractions, our approach is conservative. Unlike other approaches reviewed in Section 1.5, we construct a search tree which guarantees the completeness of the finite problem (finding a subset of transitions for a finite system, such that a temporal logic formula is satisfied) [Yordanov and Belta, 2008b]. In addition, we show that by restricting parameters of the system to appropriate subsets, bisimulation quotients can be constructed directly. Our results on the synthesis of parameters of PWA systems are presented in Chapter 5.

By constructing finite abstractions, we develop symbolic control strategies for PWA systems and use LTL formulas as specifications. Compared to other methods for control of PWA systems reviewed in Section 1.5, this significantly extends the expressivity and, due to the resemblance of LTL to natural language, allows more intuitive specifications to be used. In addition, by generating the control strategy on the abstract (symbolic) model instead of the concrete one, we can guarantee its robustness in the sense that bounded state measurement errors or noise in the applied input do not affect its correctness (*i.e.* the satisfaction of the specification is still guaranteed).

We reduce the problem of controlling an infinite PWA system to the control of finite transition system through the construction of finite abstractions. Our initial solution [Yordanov and Belta, 2009] was based on previous results from [Kloetzer and Belta, 2008a], where completeness was guaranteed only for the LTL fragment generated by deterministic Büchi automata. As part of this dissertation, we develop a general and fully automatic framework for controlling finite nondeterministic transition systems from specifications given as arbitrary LTL formulas, where the com-

pleteness of the solution is guaranteed [Tůmová et al., 2010]. This extends the results from [Kloetzer and Belta, 2008a] and increases the expressivity of the specification language significantly. By dealing with the stuttering behavior inherent in the finite abstraction, we also reduce the conservativeness of the approach from [Yordanov and Belta, 2009]. By seamlessly combining the abstraction and control procedures into a computational framework, we allow for fully automatic generation of robust PWA feedback control strategies from rich, high-level LTL specifications. Our results on the control of PWA systems are presented in Chapter 6.

The tools developed as part of this dissertation (CONPAS, FAPAS, and PARSYPAS, available at <http://hyness.bu.edu/Software>) offer several advantages over the previously available ones reviewed in Section 1.6. Our framework allows high level specifications to be expressed as rich temporal logic formulas. We use PWA systems as a modeling formalism which, as already discussed, is more general than PLDEs and rule-based models. Although piecewise multi-affine systems are more general than PWA systems, formal analysis is restricted only to cases when the state space partitioning is hyper-rectangular [Belta and Habetz, 2004]. While initial state partitions for genetic networks generally satisfy this requirement (the partition is given by thresholds in the concentrations of the species), specifications given over linear predicates involving ratios of species or partition refinement destroy that structure.

By combining refinement, analysis, parameter synthesis, and control procedures we are able to develop an unified framework for studying complex PWA models from rich temporal logic specifications. This leads to the development of tools that can automatically analyze realistic models of synthetic gene networks from high level specifications. In Chapter 7, we suggest one possible application of our analysis methods and study a number of synthetic gene networks constructed from a set of available parts. Such approaches can be used to automatically explore large design

spaces and select only a few candidate devices to be implemented and tested further. Perhaps, our tools can find their place in a unified software framework such as SBW [Bergmann and Sauro, 2006] or CLOTHO [Densmore et al., 2009], where heterogeneous applications and online repositories exchange information, and analysis results can be propagated all the way down to lab management systems for automatic device construction through liquid handling robots.

## 1.8 Organization of the dissertation

The remainder of the dissertation is organized as follows. In Chapter 2, we review some mathematical preliminaries used throughout this dissertation. In Chapter 3, we embed PWA systems as infinite transition systems and discuss the construction of finite quotients. In Chapter 4, we develop a procedure for the analysis of infinite transition systems, such as the embeddings of PWA systems. In Chapter 5, we discuss a procedure for the synthesis of parameters for PWA systems, resulting in the satisfaction of a specification or the construction of bisimulation quotients. In Chapter 6, we extend our methods to a control framework and discuss the construction of control abstractions and the computation of control strategies for finite, nondeterministic transition systems. In Chapter 7, we present results from the application of our analysis procedure to a synthetic biology problem, where bistable and oscillating systems are designed. Concluding remarks and directions of future work are presented in Chapter 8.



## Chapter 2

# Mathematical Preliminaries

In this chapter we outline some mathematical preliminaries that are used throughout the rest of this dissertation.

### 2.1 Polytopes

Given a set  $C$ , we use  $|C|$  to denote the cardinality (*i.e.* number of elements) of  $C$ . We use  $2^C$  to denote the powerset (*i.e.* set of all subsets) of  $C$ . Let  $N \in \mathbb{N}$  and consider the  $N$  - dimensional Euclidean space  $\mathbb{R}^N$ .

**Definition 1** (Convex Set). *A set  $C \subset \mathbb{R}^N$  is convex if the line segment between any two points in  $C$  lies in  $C$ . In other words, for all  $x_1, x_2 \in C$  and  $0 \leq \lambda \leq 1$  we have  $\lambda x_1 + (1 - \lambda)x_2 \in C$ .*

**Definition 2** (Convex Combination). *A point  $x = \sum_{i=1}^n \lambda_i x_i$ , where  $\sum_{i=1}^n \lambda_i = 1$  and  $\lambda_i \geq 0$  for all  $i = 1, \dots, n$  is a convex combination of points  $x_1, \dots, x_n$ .*

Similarly, a point  $x = \sum_{i=1}^n \lambda_i x_i$ , where  $\sum_{i=1}^n \lambda_i = 1$  is an *affine combination* of points  $x_1, \dots, x_n$ . Points  $x_1, \dots, x_n$  are called *affinely independent* if there does not exist an  $1 \leq i \leq n$  such that point  $x_i$  is an affine combination of points  $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n$ .

**Definition 3** (Convex Hull). *The convex hull of a set  $C$ , denoted as  $\text{hull}(C)$ , is the set of all convex combinations from points in  $C$ :*

$$\text{hull}(C) = \{x \in \mathbb{R}^N \mid x = \sum_{i=1}^n \lambda_i x_i, \lambda_i \geq 0, x_i \in C, i = 1, \dots, n, \sum_{i=1}^n \lambda_i = 1\}.$$

The *convex hull* of a set  $C$  is the smallest convex set containing  $C$ . A *hyperplane* is a set of the form  $\{h^T x = k\}$ , where  $h \in \mathbb{R}^N, h \neq 0$  and  $k \in \mathbb{R}$ . A hyperplane divides  $\mathbb{R}^N$  into two *half-spaces*.

**Definition 4** (Half-space). *A closed half-space is a set of the form  $\{h^T x \leq k\}$ , where  $h \in \mathbb{R}^N, h \neq 0$  and  $k \in \mathbb{R}$ .*

A *supporting hyperplane* of a set  $C$  is a hyperplane  $\{h^T x = k\}$ , such that  $C$  is entirely contained in one of the two closed half-spaces defined by the hyperplane and  $C$  has at least one point on the hyperplane.

**Definition 5** (Polytope). *A closed full dimensional polytope  $\mathcal{X} \subset \mathbb{R}^N$  is defined as the convex hull of at least  $N + 1$  affinely independent points in  $\mathbb{R}^N$ .*

The set of points  $v_1, \dots, v_n \in \mathbb{R}^N$  whose convex hull gives  $\mathcal{X}$  and with the property that for all  $i = 1, \dots, n$ , point  $v_i$  is not contained in the convex hull of  $v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_n$  is called the set of *vertices* of  $\mathcal{X}$  and is denoted by  $\mathcal{V}(\mathcal{X})$ . A polytope is completely described by its set of vertices:

$$\mathcal{X} = \text{hull}(v_1, \dots, v_n), \quad (2.1)$$

where  $\mathcal{V}(\mathcal{X}) = \{v_1, \dots, v_n\}$ . Alternatively, a polytope  $\mathcal{X}$  can be described as the intersection of at least  $N + 1$  closed half spaces. In other words, there exists a  $n \geq N + 1$  and  $h_i \in \mathbb{R}^N, k_i \in \mathbb{R}, i = 1, \dots, n$  such that

$$\mathcal{X} = \bigcap_{i=1}^n \{x \in \mathbb{R}^N \mid h_i^T x \leq k_i\}. \quad (2.2)$$

Forms (2.1) and (2.2) are referred to as the V- and H- representations of a polytope, respectively. Given a polytope, there exist algorithms for translation between its V- and H-representations [Motzkin et al., 1953, Kvasnica et al., 2004]. A *facet* of a polytope  $\mathcal{X}$  is the intersection of  $\mathcal{X}$  with one of its supporting hyperplanes

$\{h_i^T x = k_i\}$  from equation (2.2). A polytope without its facets is called an *open polytope*. Given an open polytope  $\mathcal{X}$ , we use  $cl(\mathcal{X})$  to denote the closure of  $\mathcal{X}$  (i.e. the union of  $\mathcal{X}$  and its facets).

## 2.2 Operations on Polytopes

Given polytopes  $\mathcal{X}_1, \mathcal{X}_2 \subset \mathbb{R}^N$ , we define the following operations:

**Definition 6** (Set Difference). *The set difference of  $\mathcal{X}_1$  and  $\mathcal{X}_2$  is defined as:*

$$\mathcal{X}_1 \setminus \mathcal{X}_2 = \{x \in \mathbb{R}^N \mid x \in \mathcal{X}_1, x \notin \mathcal{X}_2\}.$$

**Definition 7** (Minkowski Sum). *The Minkowski sum of  $\mathcal{X}_1$  and  $\mathcal{X}_2$  is defined as:*

$$\mathcal{X}_1 \oplus \mathcal{X}_2 = \{x_1 + x_2 \in \mathbb{R}^N \mid x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2\}.$$

**Definition 8** (Minkowski Difference). *The Minkowski difference of  $\mathcal{X}_1$  and  $\mathcal{X}_2$  is defined as:*

$$\mathcal{X}_1 \ominus \mathcal{X}_2 = \{x_1 - x_2 \in \mathbb{R}^N \mid x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2\}.$$

The Minkowski difference  $\mathcal{X}_1 \ominus \mathcal{X}_2$  can also be computed as the Minkowski sum  $\mathcal{X}_1 \oplus (-\mathcal{X}_2)$ , where  $(-\mathcal{X}_2) = \{x \in \mathbb{R}^N \mid -x \in \mathcal{X}_2\}$  is the mirror image of  $\mathcal{X}_2$  around the origin. Note that our definition of Minkowski difference follows [LaValle, 2006] and is different from the Pontryagin (Minkowski) difference from [Kvasnica, 2008, Grieder, 2004].

**Definition 9** (Chebyshev Ball). *The Chebyshev ball of a polytope  $\mathcal{X} \subset \mathbb{R}^N$  is the largest radius ball  $B_r(x_c) = \{x \in \mathbb{R}^N \mid \|x - x_c\|_2 \leq r\}$  such that  $B_r(x_c) \subset \mathcal{X}$ . We use  $c(\mathcal{X})$  to denote the center and  $r(\mathcal{X})$  to denote the radius of the Chebyshev ball of  $\mathcal{X}$ .*

## 2.3 Affine functions on Polytopes

**Definition 10** (Affine function). *A function  $f : \mathbb{R}^N \rightarrow \mathbb{R}^M$  is called affine if it can be written as  $f(x) = Ax + b$ ,  $A \in \mathbb{R}^{M \times N}$ ,  $b \in \mathbb{R}^M$ , for all  $x \in \mathbb{R}^N$ .*

If  $\mathcal{X}$  is a full dimensional polytope in  $\mathbb{R}^N$  with set of vertices  $\mathcal{V}(\mathcal{X}) = \{v_1, \dots, v_n\}$  and  $f : \mathbb{R}^N \rightarrow \mathbb{R}^M$  is an affine function, then

$$f(\mathcal{X}) = \text{hull}\{f(v_1), \dots, f(v_n)\}, \quad (2.3)$$

*i.e.*, the image of a polytope through an affine function is the convex hull of the vertex images through the affine function.

In the particular case  $N = M$ , if matrix  $A$  is nonsingular, then the vertices, facets, and interior of the polytope map through the affine transformation to the vertices, facets, and interior of the image of the polytope, respectively. Therefore

$$f(\text{cl}(\mathcal{X})) = \text{cl}(f(\mathcal{X})) \quad (2.4)$$

## 2.4 Transition Systems

**Definition 11** (Transition System). *A transition system is a tuple  $\mathcal{T} = (Q, \rightarrow, O, o)$ , where*

- $Q$  is a (possibly infinite) set of states,
- $\rightarrow \subseteq Q \times Q$  is a transition relation,
- $O$  is a finite set of observations, and
- $o : Q \rightarrow O$  is an observation map.

A transition  $(x, x') \in \rightarrow$  is also denoted by  $x \rightarrow x'$ . Transition system  $\mathcal{T}$  is *finite* if its set of states  $Q$  is finite and *infinite* otherwise, *deterministic* if, for all  $x \in Q$ , there exists at most one  $x' \in Q$  such that  $(x, x') \in \rightarrow$ , and *non-blocking* if, for every state  $x \in Q$ , there exists  $x' \in Q$  such that  $(x, x') \in \rightarrow$ . In this work only non-blocking transition systems are considered.

A *trajectory* of  $\mathcal{T}$  starting from state  $x_0 \in Q$  is an infinite sequence  $t = x_0 x_1 x_2 \dots$  with the property that  $x_k \in Q$ , and  $(x_k, x_{k+1}) \in \rightarrow$ , for all  $k \geq 0$ . A trajectory

$t = x_0x_1x_2\dots$  defines a *word*  $w = w_0w_1w_2\dots$ , where  $w_k = o(x_k)$ . The set of all words generated by the set of all trajectories starting at  $x \in Q$  is called the *language* of  $\mathcal{T}$  originating at  $x$  and is denoted by  $\mathcal{L}_{\mathcal{T}}(x)$ . A subset  $X \subseteq Q$  is called a *region* of  $\mathcal{T}$  and the language of  $\mathcal{T}$  originating at  $X$  is  $\mathcal{L}_{\mathcal{T}}(X) = \bigcup_{x \in X} \mathcal{L}_{\mathcal{T}}(x)$ . The language of  $\mathcal{T}$  is defined as  $\mathcal{L}_{\mathcal{T}}(Q)$ , which for simplicity is denoted as  $\mathcal{L}_{\mathcal{T}}$ . For an arbitrary region  $X$ , we define the set of states  $Pre_{\mathcal{T}}(X)$  that reach  $X$  in one step as

$$Pre_{\mathcal{T}}(X) = \{x \in Q \mid \exists x' \in X, x \rightarrow x'\} \quad (2.5)$$

Similarly, we define the set of states  $Post_{\mathcal{T}}(X)$  that can be reached from  $X$  in one step as

$$Post_{\mathcal{T}}(X) = \{x' \in Q \mid \exists x \in X, x \rightarrow x'\} \quad (2.6)$$

Note that transition system  $\mathcal{T}$  from Definition 11 is autonomous (*i.e.* it does not have a set of inputs). In Chapter 6 we will extend this definition further and consider transition systems with inputs but for the subsequent chapters this definition is sufficient and leads to simpler notation. In addition, note that in a more general definition of a transition system,  $o$  maps a state  $x \in Q$  to a set of observations  $o(x) \in 2^{\mathcal{O}}$  but for the purposes of this work a state can only have a single observation. As it become clear later this is always the case for the systems we consider and does not restrict the generality of the developed methods.

## 2.5 Simulation and Bisimulation Quotients

The observation map  $o$  of a transition system  $\mathcal{T} = (Q, \rightarrow, O, o)$  induces an equivalence relation  $\sim \subseteq Q \times Q$  over the set of states  $Q$  of  $\mathcal{T}$ .

**Definition 12** (Observational Equivalence). *States  $x_1, x_2 \in Q$  are observationally equivalent (written as  $x_1 \sim x_2$ ) if and only if  $o(x_1) = o(x_2)$ .*

The equivalence relation naturally induces a *quotient transition system*  $\mathcal{T}/\sim =$

$(Q/\sim, \rightarrow\sim, O, o\sim)$ .  $Q/\sim$  is the quotient space (the set of all equivalence classes). Given an equivalence class  $X \in Q/\sim$ , we denote the set of all equivalent states in that class by  $con(X) \subseteq Q$ , where  $con$  stands for concretization map. If  $\mathbb{X} \in 2^{Q/\sim}$  is a region of  $\mathcal{T}/\sim$ , then  $con(\mathbb{X}) = \bigcup_{X \in \mathbb{X}} con(X)$  is a region of  $\mathcal{T}$ . Since all states  $x \in Q$  in an equivalence class  $X \in Q/\sim$  have the same observation,  $o\sim(X)$  is well defined and given by  $o\sim(X) = o(x), x \in con(X)$ . The transition relation  $\rightarrow\sim$  is defined as follows: for  $X_1, X_2 \in Q/\sim$ ,  $X_1 \rightarrow\sim X_2$  if and only if there exist  $x_1 \in con(X_1)$  and  $x_2 \in con(X_2)$  such that  $x_1 \rightarrow x_2$ . It is easy to see that for all  $X \in Q/\sim$ ,

$$\mathcal{L}_{\mathcal{T}}(con(X)) \subseteq \mathcal{L}_{\mathcal{T}/\sim}(X). \quad (2.7)$$

The quotient transition system  $\mathcal{T}/\sim$  is said to *simulate* the original system  $\mathcal{T}$ .

**Definition 13** (Bisimulation). *The equivalence relation  $\sim$  induced by the observation map  $o$  is a bisimulation of a transition system  $\mathcal{T} = (Q, \rightarrow, O, o)$  if, for all states  $x_1, x_2 \in Q$ , if  $x_1 \sim x_2$  and  $x_1 \rightarrow x'_1$ , then there exist  $x'_2 \in Q$  such that  $x_2 \rightarrow x'_2$  and  $x'_1 \sim x'_2$ .*

If  $\sim$  is a bisimulation, then the quotient transition system  $\mathcal{T}/\sim$  is called a *bisimulation quotient* of  $\mathcal{T}$ , and the transition systems  $\mathcal{T}$  and  $\mathcal{T}/\sim$  are called *bisimilar*. An immediate consequence of bisimulation is language equivalence, *i.e.*, for all  $X \in Q/\sim$ ,

$$\mathcal{L}_{\mathcal{T}}(con(X)) = \mathcal{L}_{\mathcal{T}/\sim}(X). \quad (2.8)$$

Using the  $Pre_{\mathcal{T}}()$  operator defined in Equation (2.5), a characterization of bisimulation can be given as follows: the equivalence relation  $\sim$  is a bisimulation if and only if for all equivalence classes  $X' \in Q/\sim$ ,  $Pre_{\mathcal{T}}(con(X'))$  is either empty or a finite union of equivalence classes. Equivalently, the bisimulation property (Definition 13) is violated at  $X \in Q/\sim$  if there exists a state  $X' \in Q/\sim$ , such that

$$\emptyset \subset con(X) \cap Pre_{\mathcal{T}}(con(X')) \subset con(X). \quad (2.9)$$

This leads to an iterative procedure for the construction of the coarsest bisimulation  $\sim$ , known as the “bisimulation algorithm” [Bouajjani et al., 1991, Kanellakis and Smolka, 1990], which is summarized in Algorithm 1. In order to execute an iteration of the algorithm, one must be able to represent (possibly infinite) state sets, perform Boolean operations, check emptiness, and compute the predecessor operation  $Pre_{\mathcal{T}}$  on the representation of the state sets. If the bisimulation Algorithm 1 terminates then  $\mathcal{T}/\sim$  is a finite bisimulation quotient but, in general, termination cannot be guaranteed, since an infinite transition system does not always have a finite bisimulation quotient.

---

**Algorithm 1**  $\sim = \text{BISIMULATION}(\mathcal{T})$ : Construct the coarsest bisimulation  $\sim$  of  $\mathcal{T}$

---

- 1: Initialize  $\sim$  with observational equivalence
  - 2: **while** there exist  $X, X' \in Q/\sim$  such that  $\emptyset \subset con(X) \cap Pre_{\mathcal{T}}(con(X')) \subset con(X)$   
**do**
  - 3:   Construct state  $X_1$  such that  $con(X_1) := con(X) \cap Pre_{\mathcal{T}}(con(X'))$ ;
  - 4:   Construct state  $X_2$  such that  $con(X_2) := con(X) \setminus Pre_{\mathcal{T}}(con(X'))$ ;
  - 5:    $Q/\sim := Q/\sim \setminus \{X\} \cup \{X_1, X_2\}$ ;
  - 6: **end while**
  - 7: return  $\sim$ ;
- 

## 2.6 Linear Temporal Logic

We use Linear Temporal Logic (LTL) formulas [Clarke et al., 1999, Baier and Katoen, 2008] in order to specify properties for system trajectories. Temporal logic formulas are constructed using atomic proposition (which, as it will become clear later, will be the observations of a transition system) and basic temporal and Boolean operators. We use the standard notation for the Boolean operators (*i.e.*,  $\top$  (true),  $\neg$  (negation),  $\wedge$  (conjunction)) and the graphical notation for the temporal operators (*e.g.*,  $\bigcirc$  (“next”),  $\mathbf{U}$  (“until”)). The  $\bigcirc$  operator is a unary prefix operator and is followed by a single LTL formula, while  $\mathbf{U}$  is a binary infix operator. Formally, we define the

syntax of LTL formulas as follows:

**Definition 14** (LTL Syntax). *A (propositional) linear temporal logic (LTL) formula  $\phi$  over a given set of atomic propositions  $O$  is recursively defined as*

$$\phi = \top \mid o \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \bigcirc\phi \mid \phi_1 \mathbf{U}\phi_2, \quad (2.10)$$

where  $o \in O$  is an atomic proposition.

Unary operators have a higher precedence than binary ones and  $\neg$  and  $\bigcirc$  bind equally strong. The temporal operator  $\mathbf{U}$  takes precedence over  $\neg$  and  $\wedge$  and is right-associative (e.g.  $\phi_1 \mathbf{U}\phi_2 \mathbf{U}\phi_3$  stands for  $\phi_1 \mathbf{U}(\phi_2 \mathbf{U}\phi_3)$ ).

To obtain the full expressivity of propositional logic, additional operators are defined as

$$\begin{aligned} \phi_1 \vee \phi_2 &= \neg(\neg\phi_1 \wedge \neg\phi_2) \\ \phi_1 \rightarrow \phi_2 &= \neg\phi_1 \vee \phi_2 \\ \phi_1 \leftrightarrow \phi_2 &= (\phi_1 \rightarrow \phi_2) \wedge (\phi_2 \rightarrow \phi_1) \\ &\dots \end{aligned}$$

In addition, the temporal operators  $\diamond$  (“eventually”) and  $\square$  (“always”) are defined:

$$\begin{aligned} \diamond\phi &= \top \mathbf{U}\phi \\ \square\phi &= \neg\diamond\neg\phi \end{aligned}$$

By combining the various temporal operators, more complicated expressions can be obtained. For example, we will frequently use the combinations  $\diamond\square$  (“eventually always”) and  $\square\diamond$  (“always eventually”).

LTL formulas are interpreted over infinite words in a set of observation  $O$ , as those generated by the transition system  $\mathcal{T}$  from Definition 11 (i.e. words in the language  $\mathcal{L}_{\mathcal{T}}$ ). Note that more generally the semantics of LTL formulas can be given



over infinite words in the power set of a set of observations  $2^O$  [Clarke et al., 1999] but as we discussed previously, the states of the transition systems we consider can only have one observation (*i.e.*  $\forall x \in Q, o(x) \in O$ ).

**Definition 15** (Semantics of LTL formulas). *The satisfaction of formula  $\phi$  over a set of observations  $O$  at position  $k \in \mathbb{N}$  of word  $w$ , denoted by  $w_k \models \phi$ , is defined recursively as follows:*

- $w_k \models \top$ ,
- $w_k \models o$  for some  $o \in O$  if  $w_k = o$ ,
- $w_k \models \neg\phi$  if  $w_k \not\models \phi$ ,
- $w_k \models \phi_1 \vee \phi_2$  if  $w_k \models \phi_1$  or  $w_k \models \phi_2$ ,
- $w_k \models \bigcirc\phi$  if  $w_{k+1} \models \phi$ ,
- $w_k \models \phi_1 \mathcal{U} \phi_2$  if there exist a  $j \geq k$  such that  $w_j \models \phi_2$  and for all  $k \leq i < j$  we have  $w_i \models \phi_1$

A word  $w$  satisfies an LTL formula  $\phi$ , written as  $w \models \phi$ , if  $w_1 \models \phi$ . We denote the language of infinite words that satisfy formula  $\phi$  by  $\mathcal{L}_\phi$ . The transition system  $\mathcal{T}$  satisfies formula  $\phi$  from region  $X \subseteq Q$ , written as  $\mathcal{T}(X) \models \phi$ , if and only if  $w \models \phi$  for all  $w \in \mathcal{L}_\mathcal{T}(X)$ . In other words,  $\mathcal{T}(X) \models \phi$  if and only if  $\mathcal{L}_\mathcal{T}(X) \subseteq \mathcal{L}_\phi$ .

In the following, we give an informal interpretation of the satisfaction of some frequently used LTL formulas.

- $\bigcirc\phi$  is satisfied at the current step if  $\phi$  is satisfied at the next step.
- $\phi_1 \mathbf{U} \phi_2$  is satisfied if  $\phi_1$  is satisfied “until”  $\phi_2$  becomes satisfied,
- $\Box\phi$  is satisfied if  $\phi$  is satisfied at each step (*i.e.*  $\phi$  is “always” satisfied).
- $\Box\neg\phi$  is satisfied if  $\neg\phi$  is satisfied at each step (*i.e.*  $\phi$  is “never” satisfied).
- $\Diamond\phi$  is satisfied if  $\phi$  is satisfied at some future step (*i.e.*  $\phi$  is “eventually” satisfied).

- $\diamond\Box\phi$  is satisfied if  $\phi$  becomes satisfied at some future step and remains satisfied for all following steps (*i.e.*  $\phi$  is satisfied “eventually forever”).
- Formula  $\Box\diamond\phi$  is satisfied if  $\phi$  always becomes satisfied at some future step (*i.e.*  $\phi$  is satisfied “infinitely often”).

## 2.7 Büchi and Rabin Automata

**Definition 16** (Büchi automaton). *A (nondeterministic) Büchi automaton is a tuple  $\mathcal{B} = (S_{\mathcal{B}}, S0_{\mathcal{B}}, O, \delta_{\mathcal{B}}, F_{\mathcal{B}})$ , where*

- $S_{\mathcal{B}}$  is a finite set of states,
- $S0_{\mathcal{B}} \subseteq S_{\mathcal{B}}$  is the set of initial states,
- $O$  is the input alphabet,
- $\delta_{\mathcal{B}} : S \times O \rightarrow 2^{S_{\mathcal{B}}}$  is a nondeterministic transition function, and
- $F \subseteq S$  is the set of accepting (final) states.

A Büchi automaton is deterministic if both  $S0_{\mathcal{B}}$  and  $\delta_{\mathcal{B}}(s, o)$  are singletons for all  $s \in S_{\mathcal{B}}$  and  $o \in O$ . The semantics of a Büchi automaton are defined over infinite input words in  $O^{\omega}$ . A run of  $\mathcal{B}$  over a word  $w = w_0w_1w_2\dots \in O^{\omega}$  is a sequence  $\rho = s_0s_1s_2\dots$ , where  $s_0 \in S0_{\mathcal{B}}$  and  $s_{k+1} \in \delta_{\mathcal{B}}(s_k, w_k)$  for all  $k \geq 1$ . Let  $\text{inf}(\rho)$  denote the set of states that appear in the run  $\rho$  infinitely often.

**Definition 17** (Büchi acceptance). *A run  $\rho$  of  $\mathcal{B}$  is accepting if and only if  $\text{inf}(\rho) \cap F_{\mathcal{B}} \neq \emptyset$ . In other words, an input word  $w$  is accepted by  $\mathcal{B}$  if and only if there exists at least one run over  $w$  that visits  $F_{\mathcal{B}}$  infinitely often.*

We denote by  $\mathcal{L}_{\mathcal{B}}$  the language accepted by  $\mathcal{B}$ , *i.e.* the set of all words accepted by  $\mathcal{B}$ . An LTL formula  $\phi$  over a set  $O$  can always be translated into a Büchi automaton  $\mathcal{B}_{\phi}$  that accepts all and only words satisfying  $\phi$  [Wolper et al., 1983] (*i.e.*  $\mathcal{L}_{\mathcal{B}_{\phi}} = \mathcal{L}_{\phi}$ ). This translation can be performed using efficient, off-the-shelf software tools such as LTL2BA [Gastin and Oddoux, 2001].

**Definition 18** (Rabin automaton). *A (nondeterministic) Rabin automaton is a tuple  $\mathcal{R} = (S_{\mathcal{R}}, S0_{\mathcal{R}}, O, \delta_{\mathcal{R}}, F_{\mathcal{R}})$ , where*

- $S_{\mathcal{R}}$  is a finite set of states,
- $S0_{\mathcal{R}} \subseteq S_{\mathcal{R}}$  is the set of initial states,
- $O$  is the input alphabet,
- $\delta_{\mathcal{R}} : S \times O \rightarrow 2^{S_{\mathcal{R}}}$  is a transition map, and
- $F_{\mathcal{R}} = \{(G_1, B_1), \dots, (G_n, B_n)\}$  is the acceptance condition.

A Rabin automaton  $\mathcal{R}$  is deterministic if both  $S0_{\mathcal{R}}$  and  $\delta_{\mathcal{R}}(s, o)$  are singletons for all  $s \in S_{\mathcal{R}}$  and  $o \in O$ . The semantics of a Rabin automaton are defined over infinite input words in  $O^\omega$ . A run of  $\mathcal{R}$  over a word  $w_0w_1w_2\dots \in O^\omega$  is a sequence  $\rho = s_0s_1s_2\dots$ , where  $s_0 \in S0_{\mathcal{R}}$  and  $s_{k+1} \in \delta_{\mathcal{R}}(s_k, o_k)$  for all  $k \geq 1$ . Let  $\text{inf}(\rho)$  denote the set of states that appear in the run  $\rho$  infinitely often.

**Definition 19** (Rabin acceptance). *A run  $\rho$  is accepting if  $\text{inf}(\rho) \cap G_i \neq \emptyset \wedge \text{inf}(\rho) \cap B_i = \emptyset$  for some  $i \in \{1, \dots, n\}$ . An input word is accepted by an automaton if some run over it is accepting.*

We denote by  $\mathcal{L}_{\mathcal{R}}$  the language accepted by  $\mathcal{R}$ , *i.e.*, the set of all words accepted by  $\mathcal{R}$ . Given an LTL formula  $\phi$ , one can build a deterministic Rabin automaton  $\mathcal{R}$  with  $2^{2^{\mathcal{O}(|\phi| \cdot \log |\phi|)}}$  states and  $2^{\mathcal{O}(|\phi|)}$  pairs in its acceptance condition, such that  $\mathcal{L}_{\mathcal{R}} = \mathcal{L}_{\phi}$  [Vardi and Wolper, 1986, Safra, 1988]. The translation can be done using off-the-shelf software tools such as LTL2DSTAR [Klein and Baier, 2006].

## 2.8 Model Checking

Given a finite transition system  $\mathcal{T} = (Q, \rightarrow, O, o)$  and an LTL formula  $\phi$  over  $O$ , checking whether the words of  $\mathcal{T}$  satisfy  $\phi$  is called LTL model checking, or simply model checking. An off-the-shelf model checker, such as SPIN [Holzmann, 1997],

NuSMV [Cimatti et al., 2002] or DiVinE [Barnat et al., 2009], takes as input a finite transition system  $\mathcal{T}$  and a formula  $\phi$ , and returns the set of states  $Q_\phi$  of  $\mathcal{T}$  at which the formula is satisfied (*i.e.*, the set for which  $\mathcal{L}_\mathcal{T}(Q_\phi) \subseteq \mathcal{L}_\phi$ ). For the non-satisfying states, a model checker returns a non-satisfying trajectory as a certifying counter-example.

We use an in-house implementation of LTL model checking [Kloetzer and Belta, 2008b] (denoted by `MODEL-CHECK()`) in order to separate the translation of a formula  $\phi$  to the accepting Büchi automaton [Gastin and Oddoux, 2001] from the rest of the computation involved in model checking. This gives us more control over the model checking process and leads to more efficient implementation of the proposed methods.

Given a region  $X \subseteq Q$ ,  $\text{MODEL-CHECK}(\mathcal{T}, X, \phi) = \{x \in X \mid \mathcal{T}(x) \models \phi\}$  is the subset of  $X$  satisfying the formula. Let

$$X_\mathcal{T}^\phi = \{x \in Q \mid \mathcal{T}(x) \models \phi\}. \quad (2.11)$$

Note that  $X_\mathcal{T}^\phi = \text{MODEL-CHECK}(\mathcal{T}, Q, \phi)$  and if  $x \notin X_\mathcal{T}^\phi$ , then there exists a word in  $\mathcal{L}_\mathcal{T}(x)$  that violates  $\phi$ . Therefore,  $X_\mathcal{T}^\phi$  is the largest region of  $\mathcal{T}$  satisfying  $\phi$ .

If  $\mathcal{T}/\sim$  is a quotient of  $\mathcal{T}$ , then for any equivalence class  $S \in Q/\sim$  and formula  $\phi$ , we have:

$$\mathcal{T}/\sim(S) \models \phi \Rightarrow \mathcal{T}(\text{con}(S)) \models \phi \quad (2.12)$$

In addition, if  $\sim$  is a bisimulation, then

$$\mathcal{T}/\sim(S) \models \phi \Leftrightarrow \mathcal{T}(\text{con}(S)) \models \phi \quad (2.13)$$

Properties (2.12) and (2.13) (which follow immediately from (2.7) and (2.8)) allow one to model check finite quotients and extend the results to the (possibly infinite) original transition systems.

## Chapter 3

# Finite Abstractions of PWA systems

In Chapters 4,5, and 6 we develop methods for the analysis, parameter synthesis and control of piecewise affine (PWA) systems from specifications given as temporal logic formulas. In this chapter, we establish the foundation for these methods by developing a framework under which formal specifications of trajectories of PWA systems can be expressed and studied.

First, in Section 3.1 we provide a formal definition of PWA systems with uncertain parameters and discuss their semantics. We use the concepts defined in Chapter 2 to formalize the satisfaction of temporal logic formulas by trajectories of a PWA system by embedding it into a transition system. In Section 3.2, we describe the computation of a finite quotient of an embedding transition system. We show that for PWA systems, either exact or over-approximation finite quotients can be computed through polyhedral operations.

The results presented in this chapter are sufficient to allow the model checking of PWA systems. However, standard model checking techniques can only lead to positive verification guarantees when all trajectories of the system satisfy the specification, which is restrictive, especially when uncertain parameter systems are considered. Therefore, based on the results presented in this chapter, in Chapter 4 we will develop methods for the analysis of PWA systems that can provide more informative results.

### 3.1 PWA Systems with Uncertain Parameters

Let  $\mathcal{X}_l, l \in L$  be a set of open polytopes in  $\mathbb{R}^N$ , where  $L$  is a finite index set, such that  $\mathcal{X}_{l_1} \cap \mathcal{X}_{l_2} = \emptyset$  for all  $l_1, l_2 \in L, l_1 \neq l_2$  and  $\mathcal{X} = \bigcup_{l \in L} \text{cl}(\mathcal{X}_l)$  is a closed full-dimensional polytope in  $\mathbb{R}^N$  ( $\text{cl}(\mathcal{X}_l)$  denotes the closure of set  $\mathcal{X}_l$ ). A discrete-time piecewise affine (PWA) system with polytopic parameter uncertainty is defined as:

$$\Xi : x_{k+1} = Ax_k + b, x_k \in \mathcal{X}_l, l \in L, k = 0, 1, 2, \dots, \quad (3.1)$$

where parameters  $A$  and  $b$  are uncertain, but known to belong to polytopic uncertainty sets  $\mathcal{P}_l^A \subset \mathbb{R}^{N \times N}$  and  $\mathcal{P}_l^b \subset \mathbb{R}^N$ , respectively. We call  $\Xi$  a *fixed parameter* or a *deterministic* PWA system if, for all  $l \in L$ , both  $\mathcal{P}_l^A$  and  $\mathcal{P}_l^b$  are singletons. The particular case when only the matrix component of the parameters is fixed is of special interest (*i.e.* for all  $l \in L, \mathcal{P}_l^A$  are singletons). This corresponds to a system subjected to additive noise only and in Section 3.2 we will show that such additional constraints can be exploited.

System  $\Xi$  evolves along different affine dynamics in different regions of the continuous state space  $\mathcal{X}$ . When  $\Xi$  is in state  $x_k \in \mathcal{X}_l$  for some  $l \in L$ , the next visited state  $x_{k+1}$  is computed according to the affine map of Equation 3.1, where all parameters  $A \in \mathcal{P}_l^A$  and  $b \in \mathcal{P}_l^b$  are possible. It is important to note that we treat the choice of available parameters as nondeterministic rather than probabilistic.

As it will become clear later, we are interested in studying properties of trajectories of system (3.1) specified in terms of the polytope index set  $L$  from its definition. Informally, the semantics of system (3.1) can be understood in the following sense: a trajectory  $x_0 x_1 x_2 \dots$  starting at  $x_0 \in \mathcal{X}_{l_0}$  for some  $l_0 \in L$  can be obtained by arbitrarily selecting parameters  $A_{l_0} \in \mathcal{P}_{l_0}^A$  and  $b_{l_0} \in \mathcal{P}_{l_0}^b$ , applying the affine map of Equation (3.1) to compute  $x_1$ , finding  $l_1 \in L$  such that  $x_1 \in \mathcal{X}_{l_1}$ , and repeating this

procedure for each subsequent step. A trajectory produces a word  $l_0l_1l_2\dots$ , where  $l_i \in L$  is the index of the polytope visited at step  $i$  (i.e.  $x_i \in \mathcal{X}_{l_i}$ ).

In general, it is possible that trajectories of (3.1) leave polytope  $\mathcal{X}$ . While we are not interested in such trajectories, we capture them by defining an additional observation *Out*, and trivial dynamics  $x_{k+1} = x_k$  when  $x_k \notin \mathcal{X}$  (e.g., a trajectory  $x_0x_1x_2x_3\dots$  satisfying  $x_0, x_1 \in \mathcal{X}_{l_0}, x_2 \in \mathcal{X}_{l_1}$  for some  $l_0, l_1 \in L$  and  $x_3 \notin \mathcal{X}$  produces a word  $l_0l_1Out\dots$ , where *Out* is repeated infinitely). In subsequent sections, we will provide a formal way to specify and forbid such behavior.

System (3.1) is always nonblocking and therefore all its trajectories are infinite and produce infinite words. An LTL formula over  $L \cup \{Out\}$  can then be interpreted over trajectories of the system (see Section 2.6). We formalize the satisfaction of LTL formulas by trajectories of (3.1) through an embedding into a transition system.

**Definition 20** (Embedding Transition System). *The embedding transition system  $\mathcal{T}_\Xi = (Q_\Xi, \rightarrow_\Xi, O_\Xi, o_\Xi)$  for piecewise affine system  $\Xi$  from (3.1) is defined as:*

- $Q_\Xi = \mathbb{R}^N$ ,
- $(x, x') \in \rightarrow_\Xi$  if and only if  $x \notin \mathcal{X}$  and  $x = x'$ , or there exist  $l \in L$  such that  $x \in \mathcal{X}_l$  and there exist  $A_l \in \mathcal{P}_l^A, b_l \in \mathcal{P}_l^b$  such that  $x' = A_lx + b_l$ ,
- $O_\Xi = L \cup \{Out\}$ ,
- $o_\Xi(x) = l$  if and only if there exist  $l \in L$  such that  $x \in \mathcal{X}_l$  and  $o_e(x) = Out$  otherwise

Note that the embedding transition system  $\mathcal{T}_\Xi$  has an infinite number of states and is always non-blocking. Furthermore, if the parameters of the PWA system are fixed then  $\mathcal{T}_\Xi$  is deterministic.

**Definition 21.** *Given a subset  $X \subseteq Q_\Xi$ , we say that all trajectories of system  $\Xi$  (Equation (3.1)) originating in  $X$  satisfy formula  $\phi$  if and only if  $\mathcal{T}_\Xi(X)$  satisfies  $\phi$ .*

In subsequent chapters we study the satisfaction of a specification  $\phi$ , given as an LTL formula over  $L \cup \{Out\}$ , by the embedding transition system  $\mathcal{T}_\Xi$ . From Definition 21, this is equivalent to the satisfaction of  $\phi$  by trajectories of the piecewise affine system  $\Xi$ . However, it is important to discuss the assumptions, which are implicit in Definitions 20 and 21 and might seem restrictive. First, we capture only the reachability of open full dimensional polytopes in the semantics of the embedding. This is enough for practical purposes, since only sets of measure zero are disregarded, and it is unreasonable to assume that equality constraints can be detected in real-world applications. There are two situations in which the boundaries can affect the semantics of the trajectories non-trivially: (1) when trajectories originate and remain in such sets for all times, and (2) when trajectories start in open polytopes and then "vanish" in the boundaries. For both these situations, the system dynamics and the polytopes need to satisfy special conditions. Second, the specification is given over the indexes  $l \in L$  of the polytopes  $\mathcal{X}_l$  from the system definition. However, arbitrary linear inequalities can be accommodated simply by refining the polytopic partition. The resulting PWA system will have some polytopes with identical dynamics. A region satisfying an inequality can then be represented as the disjunction of the indexes  $l \in L$  of all polytopes satisfying the inequality.

### 3.2 Finite Quotients of PWA Systems

In Section 2.5 we defined the quotient  $\mathcal{T}/\sim = (Q/\sim, \rightarrow_{\sim}, O, o_{\sim})$  of a transition system  $\mathcal{T}$ . In this section we show that the quotient  $\mathcal{T}_\Xi/\sim = (Q_\Xi/\sim, \rightarrow_{\Xi, \sim}, O_\Xi, o_{\Xi, \sim})$  of the embedding  $\mathcal{T}_\Xi$  of PWA system  $\Xi$ , defined in Section 3.1, is finite and computable through polyhedral operations. We show that if the matrix component of the parameters of  $\Xi$  is fixed, then the quotient can be computed exactly, while otherwise an over-approximation quotient can be computed.



From the definitions of the observational equivalence relation  $\sim$  (Section 2.5), induced by observation map  $o_{\Xi}$  and  $\mathcal{T}_{\Xi}$  (Definition 20), the set of states  $Q_{\Xi}/\sim$ , of the finite quotient  $\mathcal{T}_{\Xi}/\sim$ , is simply the set of observations  $Q_{\Xi}/\sim = O_{\Xi} = L \cup \{Out\}$  and the observation map is identity. Given a state  $l \in Q_{\Xi}/\sim, l \neq Out, con(l) = \mathcal{X}_l$  is a polytope from the system definition (Equation (3.1)). Although the set  $con(Out) = \bigcup_{l \in L} \mathcal{X}_l$  is infinite and cannot be easily represented, its explicit computation is not required for our methods.

In order to finish the construction of the quotient, we need to compute the set of transitions  $\rightarrow_{\Xi, \sim}$ . By the definition of quotient transitions  $\rightarrow_{\sim}$  (Section 2.5) and the set of reachable states given in Equation (2.6), the transition relation  $\rightarrow_{\Xi, \sim}$  can be constructed if  $Post_{\mathcal{T}_{\Xi}}()$  is computable. Explicitly, for any two equivalence classes  $l, l' \in (Q_{\Xi}/\sim \setminus \{Out\})$ , we have:

$$(l, l') \in \rightarrow_{\Xi, \sim} \text{ if and only if } Post_{\mathcal{T}_{\Xi}}(con(l)) \cap con(l') \neq \emptyset$$

or equivalently,

$$(l, l') \in \rightarrow_{\Xi, \sim} \text{ if and only if } Post_{\mathcal{T}_{\Xi}}(\mathcal{X}_l) \cap \mathcal{X}_{l'} \neq \emptyset \quad (3.2)$$

Similarly, given a state  $l \in Q_{\Xi}/\sim, l \neq Out$ , transitions to state  $Out$  can be assigned as:

$$(l, Out) \in \rightarrow_{e, \sim} \text{ if and only if } Post_{\mathcal{T}_{\Xi}}(\mathcal{X}_l) \not\subseteq \mathcal{X}, \quad (3.3)$$

and state  $Out$  only has a transition to itself (*i.e.*  $(Out, Out) \in \rightarrow_{\Xi, \sim}$ ).

Given a polytope  $\mathcal{X}_l$  for some  $l \in L$ , the set  $Post_{\mathcal{T}_{\Xi}}(\mathcal{X}_l)$  is convex and can be computed exactly in the particular case when the matrix component of the parameters of the PWA system (3.1) is fixed, (*i.e.*,  $A = \mathcal{P}_l^A$ , is a singleton). Explicitly,

$$Post_{\mathcal{T}_{\Xi}}(\mathcal{X}_l) = A\mathcal{X}_l \oplus \mathcal{P}_l^b, \quad (3.4)$$

where  $A\mathcal{X}_i$  is the image of the polytope  $\mathcal{X}_i$  through matrix  $A$  (see Section 2.3) and “ $\oplus$ ” stands for Minkowski (set) sum (Definition 7). Therefore, the set of transitions  $\rightarrow_{\Xi, \sim}$  can be computed using polyhedral operations and the finite quotient  $\mathcal{T}_{\Xi}/\sim$  can be constructed. The computation for the case when the matrix component of the parameters of  $\Xi$  is fixed is summarized in Algorithm 3.2.

---

**Algorithm 2**  $\mathcal{T}_{\Xi}/\sim = \text{QUOTIENT}(\Xi)$  : Compute the finite quotient  $\mathcal{T}_{\Xi}/\sim$  of PWA system  $\Xi$

---

```

1:  $Q_{\Xi}/\sim = L \cup \{Out\}$ 
2:  $O_{\Xi} = Q_{\Xi}/\sim$ 
3:  $o_{\Xi, \sim}$  is identity
4: for each  $l \in Q_{\Xi}/\sim$  do
5:   if  $\mathcal{X}_l \not\subseteq \mathcal{X}$  then
6:     add transition  $(l, Out) \in \rightarrow_{\Xi, \sim}$ 
7:   end if
8:   for each  $l' \in Q_{\Xi}/\sim$  do
9:     if  $Post_{\mathcal{T}_{\Xi}}(\mathcal{X}_l) \cap \mathcal{X}_{l'} \neq \emptyset$  then
10:      add transition  $(l, l') \in \rightarrow_{\Xi, \sim}$ 
11:     end if
12:   end for
13: end for
14: return  $\mathcal{T}_{\Xi}/\sim = (Q_{\Xi}/\sim, \rightarrow_{\Xi, \sim}, O_{\Xi}, o_{\Xi, \sim})$ 

```

---

When the matrix component of the parameters is allowed to vary, given a polytope  $\mathcal{X}_i$ , the set  $Post_{\mathcal{T}_{\Xi}}(\mathcal{X}_i)$  is not necessarily convex and, in general, there are no algorithms capable of its exact computation.

**Proposition 1.** *Given a polytope  $\mathcal{X}_i$ , a convex over-approximation of  $Post_{\mathcal{T}_{\Xi}}(\mathcal{X}_i)$  can be computed as:*

$$\overline{Post_{\mathcal{T}_{\Xi}}(\mathcal{X}_i)} = \text{hull}\{Ax \mid A \in \mathcal{V}(\mathcal{P}_i^A), x \in \mathcal{V}(\mathcal{X}_i)\} \oplus \mathcal{P}_i^b, \quad (3.5)$$

where  $\text{hull}()$  and  $\mathcal{V}()$  denote the convex hull and set of vertices, respectively (see Section 2.1).

*Proof.* Let  $\mathcal{V}(\mathcal{X}_i) = \{v_1, \dots, v_R\}$  and  $\mathcal{V}(\mathcal{P}_i^A) = \{w_1, \dots, w_M\}$ . Let  $x \in \mathcal{X}_i$  and

$A \in \mathcal{P}_l^A$ . Then  $x = \sum_{r=1}^R \lambda_r v_r$ ,  $A = \sum_{m=1}^M \mu_m w_m$ , and

$$Ax = \left( \sum_{m=1}^M \mu_m w_m \right) \left( \sum_{r=1}^R \lambda_r v_r \right) = \sum_{m=1}^M \sum_{r=1}^R \mu_m \lambda_r w_m v_r$$

Since  $\mu_m, \lambda_r \geq 0$  and  $\sum_{m=1}^M \mu_m = \sum_{r=1}^R \lambda_r = 1$  then  $\mu_m \lambda_r \geq 0$  for any  $m, r$  and  $\sum_{m=1}^M \sum_{r=1}^R \mu_m \lambda_r = 1$ . Therefore

$$Ax \in \text{hull}\{wv, w \in \mathcal{V}(\mathcal{P}_l^A), v \in \mathcal{V}(\mathcal{X})\}$$

and the rest of the proof follows from Definition 7.  $\square$

A related treatment of 1 can be found in in [Barmish and Sankaran, 1979], where it is shown that this over-approximation is the smallest convex set containing  $Post_{\mathcal{T}_{\Xi}}(\mathcal{X}_l)$ :

$$\overline{Post_{\mathcal{T}_{\Xi}}(\mathcal{X}_l)} \subseteq Post_{\mathcal{T}_{\Xi}}(\mathcal{X}_l) \quad (3.6)$$

Although a precise distance between the real set and its over-approximation is hard to quantify, it has been established through extensive simulation that in general, the volume of  $Post_{\mathcal{T}_{\Xi}}()$  is not significantly increased by the approximation.

Using the over-approximation  $\overline{Post_{\mathcal{T}_e}(\mathcal{X}_l)}$ , instead of the exact  $Post_{\mathcal{T}_{\Xi}}(\mathcal{X}_l)$  an over-approximation quotient  $\overline{\mathcal{T}_{\Xi}/\sim} = (Q_{\Xi}/\sim, \overrightarrow{\Xi}/\sim, O, o_{\sim})$  can be constructed. From Equation (3.6), it follows that for all  $l \in Q_{\Xi}/\sim$ , we have  $\rightarrow_{e,\sim} \subseteq \overrightarrow{e,\sim}$ , which leads to

$$\mathcal{L}_{\mathcal{T}_{\Xi}} \subseteq \mathcal{L}_{\mathcal{T}_{\Xi}/\sim} \subseteq \mathcal{L}_{\overline{\mathcal{T}_{\Xi}/\sim}}. \quad (3.7)$$

Therefore, the over-approximation quotient  $\overline{\mathcal{T}_{\Xi}/\sim}$  simulates the exact quotient  $\mathcal{T}_{\Xi}/\sim$  and the embedding transition system  $\mathcal{T}_{\Xi}$  and can be used instead of  $\mathcal{T}_{\Xi}/\sim$  for the methods we develop in subsequent chapters but the results become more conservative. The over-approximation quotient  $\overline{\mathcal{T}_{\Xi}/\sim}$  can be computed through Algorithm 3.2 by substituting the  $Post_{\mathcal{T}_{\Xi}}()$  operation with its over-approximation  $\overline{Post_{\mathcal{T}_{\Xi}}}$  when the

matrix component of the parameters of  $\Xi$  is uncertain.

The results presented in this chapter are sufficient to allow PWA systems to be model checked using standard tools such as SPIN [Holzmann, 1997], NuSMV [Cimatti et al., 2002] or DiVinE [Barnat et al., 2009]. Given a PWA system  $\Xi$ , the exact finite quotient  $\mathcal{T}_\Xi/\sim$  or the over-approximation finite quotient  $\overline{\mathcal{T}_\Xi/\sim}$  can be constructed as described in Section 3.2. Then,  $\mathcal{T}_\Xi/\sim$  or  $\overline{\mathcal{T}_\Xi/\sim}$  can be checked against an LTL formula  $\phi$  over the index set  $L$  of  $\Xi$ . In addition, the special observation *Out* can be used as an atomic proposition in  $\phi$ .

As a simple example, we consider the problem of guaranteeing that region  $\mathcal{X}$  is an invariant for all trajectories of a PWA system  $\Xi$ . We formulate the specification  $\phi = \Box \neg \text{Out}$  requiring that trajectories of the system never visit the region labeled by *Out*. In other words, satisfying trajectories of  $\Xi$  will never leave  $\mathcal{X}$ . We can model check the quotient  $\mathcal{T}_\Xi/\sim$  against  $\phi$  using standard tools and if  $\mathcal{T}_\Xi/\sim$  satisfies  $\phi$  we can guarantee that all trajectories of  $\Xi$  satisfy the specification (the same is true for the over-approximation quotient  $\overline{\mathcal{T}_\Xi/\sim}$ ). In subsequent chapters we will use this strategy to guarantee that trajectories of  $\Xi$  do not leave the defined state space of the system.

It is important to note that simply model checking the quotient  $\mathcal{T}_\Xi/\sim$  (and especially the over-approximation  $\overline{\mathcal{T}_\Xi/\sim}$ ) is restrictive. While the satisfaction of the formula can be guaranteed for all trajectories of  $\Xi$  when the quotient satisfies the formula, nothing can be guaranteed if the formula is violated. Since, in general,  $\mathcal{T}_\Xi/\sim$  is coarse (it contains few states), positive verification results can be rarely obtained. In Chapter 4 we extend the standard model checking methods in order to obtain more informative results and develop an analysis method for PWA systems based on the construction of finite quotients described in this chapter.

## Chapter 4

# Formal Analysis of PWA systems

In Chapter 3 we defined the satisfaction of LTL formulas by trajectories of PWA system  $\Xi$  through an embedding into an infinite transition system  $\mathcal{T}_\Xi$ . We showed that the quotient  $\mathcal{T}_\Xi/\sim$  (Section 2.5) is finite and can be computed through a procedure based on polyhedral operations, when the matrix component of the system parameters was fixed. For an uncertain parameter system, we showed that a finite, over-approximation quotient  $\overline{\mathcal{T}_\Xi}/\sim$  can be computed through polyhedral operations. The results presented in Chapter 3 were sufficient to allow the model checking of PWA systems. In this chapter, we use those results to formulate a procedure for the analysis of PWA systems from specifications given as temporal logic formulas. Specifically, we consider the following problem:

**Problem 1.** *Given a discrete-time piecewise affine system (3.1) and an LTL formula  $\phi$ , find the largest region of initial states, from which all trajectories of the system satisfy  $\phi$ , while always remaining within  $\mathcal{X}$ .*

Unlike approaches that attempt to synthesize parameters for the system from a temporal logic specification, which will be discussed in Chapter 5, here we assume that uncertainty is inherent and therefore the parameter ranges cannot be restricted further. Instead, we attempt to guarantee the satisfaction of a property by selecting appropriate initial states for the system. Our approach is based on the iterative model checking [Clarke et al., 1999] and construction of discrete abstractions in the form of finite transition systems as described in Chapter 3.

The solution to Problem 1 amounts to the computation of  $X_{\mathcal{T}_\Xi}^{\phi'}$  (see Equation (2.11)), where  $\phi' = \phi \wedge \Box \neg \text{Out}$ . This guarantees that all trajectories originating there satisfy  $\phi$  and always remain within  $\mathcal{X}$ . In addition,  $X_{\mathcal{T}_\Xi}^{\phi'}$  is the largest satisfying region, since there exist trajectories originating in all states  $x \notin X_{\mathcal{T}_\Xi}^{\phi'}$  that either violate  $\phi$  or leave  $\mathcal{X}$ .

Since  $\mathcal{T}_\Xi$  has an infinite number of states, it cannot be analyzed directly but using results from Chapter 3, we can compute the finite quotient  $\mathcal{T}_\Xi / \sim = \text{QUOTIENT}(\Xi)$  (Algorithm 3.2) or its over-approximation. In Section 4.1 we discuss the general problem of finding the largest region  $X_{\mathcal{T}}^{\phi}$  satisfying LTL formula  $\phi$  for a (possibly infinite) transition system  $\mathcal{T}$  under the assumption that the quotient  $\mathcal{T} / \sim$  is computable. We propose algorithms for iterative refinement and model checking, where the results are valid in general for any transition system. As it will become clear later, our approach is conservative, in the sense that, we can only “try” to find the satisfying region  $X_{\mathcal{T}}^{\phi}$  but, in general, we can only guarantee to obtain subsets of it. In Section 4.2 we discuss conditions guaranteeing that an exact solution is obtained and propose additional optimizations based on the construction of formula-equivalent finite quotients. The implementation of the model checking and refinement procedures developed in this chapter for  $\mathcal{T}_\Xi$  are discussed in Sec. 4.3, which provides the solution to Problem 1.

As mentioned in Chapter 3, when all parameters of PWA system  $\Xi$  are fixed (*i.e.*  $\mathcal{P}_l^A$  and  $\mathcal{P}_l^b$  in Equation (3.1) are singletons for all  $l \in L$ ) the embedding  $\mathcal{T}_\Xi$  is deterministic. Motivated by this, we highlight additional efficiency improvements to our methods when the transition system being analyzed is infinite but deterministic.

## 4.1 Formal Analysis of Infinite Transition Systems

The solution to Problem 1 amounts to the computation of the largest region of  $T_{\Xi}$  satisfying an LTL formula  $\phi$ . However, the embedding transition system  $T_e$  (Definition 20) is infinite and therefore the computation cannot be performed directly.

In this section, we consider the following problem:

**Problem 2.** *Given an infinite transition system  $\mathcal{T}$  (Definition 11) and an LTL formula  $\phi$  over its set of observations  $O$ , find  $X_{\mathcal{T}}^{\phi}$  (Equation (2.11)).*

We assume that, given the observational equivalence relation  $\sim$  (see Section 2.5), the finite quotient  $\mathcal{T}/\sim$  is computable (the computation of a finite quotient for  $T_{\Xi}$  was discussed in Chapter 3). Then,  $X_{\mathcal{T}/\sim}^{\phi}$  can be computed by model checking and from Equation (2.12) it follows that  $con(X_{\mathcal{T}/\sim}^{\phi})$  is a satisfying region in  $\mathcal{T}$  but, in general, it is not the largest satisfying region (*i.e.*,  $con(X_{\mathcal{T}/\sim}^{\phi}) \subseteq X_{\mathcal{T}}^{\phi}$ ). If, on the other hand,  $\sim$  is a bisimulation of  $\mathcal{T}$  then from Equation (2.13) it follows that for any LTL formula  $\phi$

$$con(X_{\mathcal{T}/\sim}^{\phi}) = X_{\mathcal{T}}^{\phi}. \quad (4.1)$$

The most intuitive solution to Problem 2 would then be to apply the bisimulation algorithm (Algorithm 1 in Section 2.5) and refine  $\sim$  to compute the coarsest bisimulation of  $\mathcal{T}$ . Then, by model checking the bisimulation quotient  $\mathcal{T}/\sim$  the satisfying region  $X_{\mathcal{T}/\sim}^{\phi}$  can be computed and from Equation (2.13), it follows that  $con(X_{\mathcal{T}/\sim}^{\phi}) = X_{\mathcal{T}}^{\phi}$  is the largest satisfying region of  $\mathcal{T}$ . However, such a procedure would only work for the particular case when  $\mathcal{T}$  admits a finite bisimulation quotient.

Alternatively, the equivalence relation produced at each step of the bisimulation algorithm 1 can be used to construct finite simulation quotients, which can then be model checked against an LTL formula. This is the approach we follow in this section in order to obtain a solution to Problem 2. While the method is conservative, we

discuss conditions guaranteeing that an exact solution is obtained.

#### 4.1.1 Iterative model checking

Even though the quotient  $\mathcal{T}/\sim$  cannot always be refined enough to be bisimilar with  $\mathcal{T}$ , region  $X_{\mathcal{T}/\sim}^\phi$  can be computed at each step of a refinement procedure. Then,  $\text{con}(X_{\mathcal{T}/\sim}^\phi)$  can provide a conservative solution to Problem 2, which can be improved by additional refinement. If  $\hat{\mathcal{T}}/\sim = (\hat{Q}/\sim, \hat{\rightarrow}\sim, O, \hat{o}\sim)$  is the quotient after some refinement has been performed, we have

$$\mathcal{L}_T \subseteq \mathcal{L}_{\hat{\mathcal{T}}/\sim} \subseteq \mathcal{L}_{\mathcal{T}/\sim} \quad (4.2)$$

and

$$\text{con}(X_{\hat{\mathcal{T}}/\sim}^\phi) \subseteq \text{con}(X_{\mathcal{T}/\sim}^\phi) \subseteq X_T^\phi. \quad (4.3)$$

A related idea was used in [Chutinan and Krogh, 2001] for verification from formulas in the universal fragment ACTL of CTL. Such approaches face computational challenges, due to the possible explosion in the number of states of  $\hat{\mathcal{T}}/\sim$  as refinement progresses, and therefore minimizing the amount of refinement required to obtain a solution is critical.

Our methods aim at refining and model checking the quotient only at states where this can improve the solution (*i.e.* increase  $X_{\hat{\mathcal{T}}/\sim}^\phi$ ). Clearly, refinement of any state of the quotient  $X \in \hat{Q}/\sim$ , where  $X \in X_{\hat{\mathcal{T}}/\sim}^\phi$  is unnecessary, since all trajectories originating there satisfy the formula. Similarly, the set  $X_{\hat{\mathcal{T}}/\sim}^{-\phi}$  can be computed and refinement of any state  $X \in X_{\hat{\mathcal{T}}/\sim}^{-\phi}$  is also unnecessary, since only trajectories violating the formula originate there.

In general, it is possible that only some but not all of the trajectories originating from a particular state  $X$  satisfy the formula, in which case  $X \notin X_{\hat{\mathcal{T}}/\sim}^\phi$  and  $X \notin X_{\hat{\mathcal{T}}/\sim}^{-\phi}$ . Refinement of such states might isolate a subregion containing only satisfying



trajectories, which can then be included in  $X_{\hat{T}/\sim}^\phi$ . Refinement of the quotient at any state does not change the satisfaction of the formula at state  $X$ , where  $X \in X_{\hat{T}/\sim}^\phi$  or  $X \in X_{\hat{T}/\sim}^{\neg\phi}$ . Therefore, once a state has been identified as satisfying the formula or its negation it is no longer considered for refinement or model checking. More specifically, for any state of the quotient  $X \in X_{\hat{T}/\sim}^\phi$  additional refinement of any other states can only “shrink” the language from  $X$ , which guarantees the satisfaction of  $\phi$  (the same holds for the negation of the formula). Similarly, refining a state  $X$  can only shrink the language produced from states outside  $X$ .

Our selective approach limits the explosion in the number of states that have to be considered as refinement progresses and leads to a procedure that iteratively refines the quotient  $\hat{T}/\sim$  and possibly expands  $X_{\hat{T}/\sim}^\phi$  and  $X_{\hat{T}/\sim}^{\neg\phi}$  at each iteration (Algorithm 3).

---

**Algorithm 3**  $con(X_{\hat{T}/\sim}^\phi) = \text{ANALYZE}(\mathcal{T}, \phi)$ : Find an under-approximation of the largest region of an infinite transition system  $\mathcal{T}$  satisfying an LTL formula  $\phi$  (i.e.  $con(X_{\hat{T}/\sim}^\phi) \subseteq X_{\mathcal{T}}^\phi$ )

---

```

Construct  $\mathcal{T}/\sim$ 
Initialize  $\hat{\mathcal{T}}/\sim := \mathcal{T}/\sim$ 
Initialize  $X_{\hat{\mathcal{T}}/\sim}^\phi := \text{MODEL-CHECK}(\hat{\mathcal{T}}/\sim, \hat{Q}/\sim, \phi)$ 
Initialize  $X_{\hat{\mathcal{T}}/\sim}^{\neg\phi} := \text{MODEL-CHECK}(\hat{\mathcal{T}}/\sim, \hat{Q}/\sim, \neg\phi)$ 
repeat
   $\mathbb{X}_r := \{X \in \hat{Q}/\sim \mid X \text{ is large enough, } X \notin X_{\hat{\mathcal{T}}/\sim}^\phi, X \notin X_{\hat{\mathcal{T}}/\sim}^{\neg\phi}\}$ 
  for each  $X \in \mathbb{X}_r$  do
     $\hat{\mathcal{T}}/\sim := \text{REFINE}(\hat{\mathcal{T}}/\sim, X)$ 
  end for
   $\hat{\mathbb{X}}_r := \hat{Q}/\sim \setminus (X_{\hat{\mathcal{T}}/\sim}^\phi \cup X_{\hat{\mathcal{T}}/\sim}^{\neg\phi})$ 
   $X_{\hat{\mathcal{T}}/\sim}^\phi := X_{\hat{\mathcal{T}}/\sim}^\phi \cup \text{MODEL-CHECK}(\hat{\mathcal{T}}/\sim, \hat{\mathbb{X}}_r, \phi)$ 
   $X_{\hat{\mathcal{T}}/\sim}^{\neg\phi} := X_{\hat{\mathcal{T}}/\sim}^{\neg\phi} \cup \text{MODEL-CHECK}(\hat{\mathcal{T}}/\sim, \hat{\mathbb{X}}_r, \neg\phi)$ 
until  $\mathbb{X}_r = \emptyset$ 
return  $con(X_{\hat{\mathcal{T}}/\sim}^\phi)$ 

```

---

The set  $\mathbb{X}_r \subseteq Q/\sim$ , computed in Algorithm 3, contains the states of the quotient

where refinement should be targeted in order to improve the solution (*i.e.* expand  $X_{\hat{T}/\sim}^\phi$ ).  $\mathbb{X}_r$  contains only states that are “large enough” to undergo refinement (see Section 4.3 for a description of such a measure for  $\mathcal{T}_e$ ), guaranteeing that the algorithm will terminate if a sufficient number of iterations is performed. Refinement of any state  $X \notin \mathbb{X}_r$  will not expand  $X_{\hat{T}/\sim}^\phi$  significantly since  $X$  is either satisfying (*i.e.*  $X \in X_{\hat{T}/\sim}^\phi$ ), violating (*i.e.*  $X \in X_{\hat{T}/\sim}^{-\phi}$ ), or small.

In order to implement Algorithm 3, the finite quotient  $\mathcal{T}/\sim$  must be computable, the possibly infinite  $\text{con}(X_{\hat{T}/\sim}^\phi)$  must be represented and a refinement procedure for  $\mathcal{T}/\sim$  that can be applied locally at a state  $X \in \mathcal{Q}/\sim$  is required. Note that the refinement strategy  $\text{REFINE}(\hat{\mathcal{T}}/\sim, X)$  is not yet specified. In the following subsections, we will propose different refinement strategies that exploit the structure of the system and discuss the details of their implementations.

#### 4.1.2 Quotient Refinement

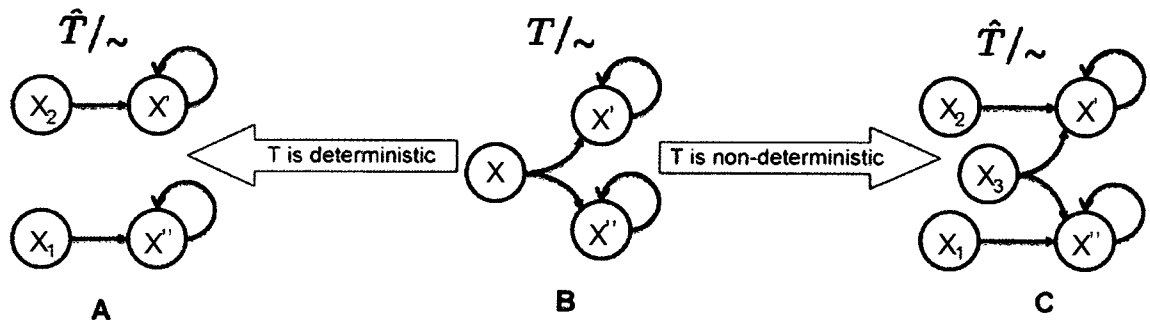


Figure 4-1: An example of the application of  $\text{REFINE}(\mathcal{T}/\sim, X)$  to the quotient  $\mathcal{T}/\sim$ .

If  $\mathcal{T}/\sim$  is a finite bisimulation quotient, then an exact solution to Problem 2 can be obtained by applying Algorithm 3. Motivated by this, we formulate a refinement procedure  $\text{REFINE}()$  (Algorithm 4) inspired by the bisimulation algorithm (see Section 2.5 and Algorithm 1). Unlike the bisimulation algorithm, which refines the equivalence relation  $\sim$  globally,  $\text{REFINE}(\mathcal{T}/\sim, X)$  refines the quotient  $\mathcal{T}/\sim$  locally at

a state  $X \in Q/\sim$ . This allows us to target refinement to specific states (as in Algorithm 3), while the quotient is updated instead of recomputed every time refinement is performed.

---

**Algorithm 4**  $\hat{T}/\sim = \text{REFINE}(T/\sim, X)$ : Refine  $T/\sim$  at state  $X \in Q/\sim$

---

```

Initialize  $\mathbb{X}_r := \{X\}$ 
while there exist  $X_r \in \mathbb{X}_r, X' \in \text{Post}_{T/\sim}(X)$  such that
 $\emptyset \subset \text{con}(X_r) \cap \text{Pre}_T(\text{con}(X')) \subset \text{con}(X_r)$  do
  Construct states  $X_1, X_2$  such that:
     $\text{con}(X_1) := \text{con}(X_r) \cap \text{Pre}_T(\text{con}(X'))$ 
     $\text{con}(X_2) := \text{con}(X_r) \setminus \text{Pre}_T(\text{con}(X'))$ 
   $\mathbb{X}_r := (\mathbb{X}_r \setminus X_r) \cup \{X_1, X_2\}$ 
end while
update  $\hat{\rightarrow}_\sim$ 
update  $\hat{o}_\sim$ 
return  $\hat{T}/\sim = (\hat{Q}/\sim, \hat{\rightarrow}_\sim, O, \hat{o}_\sim)$ 

```

---

$\text{REFINE}(T/\sim, X)$  partitions state  $X$  in such a way that all resulting subsets of  $X$  satisfy the bisimulation property (*i.e.* for all subsets of  $X$  there does not exist a state  $X' \in Q/\sim$  such that Equation (2.9) is satisfied).

From the construction of the quotient  $T/\sim$  described in Chapter 3, it follows that for any states  $X, X' \in Q/\sim$ ,  $\text{con}(X) \cap \text{Pre}_T(\text{con}(X')) \neq \emptyset$  if and only if  $X' \in \text{Post}_{T/\sim}(X)$  (*i.e.*, if  $X'$  is reachable from  $X$  in  $T/\sim$ ). Then, all nonempty intersections  $\text{con}(X) \cap_{X' \in \mathbb{X}'} \text{Pre}_T(\text{con}(X')) \setminus \cup_{X'' \in \mathbb{X}''} \text{Pre}_T(\text{con}(X''))$ , where  $\mathbb{X}' \in 2^{\text{Post}_{T/\sim}(X)}$  and  $\mathbb{X}'' = \text{Post}_{T/\sim}(X) \setminus \mathbb{X}'$ , provide a partition of  $X$  such that all resulting states satisfy the bisimulation property. Therefore, applying  $\text{REFINE}(T/\sim, X)$  results in at most  $2^{|\text{Post}_{T/\sim}(X)|}$  subsets.

In the particular case when  $T$  is deterministic, given states  $X, X', X'' \in Q/\sim$  such that  $X', X'' \in \text{Post}_{T/\sim}(X)$ , we have  $\text{con}(X) \cap \text{Pre}_T(\text{con}(X')) \cap \text{Pre}_T(\text{con}(X'')) = \emptyset$ . Then,  $\text{con}(X) \cap \text{Pre}_T(\text{con}(X'))$  of all  $X' \in \text{Post}_{T/\sim}(X)$  provide a partition of  $X$  satisfying the bisimulation property and applying  $\text{REFINE}(T/\sim, X)$  on the quotient of

a deterministic system  $T$  results in at most  $|Post_{T/\sim}(X)|$  subsets. For a deterministic  $T$ ,  $\text{REFINE}(T/\sim, X)$  is more efficient and is summarized in Algorithm 5.

---

**Algorithm 5**  $\hat{T}/\sim = \text{REFINE}(T/\sim, X)$ : Refine  $T/\sim$  at state  $X \in Q/\sim$  for a deterministic  $T$

---

```

initialize  $\hat{Q}/\sim = Q/\sim \setminus X$ 
for each state  $X'$  such that  $X \rightarrow_{\sim} X'$  do
  construct state  $X_r$  such that  $con(X_r) = con(X) \cap Pre_T(con(X'))$ 
  add state  $X_r$  to  $\hat{Q}/\sim$ 
end for
update  $\hat{\rightarrow}_{\sim}$ 
update  $\hat{o}_{\sim}$ 
return  $\hat{T}/\sim = (\hat{Q}/\sim, \hat{\rightarrow}_{\sim}, O, \hat{o}_{\sim})$ 

```

---

An example of the refinement procedure implemented in  $\text{REFINE}(T/\sim, X)$  (Algorithms 4 and 5) is shown in Figure 4-1. The transitions from state  $X$  in the quotient  $T/\sim$  (Figure 4-1-B) are defined by  $Post_{T/\sim}(X) = \{X', X''\}$ . If  $T$  is deterministic, then  $con(X) \cap Pre_T(con(X')) \cap Pre_T(con(X'')) = \emptyset$ . In that case,  $\text{REFINE}(T/\sim, X)$  partitions  $X$  into states  $X_1$  and  $X_2$ , such that  $con(X_1) = con(X) \cap Pre_T(con(X'))$  and  $con(X_2) = con(X) \cap Pre_T(con(X''))$ . Transitions  $X_1 \hat{\rightarrow}_{\sim} X'$  and  $X_2 \hat{\rightarrow}_{\sim} X''$  are implicitly induced and the resulting quotient  $\hat{T}/\sim$  is shown in Figure 4-1-A. If  $T$  is non-deterministic,  $\text{REFINE}(T/\sim, X)$  partitions  $X$  into states  $X_1, X_2$  and  $X_3$ , such that  $con(X_1) = con(X) \cap Pre_T(con(X')) \setminus Pre_T(con(X''))$ ,  $con(X_2) = con(X) \cap Pre_T(con(X'')) \setminus Pre_T(con(X'))$  and  $con(X_3) = con(X) \cap Pre_T(con(X')) \cap Pre_T(con(X''))$ . Transitions  $X_1 \hat{\rightarrow}_{\sim} X'$ ,  $X_2 \hat{\rightarrow}_{\sim} X''$ ,  $X_3 \hat{\rightarrow}_{\sim} X'$  and  $X_3 \hat{\rightarrow}_{\sim} X''$  are implicitly induced and the resulting quotient  $\hat{T}/\sim$  is shown in Figure 4-1-C.

When refinement is performed using the  $Pre_T()$  operation, outgoing transitions of the newly formed states are implicitly induced. Given states  $X, X' \in Q/\sim$  such that  $X' \in Post_{T/\sim}(X)$ , the subset  $con(X) \cap Pre_T(con(X'))$  always has a transition to state  $X'$  (in fact, this is the only transition possible in the case when  $T$  is deterministic). Additionally, any subset of  $con(X) \setminus Pre_T(con(X'))$  can never have a transition to

state  $X'$ . In the particular case when state  $X$  has a self transition ( $X \rightarrow_{\sim} X$ ), transitions from subset  $con(X) \cap Pre_T(con(X))$  to all subsets of  $X$  resulting from its refinement are possible and must be recomputed (the computation of transitions between any two states is discussed in Section 4.3). Incoming transitions from all states  $X'' \in Pre_{T/\sim}(X)$  reaching  $X$  to all newly formed states are also updated, which completes the construction of  $\hat{\rightarrow}_{\sim}$ . All subsets of a refined state inherit the observation of the parent and, therefore,  $\hat{o}_{\sim}$  is easily updated.

So far, we have discussed a refinement strategy inspired by the bisimulation algorithm and, therefore, relying on the computation of the  $Pre_T()$  operation (see Fig. 4.1 for an example). If  $Pre_T()$  is not computable, any refinement strategy can be used for the function `REFINE` in Algorithm 3 with the hope that the smaller regions produced at each step separate satisfying and violating trajectories. In this case, when refinement is performed at state  $X$ , outgoing transitions from newly formed states are not implicitly induced and must be recomputed but only target states in the set  $Post_{T/\sim}(X)$  (instead of the entire  $Q/\sim$ ) need to be considered.

### 4.1.3 Conservatism

In general, the method described in this section is conservative and the solution to Problem 2 returned by Algorithm 3 is an under-approximation. Indeed, when Algorithm 3 terminates, there could be states left which are “too small” for additional refinement and have not been assigned as either satisfying the specification or its negation. In this subsection, we discuss conditions guaranteeing that an exact solution to Problem 2 was returned by Algorithm 3.

If the refined quotient  $\hat{T}/\sim$ , computed during the execution of Algorithm 3 is a bisimulation quotient, then from Equation (2.8) we can guarantee that an exact solution to Problem 2 was obtained. We can develop procedures for the efficient

characterizations of bisimulation.

**Proposition 2.** *The equivalence relation  $\sim$  is a bisimulation if the quotient  $\mathcal{T}/\sim$  is deterministic.*

*Proof.* Assume by contradiction that  $\sim$  is not a bisimulation. Then, there exist  $X, X' \in Q/\sim$ ,  $x_1, x_2 \in X$ , and  $x'_1 \in X'$  such that  $x_1 \rightarrow x'_1$  and there does not exist  $x'_2 \in X'$  such that  $x_2 \rightarrow x'_2$ . However, since  $\mathcal{T}$  is nonblocking, there exists  $x''_2 \in Q$  and  $X'' \in Q/\sim$ ,  $X'' \neq X'$  such that  $x_2 \rightarrow x''_2$ ,  $x''_2 \in X''$ . In the quotient  $\mathcal{T}/\sim$ , this induces transitions  $X \rightarrow X'$  and  $X \rightarrow X''$ , which implies that  $\mathcal{T}/\sim$  is non-deterministic. This contradicts the hypothesis, which concludes the proof.  $\square$

Proposition 2 offers a computationally attractive bisimulation test. Instead of computing the  $Pre()$  of each region and checking the intersection with all the other regions to test bisimulation using the characterization described in Equation 2.9, Proposition 2 requires only counting the number of outgoing transitions from each state in the quotient. For a deterministic transition systems, this result becomes stronger:

**Proposition 3.** *An equivalence relation  $\sim$  defined on a deterministic transition system  $\mathcal{T}$  is a bisimulation if and only if the quotient  $\mathcal{T}/\sim$  is deterministic.*

*Proof.* From Proposition 2 it follows that if the quotient is deterministic then the equivalence relation is a bisimulation. Assume by contradiction that  $\mathcal{T}/\sim$  is not deterministic. Then, there exist  $X, X', X'' \in Q/\sim$  such that  $X \rightarrow X'$  and  $X \rightarrow X''$ . However, since  $\sim$  is a bisimulation, there exists  $x, x', x'' \in Q$ ,  $x \in X$ ,  $x' \in X'$ ,  $x'' \in X''$  such that  $x \rightarrow x'$  and  $x \rightarrow x''$ , which implies that  $\mathcal{T}$  is non-deterministic. This contradicts the hypothesis, which concludes the proof.  $\square$

Through Propositions 2 and 3, we can check if the quotient obtained in Algorithm 3 is a bisimulation quotient and, if so, guarantee that an exact solution to Problem 2 was obtained. However, since states of  $\mathcal{T}/\sim$  that satisfy the formula or its negation are not refined further, in general, Algorithm 3 does not lead to the construction of bisimulation quotients.

Even when the quotient  $\mathcal{T}/\sim$  is not a bisimulation, we can guarantee that the largest region of  $\mathcal{T}$  satisfying an LTL formula  $\phi$  (Problem 2) is obtained if the states of  $\mathcal{T}/\sim$  can be partitioned into satisfying the formula and its negation. In other words,

$$X_{\mathcal{T}/\sim}^{\phi} \cup X_{\mathcal{T}/\sim}^{\neg\phi} = Q_{\sim} \Rightarrow \text{con}(X_{\mathcal{T}/\sim}^{\phi}) = X_{\mathcal{T}}^{\phi}.$$

In Section 4.2 we study further the conditions guaranteeing that an exact solution to Problem 2 is obtained and use this information to introduce additional optimizations to our method.

## 4.2 Formula Guided Refinement

In Section 4.1, our approach to Problem 2 involved combining state refinement inspired by the bisimulation algorithm, with model checking in an iterative procedure. At each step, the set  $\text{con}(X_{\mathcal{T}/\sim}^{\phi}) \subseteq X_{\mathcal{T}}^{\phi}$  provided an under-approximation of the solution. This under-approximation could be improved by performing additional iterations but the termination of the algorithm with an exact solution could not be guaranteed. In the following, we consider conditions guaranteeing that Equation (4.1) holds and therefore an exact solution to Problem 2 can be computed. As already stated in Section 4.1, bisimulation is one such condition, but as it will become clear later, it is unnecessarily strong.

### 4.2.1 $\phi$ -equivalence

**Definition 22.** *Given an (infinite) transition system  $\mathcal{T}$  and an LTL formula  $\phi$ , an observational equivalence relation  $\sim$  is a  $\phi$ -equivalence of  $\mathcal{T}$  if and only if for all states  $x_1, x_2 \in Q$  such that  $x_1 \sim x_2$ ,*

$$T(x_1) \models \phi \Leftrightarrow T(x_2) \models \phi$$

We denote a  $\phi$ -equivalence relation as  $\sim_{\phi}$  and refer to the quotient  $\mathcal{T}/\sim_{\phi}$  as  $\phi$ -

equivalent quotient. From Equation (2.13) it follows that a bisimulation relation  $\sim$  is a  $\phi$ -equivalence for all LTL formulas  $\phi$ . Bisimulation is a sufficient condition guaranteeing that Equation (4.1) holds but since we are interested in the analysis of  $\mathcal{T}$  for a specific LTL formula  $\phi$  it can be too restrictive.

**Proposition 4.** *Given a transition system  $\mathcal{T}$  and an LTL formula  $\phi$ , Equation (4.1) is satisfied if and only if  $\sim$  is a  $\phi$ -equivalence of  $\mathcal{T}$ .*

*Proof.* Assume that  $\sim$  is a  $\phi$ -equivalence. From Definition 22 it follows that  $\forall x \in Q$  such that  $\mathcal{T}(x) \models \phi, x \in \text{con}(X), X \in \mathcal{T}/\sim$  we have  $\mathcal{T}/\sim(X) \models \phi$ . Then,  $\forall x \in Q, x \in X_{\mathcal{T}}^{\phi} \Leftrightarrow x \in \text{con}(X_{\mathcal{T}/\sim}^{\phi})$  and therefore  $X_{\mathcal{T}}^{\phi} = \text{con}(X_{\mathcal{T}/\sim}^{\phi})$ . Assume that  $\sim$  is not a  $\phi$ -equivalence. Then,  $\exists x_1, x_2 \in Q$  such that  $x_1 \sim x_2, \mathcal{T}(x_1) \models \phi$  and  $\mathcal{T}(x_2) \not\models \phi$ . Considering the equivalence class  $X \in Q/\sim$  such that  $x_1, x_2 \in \text{con}(X)$  we have  $\mathcal{T}/\sim(X) \not\models \phi$ . Then  $x_1 \in X_{\mathcal{T}}^{\phi}$  but  $x_1 \notin \text{con}(X_{\mathcal{T}/\sim}^{\phi})$  and therefore  $X_{\mathcal{T}}^{\phi} \neq \text{con}(X_{\mathcal{T}/\sim}^{\phi})$ .  $\square$

Proposition 4 shows that  $\phi$ -equivalence is a necessary and sufficient condition for Equation (4.1). Then, Problem 2 reduces to the computation of  $\text{con}(X_{\mathcal{T}/\sim_{\phi}}^{\phi})$ , where  $\mathcal{T}/\sim_{\phi}$  is a finite,  $\phi$ -equivalent quotient for  $\mathcal{T}$ . We discuss the computation of formula equivalent quotients in Section 4.2.2.

An example where a formula equivalent quotient is not a bisimulation quotient is shown in Figure 4-2. The transition system shown in Figure 4-2-A forms three equivalence classes under observational equivalence. The resulting finite quotient is shown in Figure 4-2-B. The finite quotient is clearly not a bisimulation quotient and the bisimulation property (Equation (2.9)) is violated at state  $X_1$ . However, the quotient is  $\phi$ -equivalent for LTL formula  $\phi = \bigcirc(X_2 \vee X_3)$  and can be equivalently used instead of the original system for model checking against the formula.

Our approach described in Section 4.1 involved the iterative model checking and refinement of simulation quotients of an infinite transition system. By model checking with both an LTL formula and its negation we were able to target refinement to the specific set of states from which some but not all runs satisfied the formula. Under



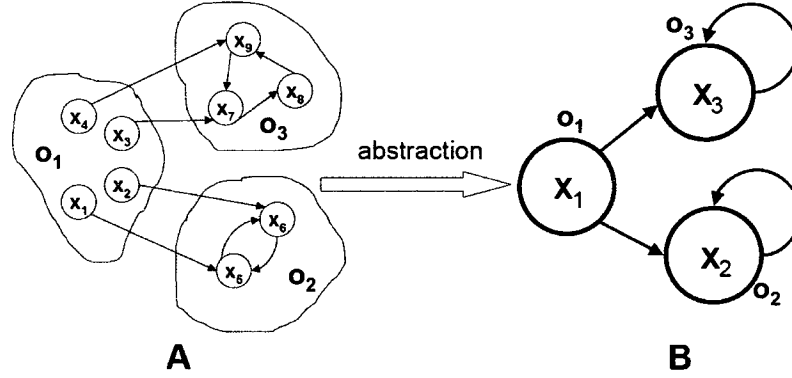


Figure 4-2: Formula equivalent quotient (B) of transition system (A).

some special conditions discussed in Section 4.1, the solution returned by Algorithm 3 can be exact rather than an under-approximation, in which case a formula formula equivalent quotients had been constructed. However, a large number of model checking and refinement steps was required to achieve this. The approach described in this section aims directly at the construction of formula equivalent quotients and is more efficient.

#### 4.2.2 Constructing $\phi$ -equivalent quotients

In this section we develop an algorithm for the computation of  $\phi$ -equivalent quotients of (possibly infinite) transitions systems, leveraging ideas from the bisimulation algorithm (Algorithm 1) and automata-based model checking. We assume that given a deterministic transition system  $T$  and the observational equivalence relation  $\sim$ , the finite quotient  $T/\sim$  is computable (in Chapter 3 we showed that this is always the case for  $T_{\Xi}$ ). For the sake of presentation, we also assume that LTL formula  $\phi$  can be translated into a deterministic Büchi automaton  $\mathcal{B}_{\phi}$ . In general, there exist LTL formulas that can only be translated to non-deterministic Büchi automata and a nondeterministic Büchi automaton cannot always be determinized [Safra, 1989].

This restricts the expressivity of the method to a fragment of LTL but, by noting that any LTL formula can be translated into a language-equivalent deterministic Rabin automaton (see Section 2.7), our solution can be adapted to allow full LTL expressivity. We will describe a method based on the use of deterministic Rabin automata in Chapter 6.

Since the computation of  $\phi$ -equivalent quotients is guided by formula  $\phi$ , it is most natural to perform the computation in the product automata  $P = \mathcal{T}/\sim \otimes \mathcal{B}_\phi$  (Definition 23), where both the structure of the system ( $\mathcal{T}/\sim$ ) and the specification ( $\mathcal{B}_\phi$ ) is captured.

**Definition 23.** *The product automaton  $\mathcal{P} = \mathcal{T}/\sim \otimes \mathcal{B}_\phi$  of a finite transition system  $\mathcal{T}/\sim = (Q/\sim, \rightarrow_\sim, O, o_\sim)$  and a Büchi automaton  $\mathcal{B}_\phi = (S_B, S_{0_B}, O, \delta_{\mathcal{B}_\phi}, F_B)$  accepting the language  $\mathcal{L}_\phi$  for some LTL formula  $\phi$  is defined as  $\mathcal{P} = (S_P, S_{P0}, \delta_P, F_P)$ , where*

- $S_P = Q/\sim \times S_B$  is the set of states,
- $S_{P0} = Q/\sim \times S_{0_B}$  is the set of initial states,
- $\delta_P$  is the transition function, where for a  $(X, s) \in S_P$ ,  
 $\delta_P((X, s)) = \{(X', s') \in S_P \mid X \rightarrow_\sim X' \text{ and } s' = \delta_{\mathcal{B}_\phi}(s, o(X))\}$ ,
- $F_P = Q/\sim \times F_B$  is the set of accepting states.

The product automaton is a nondeterministic Büchi automaton with input alphabet containing only one element, which is therefore omitted. An accepting run  $r_P = (X_1, s_1)(X_2, s_2) \dots$  in  $P$  can be projected into a run  $r_{\mathcal{T}/\sim} = X_1 X_2 \dots$  of  $\mathcal{T}/\sim$ , such that  $o(X_1)o(X_2) \dots$  is accepted by  $\mathcal{B}_\phi$  [Vardi and Wolper, 1986] and therefore satisfies  $\phi$ . Let us denote by  $\alpha : S_P \rightarrow Q/\sim$ ,  $\alpha(X, s) = X$ , the projection of states of product automaton  $\mathcal{P}$  to the states of  $\mathcal{T}/\sim$ .

The set  $X_{\mathcal{T}/\sim}^\phi$  can be computed as the projection  $\alpha(S_Y \cap S_{P0}) \subseteq Q/\sim$  where  $S_Y \subseteq S_P$  is the set of states in  $\mathcal{P}$  from which all runs are accepting (see Section 2.6).  $S_Y$  can be efficiently computed following the method described in [Kloetzer

and Belta, 2008a]. Specifically, a subset  $F_Y \subseteq F$  of accepting states, from which infinitely many revisits to  $F_P$  can be guaranteed is first identified.  $S_Y$  is then a set of states from which a visit to  $F_Y$  can be guaranteed in zero or more steps.

We can also easily identify a set of states  $S_N \subseteq S_P$  of  $\mathcal{P}$  from which no runs are accepting. The projection  $\alpha(S_N \cap S_{P0}) \subseteq Q/\sim$  corresponds to  $X_{T/\sim}^{\neg\phi}$  (i.e. the largest set of states of  $T/\sim$  from which no runs satisfy  $\phi$ ).

Let  $S_\gamma \subseteq S_P$ ,  $S_\gamma = S_P \setminus (S_Y \cup S_N)$  be the set of states from which some but not all runs are accepting in  $\mathcal{P}$ . The projection  $\alpha(S_\gamma \cap S_{P0}) \subseteq Q/\sim$  corresponds to states of  $T/\sim$  where both runs satisfying  $\phi$  and  $\neg\phi$  originate. If transition system  $\mathcal{T}$  is deterministic, the  $\phi$ -equivalence property (Definition 22) is violated at those states.

**Proposition 5.** *The equivalence relation  $\sim$  is a  $\phi$ -equivalence of a deterministic transition system  $T$  if and only if  $(S_\gamma \cap S_{P0}) = \emptyset$ . Then,  $S_\gamma = \emptyset$  guarantees that  $\sim$  is a  $\phi$ -equivalence.*

*Proof.* Let  $(S_\gamma \cap S_{P0}) \neq \emptyset$ ,  $(X, s) \in (S_\gamma \cap S_{P0})$ . Then  $X = \alpha((X, s))$  is a state of  $T/\sim$  such that  $\exists x_1, x_2 \in \text{con}(X)$ ,  $T(x_1) \models \phi$ ,  $T(x_2) \models \neg\phi$  and therefore  $\sim$  is not a  $\phi$ -equivalence. Let  $(S_\gamma \cap S_{P0}) = \emptyset$ . Then  $\forall X \in Q/\sim$  we have  $\forall x \in \text{con}(X)$ ,  $T(x) \models \phi$  or  $\forall x \in \text{con}(X)$ ,  $T(x) \models \neg\phi$  and therefore  $\sim$  is a  $\phi$ -equivalence.  $\square$

In general, the set  $S_\gamma$  is nonempty but can be made empty if accepting and non-accepting runs from each state  $(X, s) \in S_\gamma$  are separated through refinement. Following from Proposition 5 and the discussion presented in Section 4.1 this provides a solution to Problem 2. Since the structure of  $P$  is completely determined by  $\mathcal{B}_\phi$  and  $T/\sim$  and  $\mathcal{B}_\phi$  is fixed, the only way to refine states in  $P$  is through refinement of  $T/\sim$ . We refine a state  $(X, s) \in S_\gamma$  by applying the procedure  $\text{REFINE}(T/\sim, \alpha(X, s))$  that we described in Section 4.1. Once a state  $X \in Q/\sim$  is refined in  $T/\sim$  the states and transitions of  $P$  must be updated accordingly (see Definition 23) by applying the  $\text{UPDATE}(P, (X, s))$  procedure, resulting in a refined product automata  $\hat{P}$ . If transition system  $\mathcal{T}$  is nondeterministic, then the equivalence relation  $\sim$  can be a  $\phi$ -

equivalence even if  $S_?$  is not empty. In this case, both runs satisfying the formula and its negation originate in all states  $x \in \text{con}(X)$  of  $\mathcal{T}$  from all states  $X \in \alpha(S_? \cap S_{P0})$  of  $\mathcal{T}/\sim$  and  $S_?$  will not be made empty through additional refinement.

---

**Algorithm 6**  $X_{\hat{T}/\sim}^\phi = \text{FORMULA-GUIDED-REFINEMENT}(\mathcal{T}, \phi)$ : Compute satisfying region  $X_{\hat{T}/\sim}^\phi$  through formula guided refinement

---

Construct  $\mathcal{T}/\sim$ , such that  $\sim$  is observational equivalence

Construct deterministic BA  $\mathcal{B}_\phi$ , such that  $\mathcal{L}_{\mathcal{B}_\phi} = \mathcal{L}_\phi$

Build product automaton  $P = \mathcal{T}/\sim \otimes \mathcal{B}_\phi$

Initialize  $\hat{T}/\sim = \mathcal{T}/\sim$ ,  $\hat{P} = P$

**repeat**

  Compute  $S_Y$  and  $S_N$  in  $\hat{P}$

$S_? = S_{\hat{P}} \setminus (S_Y \cup S_N)$

**for all**  $(X, s) \in S_?$  **do**

**if**  $X$  not refined in  $\hat{T}/\sim$  and  $X$  is large enough **then**

$\hat{T}/\sim = \text{REFINE}(\hat{T}/\sim, X)$

**end if**

**if**  $X$  refined in  $\hat{T}/\sim$  **then**

$\text{UPDATE}(\hat{P}, (X, s))$

**end if**

**end for**

**until**  $\hat{P}$  not updated during previous iteration

**return**  $X_{\hat{T}/\sim}^\phi = \alpha(S_Y \cap S_{P0})$

---

The overall method discussed in this section is summarized in Algorithm 6. Since the regions of  $\mathcal{T}$  contain, in general, an infinite number of states, the algorithm might perform an infinite number of refinement steps. As in Algorithm 3, to ensure the algorithm terminates we refine a state only if it corresponds to a “large enough” region of  $\mathcal{T}$  (such a measure for  $\mathcal{T}_\Xi$  is discussed in Section 4.3). As a result,  $S_?$  might be nonempty when we force the algorithm to terminate, and we cannot guarantee an exact solution to Problem 2.

It is important to note that if a state  $X$  is refined in  $\hat{T}/\sim$ , not every state  $(X, s)$  is necessarily refined in  $\hat{P}$ . There is a one-to-many correspondence between the

refined product  $\hat{P}$  and the refined quotient  $\hat{T}/\sim$ , and the projection  $\alpha$  is of the type  $\alpha : S_{\hat{P}} \rightarrow 2^{\hat{Q}/\sim_\phi}$ . This might lead to a major reduction in the computational complexity of the solution: the product automaton  $\hat{P}$  after refinement might be significantly smaller than the product automaton  $\hat{T}/\sim \otimes \mathcal{B}$ , which we used for model checking in the approach from Section 4.1 (Algorithm 3).

The overall complexity of the solution can be further reduced by enforcing a specific order in which states of the product are refined. This optimization is based on the decomposition of the product automaton into maximal strongly connected components (SCCs). It can be shown that all states from a SCC belong to the same set  $S_Y$ ,  $S_N$ , or  $S_?$ . In addition, if all states from a SCC belongs to set  $S_Y$  or  $S_N$  then all states in other SCCs reachable from it must also belong to  $S_Y$  or  $S_N$ , respectively. Converting the product automaton to a SCC quotient graph provides two major improvements. First, it allows for a more efficient computation of sets  $S_Y$ ,  $S_N$ , and  $S_?$ . Second, refinement in the product automaton can be performed more efficiently in a bottom up manner, while unnecessary refinement is avoided.

### 4.3 Formal Analysis of PWA Systems

Through the embedding of PWA system  $\Xi$  (Equation (3.1)) into an infinite transition system  $\mathcal{T}_\Xi$  (Definition 20) discussed in Chapter 3, we reduced Problem 1 to Problem 2. In Section 4.1 we proposed a method for finding the largest region satisfying LTL formula  $\phi$  for a (possibly infinite) transition system through iterative refinement and model checking, which provided a conservative solution to Problem 2. In Section 4.2 we discussed conditions guaranteeing that an exact solution is obtained and proposed additional optimizations based on the construction of formula-equivalent finite quotients. In this section we discuss the implementation of the algorithms from Sections 4.1 and 4.2 for  $\mathcal{T}_\Xi$ .

In Chapter 3, we showed that when the matrix component of the parameters of  $\Xi$  was fixed, the quotient  $X_{T_{\Xi}/\sim}^{\phi}$  is finite and computable. Therefore,  $X_{T_{\Xi}/\sim}^{\phi}$  can be used in Algorithms 3 and 6 to provide a solution to Problem 1. When the matrix component of the parameters of  $\Xi$  was uncertain, we showed that a finite, over-approximation quotient  $\overline{T_{\Xi}/\sim}$  can be computed. Following from Equation 3.7,  $X_{\overline{T_{\Xi}/\sim}}^{\phi} \subseteq X_{T_{\Xi}/\sim}^{\phi} \subseteq X_{T_e}^{\phi}$ , which allows us to use the over-approximation quotient in Algorithms 3 and 6 to provide a (more conservative) solution to Problem 1.

Besides the computation of finite quotients, which was discussed in Chapter 3, an implementation of the `REFINE()` function (Algorithm 4) is required to complete the implementations of Algorithms 3 and 6 (note that both algorithms rely on the same refinement procedure). In order to implement the function `REFINE()` (Algorithm 4 or 5), given states  $l_1, l_2 \in Q_{\Xi}/\sim$  such that  $l_2 \in \text{Post}_{T_{\Xi}/\sim}(l_1)$ , we need to be able to construct a state  $l'$ , such that  $\text{con}(l') = \text{con}(l_1) \cap \text{Pre}_{T_{\Xi}}(\text{con}(l_2))$ . From the construction of  $T_{\Xi}/\sim$  in Chapter 3, for all states  $l \in Q_{\Xi}/\sim$ ,  $\text{con}(l) = \mathcal{X}_l$ , and therefore  $\text{con}(l') = \text{con}(l_1) \cap \text{Pre}_{T_{\Xi}}(\text{con}(l_2))$  can be equivalently written as  $\text{con}(l') = \mathcal{X}_{l_1} \cap \text{Pre}_{T_e}(\mathcal{X}_{l_2})$ . If the matrix component of the parameters is fixed (*i.e.*  $A_l = \mathcal{P}_l^A$  is a singleton for all  $l \in L$ ) and  $A_l$  are invertible for all  $l \in L$ , this intersection is computable as

$$\mathcal{X}_{l_1} \cap \text{Pre}_{T_{\Xi}}(\mathcal{X}_{l_2}) = \mathcal{X}_{l_1} \cap A_{l_1}^{-1}(\mathcal{X}_{l_2} \ominus \mathcal{P}_{l_1}^b),$$

where  $\ominus$  denotes the Minkowski difference (Definition 8). Therefore, `REFINE`( $T_e/\sim, X$ ) can be implemented using polyhedral operations. If a state  $l \in Q_{\Xi}/\sim$  is refined into states  $l_1, l_2$ , then  $\text{con}(l_1) \cup \text{con}(l_2) = \text{con}(l)$  and the procedure can be applied iteratively to  $l_1$  and  $l_2$ .

As already discussed in Section 4.1, if  $\mathcal{P}_l^b, l \in L$  are fixed then  $T_{\Xi}$  is deterministic and refinement can be performed more efficiently using Algorithm 5.

It is important to note that the results presented so far in this section can also

be used to demonstrate that the bisimulation algorithm (Algorithm 1) is computable for PWA systems when the matrix component of their parameter is fixed. This is not surprising, since our refinement procedure `REFINE()` was based on the application of the bisimulation algorithm locally at a particular state of the quotient.

Although a finite over-approximation quotient  $\overline{\mathcal{T}_\Xi/\sim}$  can be computed when the parameters of the system are uncertain,  $Pre_{\mathcal{T}_\Xi}()$  might be nonconvex, even when applied to a convex set. In this case, we use a  $2^N$ -tree inspired refinement approach, where each state is split along each dimension and transitions are recomputed using the over-approximation  $\overline{Post_{\mathcal{T}_\Xi}()}$  in Equation (3.2).

Finally, in order to implement Algorithm 3, we need to be able to decide if a state is “large enough” to undergo additional refinement. Given a state  $l$ , we compute the radius of the largest sphere inscribed in polytope  $con(l)$  and apply the refinement procedure only if it is larger than a certain predefined limit  $\epsilon$ . In other words, we apply the refinement procedure to state  $l$  only if  $r(\mathcal{X}_l) > \epsilon$ , where  $r(\mathcal{X}_l)$  is the radius of the Chebyshev ball of  $\mathcal{X}_l$  (Definition 9).

## 4.4 Complexity

Our method involves model checking of the finite quotient  $\mathcal{T}_\Xi/\sim$  at each step of the iterative procedure. Even though the worst case complexity of LTL model checking is exponential in the size of the formula, this upper limit is rarely reached in practice. We use an in-house model checker, which allows us to model check  $\mathcal{T}_\Xi/\sim$  from specific states only and perform computation (such as the construction of Büchi automata) only once instead of recomputing at each step.

The construction and refinement of finite quotients used in our approach is based on polyhedral operations, which also have an exponential upper bound. Therefore, the applicability of the method depends on controlling the number of states

as refinement progresses. When applied to a state  $X$ , the refinement procedure  $\text{REFINE}(\mathcal{T}_{\Xi}/\sim, X)$  can, in general, produce a maximum of  $2^k$  subsets, where  $k = |\text{Post}_{\mathcal{T}_{\Xi}/\sim}(X)|$  is the number of states reachable from state  $X$ . In the particular case when the parameters of the PWA system are fixed, only  $k$  subsets can be produced. To limit the explosion in the number of states in the quotient, we only refine states when this can improve the solution. Even so, due to its inherent complexity, this method is not suitable for the analysis of systems in high dimensions or when many iterations are required to find a solution. As expected, the method performs best if large portions of the state space can be characterized as satisfying the formula or its negation during earlier iterations.

For the case studies presented in Section 4.5 the computation required under 20 sec for the fixed parameter, two dimensional model ( $N = 2$ ) model and under 10 min for all the uncertain parameter ones, where the limit on refinement was set to  $\epsilon = 1$  and  $\epsilon = 5$ . For a three dimensional system ( $N = 3$ ) the computation required under 20 min where  $\epsilon = 5$ . All computation was performed on a 3.4GHz machine with 1GB of memory.

It is important to note that some specifications (such as  $\phi_1$  and  $\phi_2$  in Section 4.5) can be formulated as invariance and reachability properties and checked using more efficient tools [Kvasnica et al., 2004, Bemporad et al., 2000, Grieder, 2004]. However, such an approach does not apply to general LTL specifications (such as  $\phi_4$  in Section 4.5).

## 4.5 Implementation and Case Study

The algorithms presented in this chapter were implemented as a software tool for Formal Analysis of Piecewise Affine Systems FAPAS, which is freely downloadable at <http://hyness.bu.edu/software>. The tool is built under MATLAB, and uses



Table 4.1: Parameters of the PWA system used as a case study.

region ( $l$ )	$A_l$				$b_l$	
1	0.82	0.00	0.00	0.67	16.68	25.55
2	0.82	-0.37	0.00	0.67	19.37	25.55
3	0.82	0.00	0.00	0.67	3.08	25.55
4	0.82	0.00	-0.52	0.67	16.68	43.34
5	0.96	-0.39	-0.55	0.80	14.66	42.97
6	0.82	0.00	-0.52	0.67	3.08	47.65
7	0.82	0.00	0.00	0.67	16.68	2.47
8	0.82	-0.37	0.00	0.67	25.12	2.47
9	0.82	0.00	0.00	0.67	3.08	2.47

LTL2BA [Gastin and Oddoux, 2001] for the conversion of an LTL formula to a Büchi automaton and the MPT toolbox [Kvasnica et al., 2004] for polyhedral operations.

To demonstrate the methods proposed in the chapter, we present results from the analysis of a simple discrete time PWA model. We study a two dimensional ( $N = 2$ ) system that has a total of nine rectangular regions  $\mathcal{X}_1, \dots, \mathcal{X}_9$  labeled by  $L = \{1, 2, \dots, 9\}$ . The parameters for each region for an initial fixed parameter model (where  $\mathcal{P}_i^A, \mathcal{P}_i^b$  are singletons  $A_i, b_i$ , respectively) are given in Table 4.1<sup>1</sup>.

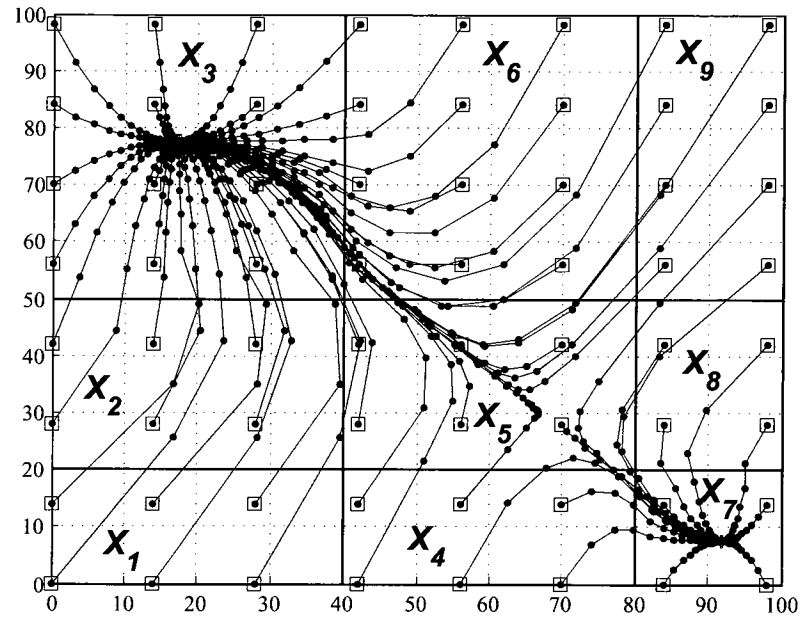
Under the fixed parameters, dynamics 3 and 7 have unique, asymptotically stable equilibria inside rectangles  $\mathcal{X}_3$  and  $\mathcal{X}_7$  (see Figure 4.3). An interesting problem is finding the regions of attraction for the two equilibria and exploring how those regions change when parameter uncertainty is introduced. By exploiting convexity properties of affine functions on polytopes, it can be easily proved that under the fixed parameters,  $\mathcal{X}_3$  and  $\mathcal{X}_7$  are invariants for dynamics 3 and 7, respectively. From this, we can immediately conclude that  $\mathcal{X}_3$  and  $\mathcal{X}_7$  are regions of attraction for the two equilibria. Therefore, our problem reduces to finding maximal regions satisfying LTL formulas  $\phi_1 = \text{"}\diamond\Box 3\text{"}$  and  $\phi_2 = \text{"}\diamond\Box 7\text{"}$ . In other words, we want to find maximal

<sup>1</sup>for notational simplicity, all parameters are presented as reshaped row vectors

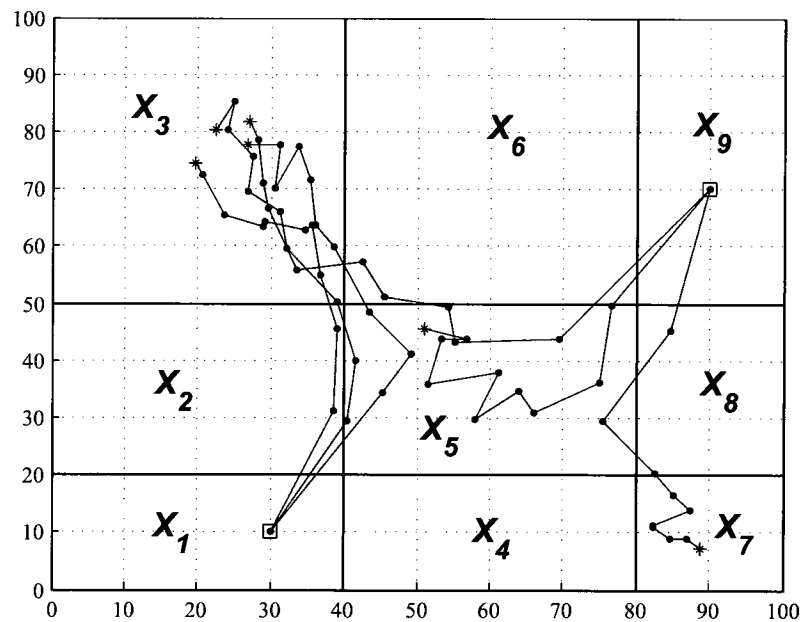
sets of initial conditions, from which trajectories will eventually reach regions  $\mathcal{X}_3$  or  $\mathcal{X}_7$  and stay there forever.

To explore how the sizes of the attractor regions change, hyper-rectangular parameter uncertainty was introduced in the model by allowing each component of the parameters  $A_l$  and  $b_l$  for region  $l \in L$  to vary in a range of size specified as a percentage of the fixed parameter value and centered around it (parameter components equal to 0 were also allowed to vary in a small range). Results from the computation with various levels of uncertainty are compared with the ones obtained under fixed parameters (Figure 4.4). Because of the rectangular initial partition of the state space,  $2^N$ -trees were used as an efficient splitting strategy for the uncertain parameter case. Our method identifies only an attracting region for the equilibria at  $\mathcal{X}_3$  for 5% uncertainty. As expected, increasing the level of uncertainty in the parameters decreases the size of the identified regions of state space (but a region identified at higher uncertainty is always a subset of the one identified at lower uncertainty).

Even if smaller limit  $\epsilon$  is used, under parameter uncertainty it is possible that a subset of the state space is never included in the identified regions- a property resulting from nondeterminism introduced in the embedding transitions system. Even though complete partitioning of the state space might not be possible, decreasing  $\epsilon$  provides further refinement and greater detail of the identified regions (initial iterations attempt to capture large satisfying (or violating) regions, while subsequent ones expand the solution less but provide greater resolution on its boundaries).



(a) Fixed parameter model



(b) Uncertain parameter model

Figure 4-3: Simulated trajectories of the PWA system used as a case study. Initial conditions are denoted by red squares.

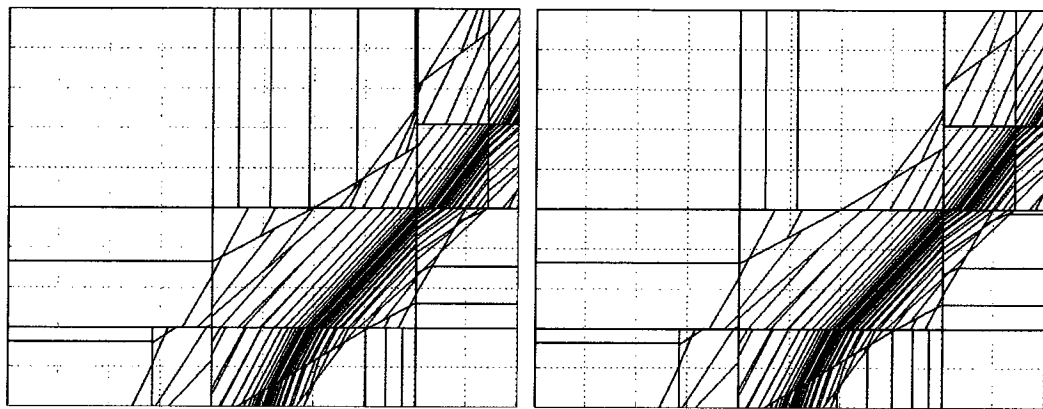
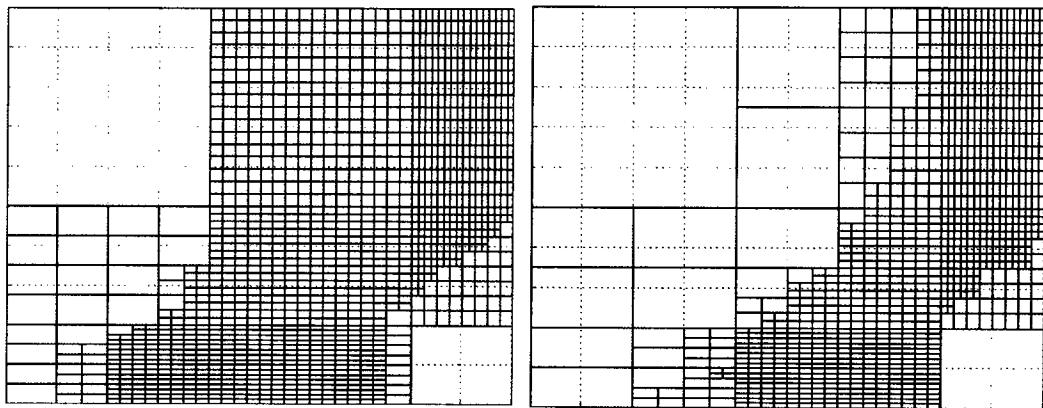
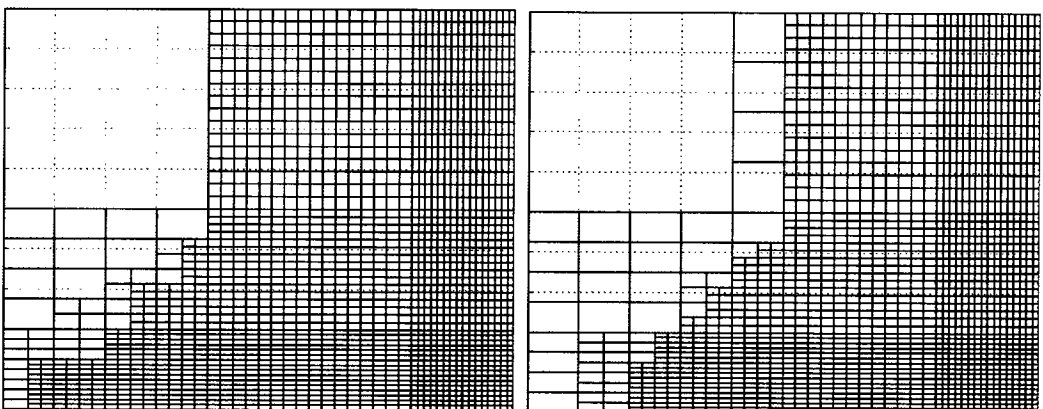
(a)  $\phi_1$  for fixed parameters(b)  $\phi_2$  for fixed parameters(c)  $\phi_1$  under 1% parameter uncertainty(d)  $\phi_2$  under 1% parameter uncertainty(e)  $\phi_1$  under 5% parameter uncertainty(f)  $\phi_2$  under 5% parameter uncertainty

Figure 4.4: Analysis results. Regions satisfying the formula are shown in green (lighter gray), while regions satisfying the negation are shown in red (darker gray).

## Chapter 5

# Parameter Synthesis for PWA systems

To study the satisfaction of LTL formulas by trajectories of PWA system  $\Xi$ , in Chapter 3 we defined the embedding transition system  $\mathcal{T}_\Xi$ , which was infinite. We showed that finite quotients of  $\mathcal{T}_\Xi$  can be computed through polyhedral operations and in Chapter 4 we used such quotients to develop an analysis procedure for PWA systems. We discussed how this procedure can be used to find a region of initial conditions of  $\Xi$ , from which all trajectories are guaranteed to satisfy the specification. Our procedure was also capable of handling PWA systems, where parameters were uncertain but restricted to polytopic ranges. We assumed that parameter uncertainty is inherent in the system and, in order to guarantee satisfaction, trajectories must satisfy the specification regardless of the (nondeterministic) choice of parameters from the allowed range.

In this chapter we take a different approach toward PWA systems with uncertain parameters. The parameters of the system are allowed to vary in predefined polytopic ranges as before, but in this chapter we assume that those ranges can be restricted further. In other words, we treat the parameter ranges not as an uncertainty inherent in the system, but rather as allowed ranges in which the system parameters can be tuned. Our goal is to find subsets of the allowed parameters for each region, such that the satisfaction of a specification can be guaranteed. Our approach involves the construction of discrete abstractions in the form of finite transition systems as described in Chapter 3.

We use a counterexample-guided strategy to identify and eliminate parameters leading to violating trajectories in the system. Unlike counterexample guided refinement [Clarke et al., 2003], where violating trajectories of a quotient are checked against a concrete model and, if spurious, removed by refinement, we use counterexamples to remove a set of (possibly spurious) violating transitions from the quotient. Then, we restrict the parameters of the systems based on the transitions removed from its quotient.

Before we formulate the main problem considered in this chapter, we make a small change in the notation used so far, in order to simplify the presentation. In Chapter 3 we defined the uncertain parameter sets for PWA system  $\Xi$  as  $\mathcal{P}_l^A$  and  $\mathcal{P}_l^b$ . For the discussion in this chapter, it will be helpful to consider a notation with only a single parameters set for each region. We define the set of parameters  $\mathcal{P}_l$  for each region  $l \in L$ , which is simply a polytope in  $\mathbb{R}^{(N^2+N)}$  that combines the dimensions of  $\mathcal{P}_l^A$  and  $\mathcal{P}_l^b$ . The linear functions  $A : \mathbb{R}^{(N^2+N)} \rightarrow \mathbb{R}^{N \times N}$  and  $b : \mathbb{R}^{(N^2+N)} \rightarrow \mathbb{R}^{N \times 1}$  take the first  $N^2$  and the last  $N$  components of  $p \in \mathbb{R}^{(N^2+N)}$  and form a  $N \times N$  matrix and  $N \times 1$  vector, respectively. The dynamics of the PWA system are then described by

$$\Xi : x_{k+1} = A(p)x_k + b(p), \quad x_k \in \mathcal{X}_l, \quad p \in \mathcal{P}_l, \quad l \in L, \quad k = 0, 1, \dots \quad (5.1)$$

We are now ready to formulate the main problem we consider in this chapter:

**Problem 3.** *Given a discrete-time piecewise affine system (3.1) and an LTL formula  $\phi$ , find sets of parameters  $\mathcal{P}_l^\phi \subseteq \mathcal{P}_l$  for each region  $l \in L$  and a set of initial states  $X^\phi$ , such that all trajectories of the system originating there satisfy the formula under all identified parameters.*

In other words, we are interested in excluding parameters from the allowed sets  $\mathcal{P}_l$  for each region  $l \in L$ , for which the formula is not satisfied. As it will become clear later, for each region  $l \in L$ , the solution will be in the form of a union of

disjoint open polytopes, which are subsets of the allowed polytope  $\mathcal{P}_l$ . In general, it is possible that for some states, no allowed parameters can be found, such that the satisfaction of the specification can be guaranteed. Therefore, the overall problem involves searching for both parameter ranges and initial states from which the satisfaction of the specification can be guaranteed (Problem 3).

To provide a solution to Problem 3, we first embed PWA system  $\Xi$  into the infinite transition system  $\mathcal{T}_\Xi$  and construct the finite, over-approximation quotient  $\overline{\mathcal{T}_\Xi/\sim}$  whose language includes the language of  $\mathcal{T}_\Xi$  as described in Chapter 3. We then use model checking to cut transitions from  $\overline{\mathcal{T}_\Xi/\sim}$  (see Section 5.1.1) and, correspondingly, cut sets of parameters from  $\Xi$  (see Section 5.1.2), until all its trajectories satisfy the formula. Alternatively, in Section 5.2, we propose a method for the direct construction of a bisimulation quotient. In both approaches, our method is conservative, as it will become clear later.

## 5.1 Counterexample-guided Parameter Synthesis

In Chapter 3 we showed that an over-approximation quotient  $\overline{\mathcal{T}_\Xi/\sim}$  can be constructed, and all operations involved are computable. In this section, we use LTL model checking to “cut” transitions from  $\overline{\mathcal{T}_\Xi/\sim}$  until we obtain a transition system  $\overline{\mathcal{T}_\Xi/\sim}^\phi$  satisfying the formula. Once a satisfying transition system is obtained, we modify the original PWA system  $\Xi$  (Equation (5.1)) by removing parameter values in such a way that the language of the new embedding transition system is included in the language of  $\overline{\mathcal{T}_\Xi/\sim}^\phi$ . In other words,  $\overline{\mathcal{T}_\Xi/\sim}^\phi$  becomes a quotient of the modified embedding which guarantees the satisfaction of the formula by PWA system  $\Xi$ .

### 5.1.1 Construction of satisfying quotients

We initialize the satisfying region as the entire set of states  $Q_{\Xi/\sim}$  and, using our LTL model checker described in [Kloetzer and Belta, 2006a], we start by searching for the shortest run<sup>1</sup> of  $\overline{\mathcal{T}_{\Xi/\sim}}$  satisfying the negation  $\neg\phi$  of LTL formula  $\phi$ . If such a counterexample is found, then we eliminate it by removing one of its transitions in  $\overline{\mathcal{T}_{\Xi/\sim}}$  and reiterate the process until we obtain the transition system  $\overline{\mathcal{T}_{\Xi/\sim}^\phi}$  satisfying  $\phi$ , when no more counterexamples can be generated.

In general, several different transitions are taken during the generation of a counterexample and removing any one of them will remove the counterexample from the language of the quotient. Selecting the best transition to remove at each step is non-trivial and, in general, it is not clear if removing a particular transition will lead to a solution (or to the best solution when more than one exists). In order to obtain more general results, we exhaustively generate all solutions by testing all transitions taken by a counterexample. This process can be seen as generating a tree, having the initial finite quotient as its root. Each child node in the tree represents a quotient that has the same set of states as the parent, but only a subset of its transitions. The children for each node are generated by removing one different transition, appearing in the shortest counterexample, from the parent.

When transitions are removed, a state of the quotient might become blocking, resulting in the appearance of finite words in its language. Since the semantics of LTL are defined only over infinite words, we make all blocking states unreachable by removing all their incoming transitions through an iterative procedure. This allows us to guarantee that blocking states are never reached. We must also guarantee that the system is not initialized in such blocking state and, therefore, we remove any

---

<sup>1</sup>An infinite run can be represented by a finite prefix and suffix, where the suffix is repeated an infinite number of times. The length of a run is the sum of the lengths of the prefix and suffix



blocking states from the satisfying set.

A leaf node in the tree represents a quotient for which computation stopped and no additional counterexamples can be generated. The quotients represented by such nodes satisfy the LTL formula, since their languages are nonempty (all initial states are non-blocking), do not contain finite words (no blocking states are reachable), and have an empty set of counterexamples.

### 5.1.2 Parameter synthesis

The finite quotient  $\mathcal{T}_{\Xi}/\sim$  is constructed so that it captures all possible transitions of the embedding  $\mathcal{T}_{\Xi}$ . By Definition 20, transitions are included in the embedding if and only if appropriate parameters for such a transition are allowed. Therefore, we can relate the transitions present in the finite quotient to sets of allowed parameters for the PWA system.

**Definition 24.** *Given two polytopes  $X$  and  $Y$  in  $\mathbb{R}^N$ , the set of parameters  $P^{X \not\sim Y}$ , for which the image of  $X$  does not have an intersection with  $Y$ , is defined as:*

$$P^{X \not\sim Y} = \{p \in \mathbb{R}^{(N^2+N)} \mid A(p)x + b(p) \notin Y \text{ for all } x \in X\} \quad (5.2)$$

**Proposition 6.** *Let  $X$  and  $Y$  be polytopes in  $\mathbb{R}^N$  given in  $V$ -representation as  $X = \text{Conv}\{v_1, \dots, v_m\}$  and  $H$ -representation as  $Y = \{x \in \mathbb{R}^N \mid c_i^T x + d_i < 0, i = 1, \dots, n\}$ , respectively. Then,*

$$\underline{P^{X \not\sim Y}} = \bigcup_{i=1}^n \{p \in \mathbb{R}^{(N^2+N)} \mid c_i^T (A(p)v_j + b(p)) + d_i > 0, \text{ for all } j = 1, \dots, m\}$$

*is an under-approximation of  $P^{X \not\sim Y}$  (i.e.,  $\underline{P^{X \not\sim Y}} \subseteq P^{X \not\sim Y}$ )*

*Proof.* Let  $p \in \bigcup_{i=1}^n \{p \in \mathbb{R}^{(N^2+N)} \mid c_i^T (A(p)v_j + b(p)) + d_i > 0, \text{ for all } j = 1, \dots, m\}$ . Then there exists an  $i$  such that  $c_i^T (A(p)v_j + b(p)) + d_i > 0$  for all  $j = 1, \dots, m$ . Then, for any  $x \in X$ ,  $c_i^T (A(p)x + b(p)) + d_i > 0$  and therefore  $A(p)x + b(p) \notin Y$  so  $p \in P^{X \not\sim Y}$ .  $\square$

In other words, a conservative under-approximation  $\underline{P^{X \neq Y}}$  of  $P^{X \neq Y}$  can be obtained as the union of polyhedral sets from the V-representation of  $X$  and the H-representation of  $Y$ .

We use the under-approximation from Proposition 6 to find sets of parameters for each region  $l \in L$ , such that, for each node of the tree described in Section 5.1.1, the corresponding PWA system is simulated by the quotient transition system at that node. Specifically, for two polytopes  $X \subseteq \mathcal{X}_l$  and  $Y$ , if the parameters in region  $l \in L$  are restricted to the set  $\mathcal{P}_l \cap \underline{P^{X \neq Y}}$ , then, by Proposition 6, the transition  $x \xrightarrow{e} y$  will not appear in the embedding  $\mathcal{T}_{\Xi}$ , for any  $x \in X$  and  $y \in Y$ . This means that, in the corresponding quotient, the transition  $X \xrightarrow{\Xi, \sim} Y$  will not exist.

Because of the over-approximation used in the construction of the quotient, a spurious transition might appear in place of a deleted one ( $(X, Y) \in \overrightarrow{\Xi, \sim}$  but  $(X, Y) \notin \rightarrow_{\Xi, \sim}$ ). We prevent this by enforcing that a deleted transition never reappears in the quotient. Additionally, if a quotient refinement procedure is applied, the structure of the PWA system allows different polytopes to share the same sets of parameters. Therefore, it is possible that additional transitions are removed from the quotient besides the target one. To account for this, we reconstruct the quotient every time parameters are cut. If, during the removal of parameters, a set  $\mathcal{P}_l$  becomes empty, then we consider all polytopes from region  $l \in L$  as blocking states, and make them unreachable.

By restricting the parameters as described above, we can ensure that, at every node of the tree constructed in Section 5.1.1, the PWA system with restricted parameters is simulated by the quotient transition system at that node. As previously stated, the leaf nodes of the computation tree contain quotients satisfying the formula and their corresponding PWA systems provide a solution to Problem 3. Our solution to Problem 3 is summarized in Algorithm 7, which returns the set of all

satisfying quotients and their corresponding PWA systems (the leaf nodes of the computation tree).

---

**Algorithm 7**  $T_{sat} = \text{SYNTHESIZE-PARAMETERS}(\Xi)$ : Generate a set of PWA systems satisfying  $\phi$ .

---

```

1:  $T_{sat} = \emptyset$ 
2:  $\overline{T_{\Xi}}/\sim = \text{QUOTIENT}(\Xi)$ 
3:  $T_{all} = \{(\Xi, \overline{T_{\Xi}}/\sim)\}$ 
4: while  $T_{all} \neq \emptyset$  do
5:   for each pair  $(\Xi', T')$   $\in T_{all}$  do
6:      $T_{all} = T_{all} \setminus (\Xi', T')$ 
7:      $X_{T'}^{\phi} = \text{MODEL-CHECK}(T', Q', \phi)$ 
8:      $X_{T'}^{\neg\phi} = \text{MODEL-CHECK}(T', Q', \neg\phi)$ 
9:      $X_r = Q' \setminus (X_{T'}^{\phi} \cup X_{T'}^{\neg\phi})$ 
10:    if  $X_r = \emptyset$  and  $X_{T'}^{\phi} \neq \emptyset$  then
11:      Add  $(\Xi', T')$  to  $T_{sat}$ 
12:    else if  $X_r \neq \emptyset$  then
13:      generate the shortest counter-example  $c \in \mathcal{L}_{T'}(X_r)$  for formula  $\phi$ 
14:      for each transition  $X \rightarrow' Y$  of counterexample  $c$  do
15:        Find  $\mathcal{X}_l$  such that  $X \subseteq \mathcal{X}_l$ 
16:        Construct  $\Xi''$  from  $\Xi'$  by setting  $\mathcal{P}_l'' = \mathcal{P}_l' \cap \underline{P^{X \neq Y}}$ 
17:        Reconstruct quotient  $T''$  from  $\Xi''$ 
18:        Ensure no previously removed transitions reappear
19:        Make blocking states of  $T''$  unreachable
20:        Add  $(\Xi'', T'')$  to  $T_{all}$ 
21:      end for
22:    end if
23:  end for
24: end while
25: return  $T_{sat}$ 

```

---

Algorithm 7 is initialized with the pair  $(\Xi, \overline{T_{\Xi}}/\sim)$ , containing the PWA system and the finite quotient of its embedding transition system. At each step, a counterexample is generated and the parameters the PWA system are restricted in order to eliminate it, as previously described. In order to prevent unnecessary computation, we combine our method with model checking against both the formula and its negation. If initial states from which the formula is satisfied are found then all trajectories of the system

originating there satisfy the formula and restricting the parameters is unnecessary. If there exist initial states from which the negation of the formula is satisfied, then there are no satisfying trajectories originating there, so a solution will not be found by refining the transitions (and corresponding parameters).

Algorithm 7 is guaranteed to terminate, since it exhaustively follows a tree of size limited by the total number of transitions in the initial quotient. Moreover, given only the purely discrete problem of modifying a quotient to satisfy a formula by taking a subset of its transitions (Section 5.1.1), our approach is complete and guaranteed to return a solution when one exists. In the combined problem of transition and parameter removal, computation will still terminate but a potential solution might be missed due to the approximations. If a solution is found, however, it is guaranteed to be correct.

When it terminates, Algorithm 7 returns a (possibly empty) set  $T_{sat}$  of satisfying PWA systems and their finite quotients. Given a pair  $(\Xi^\phi, \mathcal{T}_{\Xi^\phi}/\sim) \in T_{sat}$ , the parameters of  $\Xi^\phi$  are the satisfying parameters  $\mathcal{P}_i^\phi$  from Problem 3. The satisfying set of initial conditions can be obtained as  $X^\phi = con(X_{\mathcal{T}_{\Xi^\phi}}^\phi/\sim)$  (*i.e.* the concretization of the set of states satisfying  $\phi$  in  $\mathcal{T}_{\Xi^\phi}$ ), which provides a solution to Problem 3.

Going back to the tree construction from Section 5.1.1, in general, our method implemented in Algorithm 7 will produce more than one solution (*i.e.*  $|T_{sat}| \geq 1$ ). Selecting the "best" solution is a non-trivial problem, and might depend on the application. For example, it is possible to introduce additional constraints such as the requirement that particular transitions are always present in the solution. Alternatively, the size of the satisfying regions of potential solutions can be compared. Finally, the total number of transitions of the solutions can be compared, since more reachable states with more transitions result in a richer language. In the case study presented at the end of this chapter, we chose the latter.

Both the number of states and transitions in the embedding  $\overline{T_e/\sim}$  contribute to the complexity of Algorithm 7. A high dimensional system with many regions of different dynamics and propositions would be embedded with a high number of states. This, together with the complexity of the LTL formula affects the time required to perform model checking on the system. The number of transitions in the original embedding, on the other hand, depends on the dynamics of the system and determines how many times model checking must be performed, since the execution of the algorithm follows a finite tree described in Section 5.1.1. As a result, Algorithm 7 can perform well even on high dimensional systems, as long as the total number of transitions is low.

## 5.2 Construction of Bisimulation quotients

In this section we show that if the parameters of PWA system  $\Xi$  (Equation (3.1)) are restricted to appropriate subsets, an exact finite bisimulation quotient can be constructed without extensive iterative computation. Subsequently, satisfiability of an LTL formula by the original PWA system can be proven through model checking. Of course, by limiting the sets of parameters, certain transitions might disappear from the system and, therefore, the richness of its language might be diminished.

**Definition 25.** *Given two polytopes  $X$  and  $Y$  in  $\mathbb{R}^N$ , the set of parameters for which the image of  $X$  is completely included in  $Y$  is defined as:*

$$P^{X \rightarrow Y} = \{p \in \mathbb{R}^{(N^2+N)} \mid A(p)x + b(p) \in Y \text{ for all } x \in X\} \quad (5.3)$$

**Proposition 7.** *Let  $X$  and  $Y$  be polytopes in  $\mathbb{R}^N$  given in the  $V$ -representation as  $X = \text{Conv}\{v_1, \dots, v_m\}$  and in the  $H$ -representation as  $Y = \{x \in \mathbb{R}^N \mid c_i^T x + d_i < 0, i = 1, \dots, n\}$ , respectively. Then*

$$P^{X \rightarrow Y} = \{p \in \mathbb{R}^{(N^2+N)} \mid c_i^T (A(p)v_j + b(p)) + d_i < 0, i = 1, \dots, n, j = 1, \dots, m\}$$

*Proof.* Let  $p \in P^{X \rightarrow Y}$ . Then  $c_i^T (A(p)x + b(p)) + d_i < 0$  for all  $x \in X, i = 1, \dots, n$ . Then,  $p \in \{p \in \mathbb{R}^{(N^2+N)} \mid c_i^T (A(p)v_j + b(p)) + d_i < 0, i = 1, \dots, n, j = 1, \dots, m\}$ .

Let  $p \in \{p \in \mathbb{R}^{(N^2+N)} \mid c_i^T(A(p)v_j + b(p)) + d_i < 0, i = 1, \dots, n, j = 1, \dots, m\}$ . Then  $c_i^T(A(p)v_j + b(p)) + d_i < 0$  for all  $i = 1, \dots, n$  and  $j = 1, \dots, m$ . Let  $x \in X, x = \sum_{j=1}^m \lambda_j v_j$ .

$$\begin{aligned} c_i^T(A(p)x + b(p)) + d_i &= c_i^T(A(p) \sum_{j=1}^m \lambda_j v_j + b(p)) + d_i \\ &= \sum_{j=1}^m \lambda_j c_i^T A(p)v_j + a_i^T b(p) + d_i = \sum_{j=1}^m \lambda_j c_i^T A(p)v_j + \sum_{j=1}^m \lambda_j c_i^T b(p) + \sum_{j=1}^m \lambda_j d_i \\ &= \sum_{j=1}^m \lambda_j (c_i^T(A(p)v_j + b(p)) + d_i) \end{aligned}$$

Since  $\lambda_j \geq 0$  then  $p \in P^{X \rightarrow Y}$ . □

In other words, the polyhedral set of parameters  $P^{X \rightarrow Y}$  can be computed immediately from the V-representation of  $X$  and the H-representation of  $Y$ .

**Proposition 8.** *If in each location  $l \in L$ , the parameters of the PWA system (3.1), are restricted to  $\mathcal{P}_l \cap (\bigcup_{i \in L} P^{\mathcal{X}_i \rightarrow \mathcal{X}_i})$ , then the quotient  $T_e / \sim$  is a bisimulation quotient, and is computable.*

*Proof.* The proof for bisimulation follows immediately from Definitions 13, 20 and Proposition 7. Transitions of the quotient, are given by  $(X, Y) \in \rightarrow_{e \sim}$  if and only if  $X \subseteq \mathcal{X}_l$  and  $\mathcal{P}_l \cap P^{X \rightarrow Y} \neq \emptyset$ . □

### 5.3 Implementation and Case Study

The algorithms presented in this chapter were implemented as a software tool for Parameter Synthesis for Piecewise Affine Systems PARSYPAS, which is freely downloadable at <http://hyness.bu.edu/software>. The tool is built under MATLAB, and uses our in-house LTL model checker described in [Kloetzer and Belta, 2006a], LTL2BA [Gastin and Oddoux, 2001] for the conversion of an LTL formula to a Büchi automaton, and the MPT toolbox [Kvasnica et al., 2004] for polyhedral operations.

We illustrate methods proposed in this chapter by analyzing the simple PWA system discussed in Chapter 4. The initial model is two dimensional ( $N = 2$ ) and

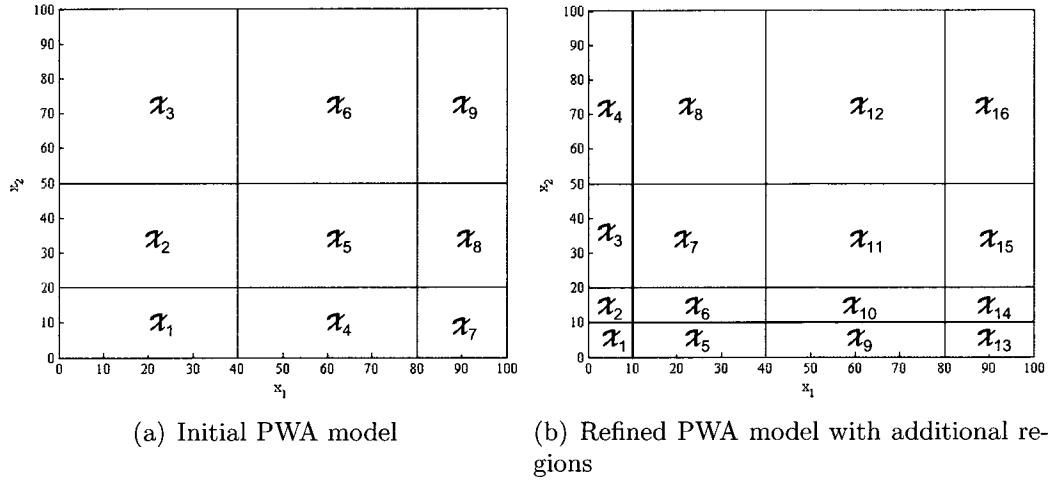


Figure 5.1: State partition for the PWA system used as a case study for parameter synthesis.

has a total of nine rectangular regions  $\mathcal{X}_1, \dots, \mathcal{X}_9$  labeled by  $L = \{1, 2, \dots, 9\}$  (Figure 5.1(a)). We are interested in analyzing the behavior of the system when it is initialized with low values for both state variables. Therefore, we refine the initial partition of the state space in order to include the required initial region  $\mathcal{X}_1$  in Figure 5.1(b). It is important to note that, while the refined partition contains more regions, some share the same parameter ranges. For example, regions  $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_5$ , and  $\mathcal{X}_6$  in Figure 5.1(b) are all subsets of  $\mathcal{X}_1$  in Figure 5.1(a) and therefore share the same set of parameters.

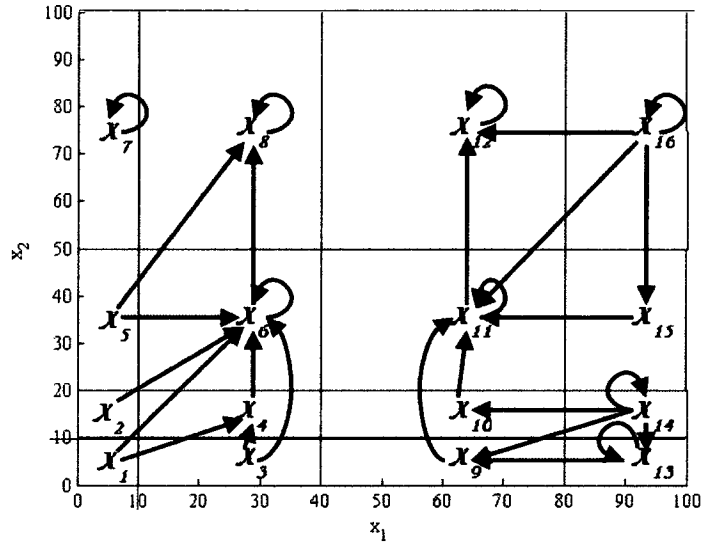
We assume hyper-rectangular parameter sets and, by using Proposition 7, we restrict the parameters for each region  $l \in L$  to subsets of  $P^{\mathcal{X}_l \rightarrow \mathcal{X}}$ , ensuring that  $\mathcal{X}$  is an invariant of all trajectories of the system. The parameter ranges of the system for all regions cannot be conveniently represented as text but are available at <http://hyness.bu.edu/software>.

First, we apply the method outlined in Section 5.2 in order to modify the parameters of the system and obtain a bisimulation quotient directly. The parameter ranges

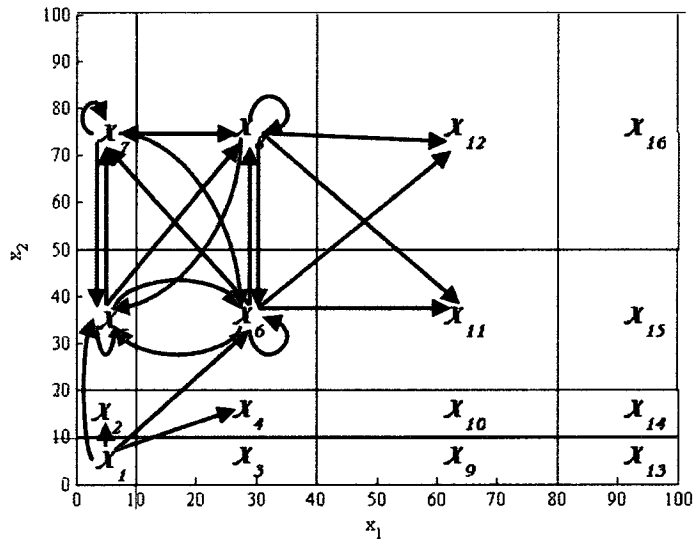
computed by the algorithm are available at <http://hyness.bu.edu/software> and a graphical representation of the resulting bisimulation quotient is shown in Figure 5.2(a). As expected, some transitions of the system are lost when parameters are restricted to smaller sets. Due to the language equivalence between the bisimulation quotient and the initial PWA system, the two systems can be used equivalently for model checking, which could provide an useful analysis tool.

Next, we apply the approach of Section 5.1.2 and find subsets of the parameters for each region of the system, such that the property  $\phi = 1 \wedge \diamond(7 \vee 8)$  is satisfied. In other words, we want to find parameters for the system such that all trajectories originating in region  $\mathcal{X}_1$  eventually reach region  $\mathcal{X}_7$  or  $\mathcal{X}_8$ . We use the same initial PWA model as before. During the execution of Algorithm 8 a number of transitions are removed from the quotient by removing appropriate sets of parameters of the system. The quotient corresponding to a solution, obtained as a leaf node in the computation tree (see Section 5.1.2) is shown in Figure 5.2(b). The set of satisfying initial states is  $\mathcal{X}_1$  and the regions of parameters for the PWA system obtained as a solution to Problem 3 are available at <http://hyness.bu.edu/software>. It is easy to see that under the remaining transitions (shown as black arrows in Figure 5.2(b)) all trajectories of the system originating in  $\mathcal{X}_1$  must visit either region  $\mathcal{X}_7$  or  $\mathcal{X}_8$ , which guarantees the satisfaction of the specification.





(a) Bisimulation Quotient



(b) Satisfying Simulation Quotient. Only transitions in the set of states reachable from the initial  $\mathcal{X}_1$  are shown.

Figure 5-2: Parameter Synthesis Results. Transitions are represented by black arrows. Transitions removed during the execution of the algorithm are shown in red.

## Chapter 6

# Formal Synthesis of Control Strategies for PWA systems

In Chapters 3, 4, and 5 we considered autonomous PWA systems. We showed that finite quotients of such systems can be computed and, based on this, developed methods for analysis and parameter synthesis from temporal logic specifications. Unlike the systems we discussed in previous chapters, which evolved autonomously, in this chapter we consider PWA control systems, which can be affected externally by applying a control signal. Then, it is possible to guarantee the satisfaction of a specification by trajectories of a PWA control system if an appropriate control signal is applied.

In this chapter we consider the following problem: given a PWA system with polytopic control constraints, and a specification in the form of a Linear Temporal Logic (LTL) formula over linear predicates in its state variables, find a set of initial states and a feedback control strategy such that all trajectories of the closed loop system originating in the initial set satisfy the formula (we formalize this problem in Section 6.2). We assume that the state of the system cannot be measured precisely and the applied inputs are corrupted by noise. Therefore, we seek control strategies that are robust both with respect to measured state and applied input. Our approach consists of two main steps. First, by partitioning the state and input spaces, we construct a finite abstraction of the PWA system in the form of a control transition

system, which extends results presented in Chapter 3 to a control framework. Second, by leveraging ideas and techniques from LTL model checking [Bianco and de Alfaro, 1995, Courcoubetis and Yannakakis, 1995] and Rabin games [Thomas, 2002], we develop an algorithm to generate a control strategy for the finite abstraction.

The remainder of this chapter is organized as follows. In Section 6.1 we extend some of the definitions from Chapter 2 for a control framework. The problem is formulated in Section 6.2, where we also present an overview of our approach. In Section 6.3 we define the control transition system and outline an algorithm for its computation. In Section 6.4, we present an approach for the synthesis of control strategies on finite transition systems (such as the control transition system) from temporal logic specifications. In Section 6.5 we use the results from Section 6.3 and Section 6.4 to formulate a solution to the main problem. In Section 6.6 we discuss a strategy for reducing the conservatism of the overall method by characterizing the stuttering behavior inherent in the construction of the control transition system. We outline the complexity associated with the proposed approach in Section 6.7 and describe its implementation and results from its application in Section 6.8.

## 6.1 Preliminaries

**Definition 26** (Transition System with Inputs). *A nondeterministic transition system with inputs is a tuple  $\mathcal{T} = (Q, \Sigma, \delta, O, o)$ , where  $Q$  and  $\Sigma$  are (possibly infinite) sets of states and inputs,  $\delta : Q \times \Sigma \rightarrow 2^Q$  is a (nondeterministic) transition map,  $O$  is a set of observations, and  $o : Q \rightarrow O$  is an observation map.*

A transition  $\delta(q, \sigma) = Q'$  indicates that, while the system is in state  $q$  it can make a transition to any state  $q' \in Q' \subseteq Q$  under input  $\sigma$ . We denote the set of inputs available at state  $q \in Q$  by  $\Sigma^q = \{\sigma \in \Sigma \mid \delta(q, \sigma) \neq \emptyset\}$ . A transition  $\delta(q, \sigma)$  is *deterministic* if  $|\delta(q, \sigma)| = 1$  and the transition system  $\mathcal{T}$  is deterministic if for all states  $q \in Q$  and all inputs  $\sigma \in \Sigma^q$ ,  $\delta(q, \sigma)$  is deterministic. Transition system  $\mathcal{T}$  is

*finite* if both its set of states  $Q$  and set of inputs  $\Sigma$  are finite.  $\mathcal{T}$  is *non-blocking* if, for every state  $q \in Q$ ,  $\Sigma^q \neq \emptyset$ . In this work, we consider only non-blocking transition systems.

An *input word* of the system is defined as an infinite sequence  $\sigma_0\sigma_1\sigma_2\dots \in \Sigma^\omega$ . A *trajectory* of  $\mathcal{T}$  produced by input word  $\sigma_0\sigma_1\sigma_2\dots$  and originating at state  $q_0 \in Q$  is an infinite sequence  $t = q_0q_1q_2\dots$  with the property that  $q_k \in Q$ , and  $q_{k+1} \in \delta(q_k, \sigma_k)$ , for all  $k \geq 1$ . We denote the set of all trajectories of  $\mathcal{T}$  originating at  $q$  by  $\mathcal{T}(q)$  (similarly, we use  $\mathcal{T}(Q') = \cup_{q' \in Q'} \mathcal{T}(q')$  to denote the set of all trajectories of  $\mathcal{T}$  originating in  $Q' \subseteq Q$ ).

For an arbitrary set of states  $Q' \subseteq Q$  and set of inputs  $\Sigma' \subseteq \Sigma$ , we define the set of states  $Post_{\mathcal{T}}(Q', \Sigma')$  that can be reached from  $Q'$  in one step by applying an input in  $\Sigma'$  as

$$Post_{\mathcal{T}}(Q', \Sigma') = \{q \in Q \mid \exists q' \in Q', \exists \sigma \in \Sigma', q \in \delta(q', \sigma)\} \quad (6.1)$$

**Definition 27.** A (*history dependent*) control function  $\Omega : Q^+ \rightarrow \Sigma$  for transition system  $\mathcal{T} = (Q, \Sigma, \delta, O, o)$  maps a finite, nonempty sequence of states to an input of  $\mathcal{T}$ . A control function  $\Omega$  and a set of initial states  $Q_0 \subseteq Q$  provide a control strategy for  $\mathcal{T}$ .

We denote a control strategy by  $(Q_0, \Omega)$ , the set of all trajectories of the closed loop system  $\mathcal{T}$  under the control strategy by  $\mathcal{T}(Q_0, \Omega)$ , and the set of all words produced by the closed loop  $\mathcal{T}$  as  $\mathcal{L}_{\mathcal{T}}(Q_0, \Omega)$ . For any trajectory  $q_0q_1q_2\dots \in \mathcal{T}(Q_0, \Omega)$  we have  $q_0 \in Q_0$  and  $q_{k+1} \in \delta(q_k, \sigma_k)$ , where  $\sigma_k = \Omega(q_1, \dots, q_k)$ , for all  $k \geq 1$ .

## 6.2 Problem Formulation and Approach

Let  $\mathcal{X}, \mathcal{X}_l, l \in L$  be a set of open polytopes in  $\mathbb{R}^N$ , where  $L$  is a finite index set, such that  $\mathcal{X}_{l_1} \cap \mathcal{X}_{l_2} = \emptyset$  for all  $l_1, l_2 \in L, l_1 \neq l_2$  and  $cl(\mathcal{X}) = \bigcup_{l \in L} cl(\mathcal{X}_l)$ , where  $cl(\mathcal{X}_l)$  denotes the closure of  $\mathcal{X}_l$ . A discrete-time piecewise affine (PWA) control system is

defined as:

$$\Xi : x_{k+1} = A_l x_k + B_l u_k + c_l, x_k \in \mathcal{X}_l, u_k \in \mathcal{U}, \quad (6.2)$$

where, at each time step  $k = 1, 2, \dots$ ,  $x_k \in \mathbb{R}^N$  is the state of the system,  $u_k$  is the input restricted to a polytopic set  $\mathcal{U} \subset \mathbb{R}^M$ , and  $A_l \in \mathbb{R}^{N \times N}$ ,  $B_l \in \mathbb{R}^{N \times M}$ ,  $c_l \in \mathbb{R}^N$  are the system parameters for mode  $l \in L$ .

We assume that at each time step  $k$  the exact state of the system ( $x_k \in \mathcal{X}_l, l \in L$ ) is unknown but we can observe the current mode  $l$ . The semantics of system (6.2) are given over words in  $L^\omega$ . Informally, a trajectory of the system produces a word by listing the index of the polytope visited at each step (*e.g.*, trajectory  $x_1 x_2 x_3 \dots$  satisfying  $x_1, x_2 \in \mathcal{X}_{l_1}$  and  $x_3 \in \mathcal{X}_{l_2}$  for some  $l_1, l_2 \in L$  will produce word  $l_1 l_1 l_2 \dots$ ). We assume that polytope  $\mathcal{X}$  is an invariant for all trajectories of the system (in Section 6.3.2 we will show that polyhedral control constraints guaranteeing this can be computed). Therefore, only infinite words are produced by the system and such words can be checked against the satisfaction of an LTL formula over  $L$  (see Section 2.6).

We consider the following problem:

**Problem 4.** *Given a PWA system (6.2) and an LTL formula  $\phi$  over  $L$ , find a control strategy, such that all trajectories of the closed loop system satisfy  $\phi$ .*

In order to complete the formulation of Problem 4, we need to formalize the definitions of a control strategy for a PWA system (6.2) and satisfaction of LTL formulas by trajectories of (6.2). As in Chapter 3, we do this through an embedding into a transition system, for which both LTL satisfaction (Section 2.6) and a control strategy (Definition 27) are clearly defined. Note that the embedding  $\mathcal{T}_e$  used in this chapter is different from the one defined in Chapter 3, since it is defined for a PWA control system.

**Definition 28.** (*Embedding transition system.*) *The embedding transition system*

$\mathcal{T}_e = (Q_e, \Sigma_e, \delta_e, O_e, o_e)$  for system (6.2) is defined as:

- $Q_e = \bigcup_{l \in L} \mathcal{X}_l$ ,
- $\Sigma_e = \mathcal{U}$ ,
- $\delta_e(x, u) = \{x'\}$  if and only if  $x' \in Q_e$  and there exist  $l \in L$  and  $u \in \mathcal{U}$  such that  $x \in \mathcal{X}_l$  and  $x' = A_l x + B_l u + c_l$ ,
- $O_e = L$ ,
- $o_e(x) = l$  if and only if  $x \in \mathcal{X}_l$ .

Note that the embedding transition system  $\mathcal{T}_e$  is always deterministic and non-blocking but both its set of states  $Q_e$  and set of inputs  $\Sigma_e$  are infinite.

**Definition 29.** Trajectories of system (6.2) originating in  $Q_0 \subseteq Q_e$  satisfy formula  $\phi$  if and only if  $\mathcal{T}_e(Q_0)$  satisfies  $\phi$ .

Problem 4 is an LTL control problem, where we seek a control strategy  $(Q_0, \Omega)$  for the infinite, deterministic transition system  $\mathcal{T}_e$ . In Section 6.4 we provide a solution to the problem of controlling a finite, nondeterministic transition system from LTL specifications (Problem 5). Then, the overall approach to Problem 4 involves the construction of a finite abstraction for  $\mathcal{T}_e$  (referred to as the control transition system  $\mathcal{T}_c$ ) such that a control strategy generated for  $\mathcal{T}_c$  can be adapted for  $\mathcal{T}_e$ .

Our approach to Problem 4 is schematically represented in Figure 6.1. We construct  $\mathcal{T}_c$  through a two step process. In the first step, by using the state equivalence relation induced by the polytopes from the definition of the PWA system, we construct a quotient  $\mathcal{T}_e/\sim$ , which has finitely many states but an infinite set of inputs. This part of the procedure is similar to the methods we used in previous chapters, with the exception that transition systems with inputs are considered. We then define an equivalence relation in the control space, which leads to the construction of the finite control transition system  $\mathcal{T}_c$ . This control transition system is then synchronized with a deterministic Rabin automaton  $\mathcal{R}$  that accepts the language satisfying

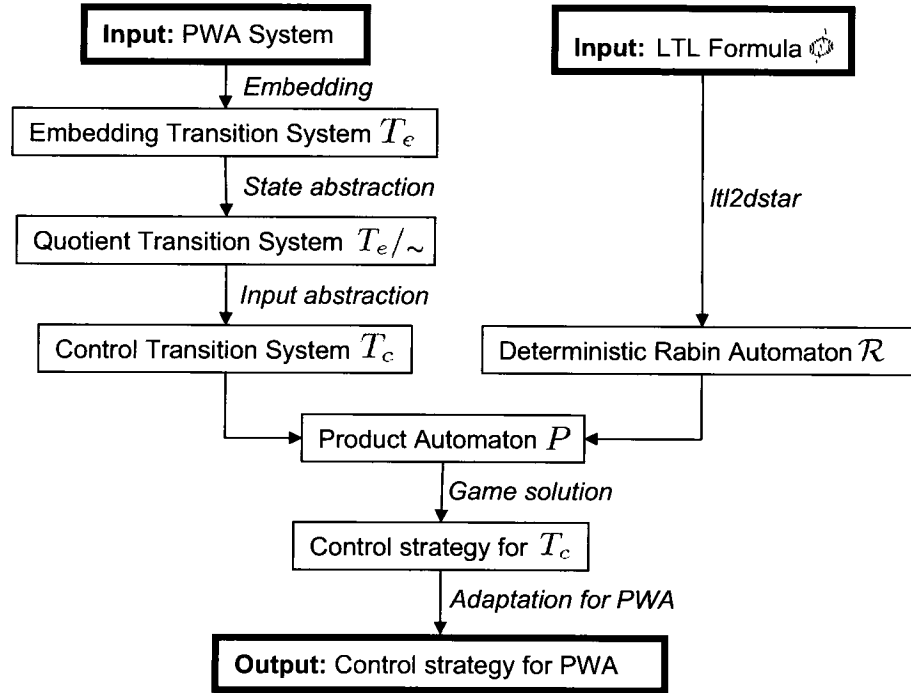


Figure 6-1: Illustration of our approach to formal synthesis of control strategies for PWA systems (Problem 4).

the formula to produce a product automaton  $\mathcal{P}$ . A game-theoretic approach is then used to generate a memoryless control strategy for  $\mathcal{P}$ , which is then translated to a (history dependent) control strategy for  $\mathcal{T}_c$ . The solution to Problem 4 is obtained by implementing the control strategy for  $\mathcal{T}_c$  as a feedback control automaton for the initial PWA system that reads the index of the region visited at each step and supplies the next input. As it will become clear later, our approach is robust in the sense that the closed loop system is guaranteed to satisfy the specification even when state measurements and applied inputs are perturbed.

The stuttering behavior (self transitions at a state of  $\mathcal{T}_c$  that can be taken infinitely in  $\mathcal{T}_c$  but do not correspond to real trajectories of  $\mathcal{T}_e$ ), which is also related to the well known Zeno behavior, was a source of conservativeness in the abstraction

procedure developed in Chapter 3. In this chapter, we explicitly characterize and deal with such behavior in order to reduce the conservativeness of our methods.

### 6.3 Control Transition System

In this section, we define the control transition system  $\mathcal{T}_c = (Q_c, \Sigma_c, \delta_c, O_c, o_c)$  for the embedding  $\mathcal{T}_e = (Q_e, \Sigma_e, \delta_e, O_e, o_e)$  (Definition 28) in Section 6.3.1 and present an algorithm for its computation in Section 6.3.2. In Section 6.5, we will show how  $\mathcal{T}_c$  is used to solve Problem 4.

#### 6.3.1 Construction

As in Chapter 3, the observation map  $o_e$  of  $\mathcal{T}_e$  induces an observational equivalence relation  $\sim$  over the set of states  $Q_e$ . However, the systems we considered before were autonomous and here, the quotient transition system  $\mathcal{T}_e/\sim = (Q_e/\sim, \Sigma_e, \delta_{e,\sim}, O_e, o_{e,\sim})$  induced by  $\sim$  has an infinite set of inputs  $\Sigma_e = \Sigma$ , which is preserved from  $\mathcal{T}_e$ . The set of transitions of  $\mathcal{T}_e/\sim$  are defined as  $l' \in \delta_{e,\sim}(l, u)$  if and only if there exist  $u \in \Sigma, x \in \mathcal{X}_l$  and  $x' \in \mathcal{X}'$  such that  $x' = \delta_e(x, u)$ . Note that  $\mathcal{T}_e/\sim$  is, in general nondeterministic, even though  $\mathcal{T}_e$  is deterministic. Indeed, for a state of the quotient  $l \in Q_e/\sim$  it is possible that different states  $x, x' \in \mathcal{X}_l$  have transitions in  $\mathcal{T}_e$  to states from different equivalence classes under the same input. The set of observations  $O_e = L$  of  $\mathcal{T}_e/\sim$  is preserved from  $\mathcal{T}_e$  and the observation map  $o_{e,\sim}$  is identity.

The transition map  $\delta_{e,\sim}$  can be related to the transitions of  $\mathcal{T}_e$  by using the *Post* operator defined in Equation (2.6):

$$\delta_{e,\sim}(l, u) = \{l' \in Q_e/\sim \mid \text{Post}_{\mathcal{T}_e}(\mathcal{X}_l, \{u\}) \cap \mathcal{X}_{l'} \neq \emptyset\}, \quad (6.3)$$

for all  $l \in Q_e/\sim$  and  $u \in \Sigma_e$ . For each state  $l \in Q_e/\sim$ , we define an equivalence relation  $\approx_l$  over the set of inputs  $\Sigma_e$  as  $(u_1, u_2) \in \approx_l$  iff  $\delta_{e,\sim}(l, u_1) = \delta_{e,\sim}(l, u_2)$ . In



other words, inputs  $u_1$  and  $u_2$  are equivalent at state  $l$  if they produce the same set of transitions in  $\mathcal{T}_e/\sim$ . Let  $U_l^{L'}$ ,  $l \in L$ ,  $L' \in 2^{Q_e/\sim}$  denote the equivalence classes of  $\Sigma_e$  in the partition induced by the equivalence relation  $\approx_l$ :

$$U_l^{L'} = \{u \in \Sigma_e \mid \delta_{e\sim}(l, u) = L'\} \quad (6.4)$$

Let  $c(U_l^{L'})$  be an input in  $U_l^{L'}$  such that  $\forall u \in \Sigma_e, d(c(U_l^{L'}), u) < \epsilon \Rightarrow u \in U_l^{L'}$  where  $d(u, u')$  denotes the distance between inputs  $u, u' \in \Sigma_e$  and  $\epsilon$  is a predefined parameter specifying the robustness of the control strategy. As it will become clear in Section 6.3.2,  $d(u, u')$  is the Euclidian distance in  $\mathbb{R}^M$  and  $c(U_l^{L'})$  can be computed as the center of a sphere inscribed in  $U_l^{L'}$ .

Initially, the states of  $\mathcal{T}_c$  are the observations of  $\mathcal{T}_e$  (*i.e.*,  $Q_c = L$ ). The set of inputs available at a state  $l \in L$  is  $\Sigma_c^l = \{c(U_l^{L'}) \mid L' \in 2^{Q_e/\sim}\}$  and the transition map is  $\delta_c(l, c(U_l^{L'})) = L'$ . In general, it is possible that at a given state  $l$ ,  $\Sigma_c^l = \emptyset$ , in which case state  $l$  is blocking. As it will become clear in Section 6.3.2, such states are removed from the system in a recursive procedure together with their incoming transitions and therefore  $Q_c \subseteq L$ . The set of observations and observation map of  $\mathcal{T}_c$  are preserved from  $\mathcal{T}_e/\sim$ , which completes the construction of the control transition system.

Following from the construction described so far, the control transition system  $\mathcal{T}_c$  is a finite transition system. In Section 6.5, we will show that a control strategy for  $\mathcal{T}_c$  can be adapted as a robust control strategy (with respect to knowledge of exact state and applied input) for the infinite  $\mathcal{T}_e$ . This will allow us to use  $\mathcal{T}_c$  as part of our solution to Problem 4.

### 6.3.2 Computation

Initially, the states of the control transition system  $\mathcal{T}_c$  are simply the labels  $L$  of the polytopes from the definition of the PWA system (Equation (6.2)). To complete its construction, we need to be able to compute the set of inputs  $\Sigma_c^l$  available at each state  $l \in Q_c$  and the transition map  $\delta_c$ , while eliminating the states that are unreachable in order to guarantee that  $\mathcal{T}_c$  remains non-blocking.

Given a polytope  $\mathcal{X}_l$  from the definition of the PWA system (Equation (6.2)), let

$$\Sigma^l = \{u \in \Sigma_e \mid \text{Post}_{\mathcal{T}_e}(\mathcal{X}_l, u) \subseteq \mathcal{X}\} \quad (6.5)$$

be the set of all inputs guaranteeing that all states from  $\mathcal{X}_l$  transit inside  $\mathcal{X}$  (*i.e.*,  $\Sigma^l$  is the set of all inputs allowed at  $l$ ). In other words, regardless which  $u \in \Sigma^l$  and  $x \in \mathcal{X}_l$  are selected,  $x$  will transit inside  $\mathcal{X}$  under  $u$  in  $\mathcal{T}_e$ . Then, in order to guarantee that  $\mathcal{X}$  is an invariant for all trajectories of the system (an assumption that we made in the formulation of Problem 4) it is sufficient to restrict the set of inputs  $\Sigma_c^l$  available at each state  $l \in Q_c$  to  $\Sigma_c^l \subseteq \Sigma^l$ .

**Proposition 9.** *Let  $\mathcal{X} = \{x \in \mathbb{R}^N \mid Hx < K\}$  be the  $H$ -representation of the polytope  $\mathcal{X}$  from the definition of the PWA system (6.2). Then,  $\Sigma^l$  is a polytope with the following  $H$ -representation:*

$$\Sigma^l = \{u \in \mathcal{U} \mid \forall v \in \mathcal{V}(\mathcal{X}_l), HB_l u < K - H(A_l v + c_l)\}, \quad (6.6)$$

where  $\mathcal{V}(\mathcal{X}_l)$  denotes the set of vertices of  $\mathcal{X}_l$ .

*Proof.* Note that the set defined in Equation (6.5) can be equivalently written as

$$\Sigma^l = \{u \in \mathcal{U} \mid \forall x \in \mathcal{X}_l, A_l x + B_l u + c_l \in \mathcal{X}\} \quad (6.7)$$

Let  $u \in \mathcal{U}$  such that  $\forall x \in \mathcal{X}_l, A_l x + B_l u + c_l \in \mathcal{X}$ . Then,

$$\begin{aligned} \forall x \in \mathcal{X}_l, H(A_l x + B_l u + c_l) < K &\Rightarrow \forall x \in \mathcal{X}_l, H B_l u < K - H(A_l x + c_l) \Rightarrow \\ &\Rightarrow \forall v \in \mathcal{V}(X_l), H B_l u < K - H(A_l v + c_l) \end{aligned}$$

Let  $u \in \mathcal{U}$  such that  $\forall v \in \mathcal{V}(X_l), H B_l u < K - H(A_l v + c_l)$ . Then,  $\forall v \in \mathcal{V}(X_l), A_l v + B_l u + c_l \in \mathcal{X}$ . Let  $m = |\mathcal{V}(X_l)|$ ,  $x = \sum_{i=1}^m \lambda_i v_i$ , where  $v_i \in \mathcal{V}(X_l)$ ,  $0 < \lambda_i < 1$  for all  $i = 1, \dots, m$  and  $\sum_{i=1}^m \lambda_i = 1$ . Then,

$$\begin{aligned} A_l x + B_l u + c_l = A_l \sum_{i=1}^m \lambda_i v_i + B_l u + c_l &= \sum_{i=1}^m \lambda_i (A_l v_i + B_l u + c_l) \in \mathcal{X} \Rightarrow \\ &\Rightarrow \forall x \in \mathcal{X}_l, A_l x + B_l u + c_l \in \mathcal{X} \end{aligned}$$

□

The set of states reachable from state  $l$  in  $\mathcal{T}_e/\sim$  under the allowed inputs is

$$Post_{\mathcal{T}_e/\sim}(l, \Sigma^l) = \{l' \in Q_e/\sim \mid Post_{\mathcal{T}_e}(\mathcal{X}_l, \Sigma^l) \cap \mathcal{X}_{l'} \neq \emptyset\} \quad (6.8)$$

and can be computed using polyhedral operations, since

$$Post_{\mathcal{T}_e}(\mathcal{X}_l, \Sigma^l) = A_l \mathcal{X}_l + B_l \Sigma^l + c_l. \quad (6.9)$$

Given a polytope  $\mathcal{X}_l$  from the definition of the PWA system (Equation (6.2)) and an arbitrary polytope  $\mathcal{X}'$ , let

$$U^{\mathcal{X}_l \rightarrow \mathcal{X}'} = \{u \in \Sigma_e \mid Post_{\mathcal{T}_e}(\mathcal{X}_l, u) \cap \mathcal{X}' \neq \emptyset\} \quad (6.10)$$

denote the set of all inputs under which  $\mathcal{T}_e$  can make a transition from a state in  $\mathcal{X}_l$  to a state inside  $\mathcal{X}'$ . Equivalently, applying any input  $u \in \mathcal{U}$ ,  $u \notin U^{\mathcal{X}_l \rightarrow \mathcal{X}'}$  guarantees that  $\mathcal{T}_e$  will not make a transition inside  $\mathcal{X}'$ , from any state in  $\mathcal{X}_l$ . The following proposition states that  $U^{\mathcal{X}_l \rightarrow \mathcal{X}'}$  is a polyhedral set that can be computed from the V- (vertex) and H- (hyperplane) representations of  $\mathcal{X}_l$  and  $\mathcal{X}'$ :

**Proposition 10.** *Let  $H$  and  $K$  be the matrices in the H-representation of the following polytope:*

$$\{\hat{x} \in \mathbb{R}^N \mid \exists x \in \mathcal{X}_l, A_l x + \hat{x} + c_l \in \mathcal{X}'\} \quad (6.11)$$

*Then  $U^{\mathcal{X}_l \rightarrow \mathcal{X}'}$  is a polytope with the following H-representation:*

$$U^{\mathcal{X}_l \rightarrow \mathcal{X}'} = \{u \in \mathcal{U} \mid HB_l u < K\} \quad (6.12)$$

*Proof.* The set defined in Equation (6.11) is a polytope with the following V-representation:

$$\text{hull}\{v' - (Av + c) \mid v \in \mathcal{V}(\mathcal{X}_l), v' \in \mathcal{V}(\mathcal{X}')\} \quad (6.13)$$

Let  $\exists x \in \mathcal{X}_l$  such that  $A_l x + \hat{x} + c_l \in \mathcal{X}'$ . Let  $m = |\mathcal{V}(\mathcal{X}_l)|$  and  $x = \sum_{i=1}^m \lambda_i v_i$ , where  $0 < \lambda_i < 1$  for all  $i = 1, \dots, m$  and  $\sum_{i=1}^m \lambda_i = 1$ . Let  $n = |\mathcal{V}(\mathcal{X}')|$  and  $x' = \sum_{j=1}^n \mu_j v'_j$ , where  $0 < \mu_j < 1$  for all  $j = 1, \dots, n$  and  $\sum_{j=1}^n \mu_j = 1$ . Then,

$$\begin{aligned} A_l \sum_{i=1}^m \lambda_i v_i + \hat{x} + c_l = \sum_{j=1}^n \mu_j v'_j &\Rightarrow \hat{x} = \sum_{j=1}^n \mu_j v'_j - A_l \sum_{i=1}^m \lambda_i v_i - c_l = \\ = \sum_{i=1}^m \sum_{j=1}^n \lambda_i \mu_j (v'_j - (A_l v_i + c_l)) &\Rightarrow \hat{x} \in \text{hull}\{v' - (Av + c) \mid v \in \mathcal{V}(\mathcal{X}_l), v' \in \mathcal{V}(\mathcal{X}')\} \end{aligned}$$

Let  $\hat{x} = \sum_{i=1}^m \sum_{j=1}^n \nu_{ij} (v'_j - (A_l v_i + c_l))$ , where  $0 < \nu_{ij} < 1, i = 1, \dots, m, j = 1, \dots, n$  and  $\sum_{i=1}^m \sum_{j=1}^n \nu_{ij} = 1$ . Let  $\lambda_i = \sum_{j=1}^n \nu_{ij}$  and  $\mu_j = \sum_{i=1}^m \nu_{ij}$ . Of course,  $0 < \lambda_i < 1$  for all  $i = 1, \dots, m$ ,  $0 < \mu_j < 1$  for all  $j = 1, \dots, n$  and  $\sum_{i=1}^m \lambda_i = \sum_{j=1}^n \mu_j = \sum_{i=1}^m \sum_{j=1}^n \nu_{ij} = 1$ . Then, for  $x = \sum_{i=1}^m \lambda_i v_i$  and  $x' = \sum_{j=1}^n \mu_j v'_j$  we have  $A_l x + \hat{x} + c_l = x'$  and therefore  $\exists x \in \mathcal{X}_l$  such that  $A_l x + \hat{x} + c_l \in \mathcal{X}'$ . To conclude the proof of Proposition 10, let  $H, K$  be the matrices in the H-representation of the set defined in Equation (6.11) and note that the set defined in Equation (6.10) can be equivalently written as

$$U^{\mathcal{X}_l \rightarrow \mathcal{X}'} = \{u \in \mathcal{U} \mid \exists x \in \mathcal{X}_l, A_l x + B_l u + c_l \in \mathcal{X}'\} \quad (6.14)$$

□

**Proposition 11.** *Given a state  $l \in Q_c$  and a set of states  $L' \in 2^{Q_c}$ , the set  $U_l^{L'}$  from Equation (6.4) can be computed as follows:*

$$U_l^{L'} = \bigcap_{l' \in L'} U^{\mathcal{X}_l \rightarrow \mathcal{X}_{l'}} \setminus \bigcup_{l'' \notin L'} U^{\mathcal{X}_l \rightarrow \mathcal{X}_{l''}} \quad (6.15)$$

*Proof.* From Equation (6.3) and (6.4) we have

$$\begin{aligned}
U_l^{L'} &= \{u \in \Sigma_e \mid \forall l' \in L', \text{Post}_{\mathcal{T}_e}(\mathcal{X}_l, u) \cap \mathcal{X}_{l'} \neq \emptyset, \\
&\quad \forall l'' \notin L', \text{Post}_{\mathcal{T}_e}(\mathcal{X}_l, u) \cap \mathcal{X}_{l''} = \emptyset\} = \\
&= \{u \in \Sigma_e; \mid \forall l' \in L', \text{Post}_{\mathcal{T}_e}(\mathcal{X}_l, u) \cap \mathcal{X}_{l'} \neq \emptyset\} \setminus \\
&\quad \{u \in \Sigma_e \mid \exists l'' \notin L', \text{Post}_{\mathcal{T}_e}(\mathcal{X}_l, u) \cap \mathcal{X}_{l''} \neq \emptyset\} = \\
&= \bigcap_{l' \in L'} U^{\mathcal{X}_l \rightarrow \mathcal{X}_{l'}} \setminus \bigcup_{l'' \notin L'} U^{\mathcal{X}_l \rightarrow \mathcal{X}_{l''}}
\end{aligned}$$

□

We can guarantee that if a state  $l'$  is not reachable from state  $l$  in  $\mathcal{T}_e/\sim$  (*i.e.*,  $l' \notin \text{Post}_{\mathcal{T}_e/\sim}(l, \Sigma^l)$ ) then  $U^{\mathcal{X}_l \rightarrow \mathcal{X}_{l'}} = \emptyset$  and therefore,  $U_l^{L'} = \emptyset$  if  $L' \not\subseteq \text{Post}_{\mathcal{T}_e/\sim}(l, \Sigma^l)$  and otherwise the computation in Equation (6.15) reduces to

$$U_l^{L'} = \bigcap_{l' \in L'} U^{\mathcal{X}_l \rightarrow \mathcal{X}_{l'}} \setminus \bigcup_{l'' \in \text{Post}_{\mathcal{T}_e/\sim}(l, \Sigma^l) \setminus L'} U^{\mathcal{X}_l \rightarrow \mathcal{X}_{l''}} \quad (6.16)$$

A non-empty input region  $U_l^{L'}$  is in general nonconvex but can always be represented as a finite union of open polytopes (see Equation (6.16)). In order to guarantee the robustness of the control strategy (as described in Section 6.3.1) we only include input sets that are "large enough" (*i.e.*,  $r(U_l^{L'}) > \epsilon$ , where  $\epsilon$  is a predefined robustness parameter and  $r(\cdot)$  is the radius of the Chebyshev ball (Definition 9). Note that in general this approach might be conservative, since a sphere inscribed in a union of polytopes from  $U_l^{L'}$  might have a larger radius. Following from the results presented in this section, the control transition system  $\mathcal{T}_c$  can be computed using polyhedral operations only (the computation is summarized in Algorithm 8).

## 6.4 LTL Control for Finite Transition Systems as a Rabin Game

In this section, we consider the following problem:

---

**Algorithm 8**  $\mathcal{T}_c = \text{CONTROL-TS}(\Xi, \epsilon)$ : Construct control transition system  $\mathcal{T}_c$

---

**Input:** PWA system (Equation (6.2)) and robustness parameter  $\epsilon$

**Output:** Control transition system  $\mathcal{T}_c = (Q_c, \Sigma_c, \delta_c, O_c, o_c)$

```

1:  $Q_c := L$ 
2: for each  $l \in Q_c$  do
3:    $\Sigma_c^l := \Sigma^l$  (Equation (6.5))
4:   compute  $\text{Post}_{\mathcal{T}_c/\sim}(l, \Sigma_c^l)$  (Equation (6.8))
5:   for each  $L' \subseteq \text{Post}_{\mathcal{T}_c/\sim}(l, \Sigma_c^l)$  do
6:     compute  $U_l^{L'}$  (Equation (6.16))
7:     if  $r(U_l^{L'}) > \epsilon$  then
8:       include input  $c(U_l^{L'})$  in  $\Sigma_c^l$ 
9:       include transition  $\delta_c(l, c(U_l^{L'})) = L'$ 
10:    end if
11:  end for
12:  if  $\Sigma_c^l = \emptyset$  then
13:    recursively make state  $l$  unreachable and set  $Q_c := Q_c \setminus l$ 
14:  end if
15: end for
16:  $\Sigma_c = \bigcup_{l \in Q_c} \Sigma_c^l$ 
17: return  $\mathcal{T}_c$ 

```

---

**Problem 5.** *Given a finite (nondeterministic) transition system  $\mathcal{T}$  (Definition 11) and an LTL formula  $\phi$ , find a control strategy (Definition 27), such that all trajectories of the closed loop system satisfy  $\phi$ .*

Problem 5 is quite general, and can be seen as the dual control formulation of the classical LTL model checking problem [Baier and Katoen, 2008, Clarke et al., 1999]. In [Kloetzer and Belta, 2008a], a solution to Problem 5 for the particular case when the LTL formula can be translated into a deterministic Büchi automaton was proposed. This solution was conservative, since not all LTL formulas can be translated into deterministic Büchi automata (*e.g.*,  $\diamond\Box\phi$  for any LTL formula  $\phi$ ). In this section, we extend the previous results and formulate a new method, capable of handling specifications over the full LTL. We first reformulate Problem 5 as a Rabin game and then adapt the solution of the Rabin game as a control strategy for

$\mathcal{T}$ . As it will become clear later, the control strategy takes the form of a “feedback automaton”, which reads the current state of  $\mathcal{T}$  and produces the control input to be applied at that state.

Given a finite transition system  $\mathcal{T} = (Q, \Sigma, \delta, O, o)$  and an LTL formula  $\phi$  over  $O$  we can translate  $\phi$  into a deterministic Rabin automaton  $\mathcal{R} = (S, S_0, O, \delta_{\mathcal{R}}, F)$  (see Section 2.7) and construct the *product automaton*  $\mathcal{P} = (S_{\mathcal{P}}, S_{\mathcal{P}0}, \Sigma, \delta_{\mathcal{P}}, F_{\mathcal{P}})$  of  $\mathcal{T}$  and  $\mathcal{R}$ , where

- $S_{\mathcal{P}} = Q \times S$  is the set of states,
- $S_{\mathcal{P}0} = Q \times S_0$  is the set of initial states,
- $\Sigma$  is the input alphabet,
- $\delta_{\mathcal{P}} : S_{\mathcal{P}} \times \Sigma \rightarrow 2^{S_{\mathcal{P}}}$  is the transition map, where  $\delta_{\mathcal{P}}((q, s), \sigma) = \{(q', s') \in S_{\mathcal{P}} \mid q' \in \delta(q, \sigma), \text{ and } s' = \delta_{\mathcal{R}}(s, o(q))\}$ , and
- $F_{\mathcal{P}} = \{(Q \times G_1, Q \times B_1), \dots, (Q \times G_n, Q \times B_n)\}$  is the Rabin acceptance condition.

The product automaton is a nondeterministic Rabin automaton with the same input alphabet  $\Sigma$  as  $\mathcal{T}$ . Each accepting run  $\rho_{\mathcal{P}} = (q_0, s_0)(q_1, s_1) \dots$  of  $\mathcal{P}$  can be projected into a trajectory  $q_0 q_1 \dots$  of  $\mathcal{T}$ , such that the word  $o(q_0)o(q_1) \dots$  is accepted by  $\mathcal{R}$  (*i.e.*, satisfies  $\phi$ ) and vice versa [Vardi and Wolper, 1986]. This allows us to reduce Problem 5 to finding a control strategy  $(W_{\mathcal{P}0}, \pi_{\mathcal{P}})$  for  $\mathcal{P}$ , such that each run of the closed loop  $\mathcal{P}$  satisfies the Rabin acceptance condition  $F_{\mathcal{P}}$ <sup>1</sup>. This problem can be viewed as a *Rabin game* played on the product automaton between two players – a protagonist and an adversary. A play is initiated in a state of the product automaton

---

<sup>1</sup>Control strategies for Rabin automata (such as  $\mathcal{P}$ ) are defined by a set of initial states  $W_{\mathcal{P}0}$  and a control function  $\pi_{\mathcal{P}}$  as for transition systems (Definition 27). The behavior of the closed loop system is analogous.

and proceeds according to the following rule: at each state, the protagonist chooses an input to be applied and the adversary determines the next state to be visited under this input (*i.e.*, the adversary resolves nondeterministic transitions). A play produces an infinite sequence of states (*i.e.*, a run) and it is won by the protagonist if the produced run satisfies the Rabin condition. A solution to the Rabin game is a control strategy: a control function determining moves of the protagonist and a set of initial states called winning region, such that each play under the strategy is won by the protagonist. Since winning strategies for Rabin games are memoryless [Emerson, 1985], the control function is simply a map  $\pi_{\mathcal{P}} : S_{\mathcal{P}} \rightarrow \Sigma$ .

Rabin games can be solved by standard algorithms [Piterman and Pnueli, 2006, Horn, 2005]. In this paper we follow the approach from [Horn, 2005], which can be adapted to deal with stuttering behavior as we will explain in Section 6.6. The basic step of the recursive algorithm is attractor construction. A protagonist's (or adversary's) attractor of a set  $S' \subseteq S_{\mathcal{P}}$  is defined as a set of states from which the protagonist (or the adversary, respectively) can enforce a visit to  $S'$ .

**Definition 30.** (*Protagonist's direct attractor*). *The protagonist's direct attractor of  $S'$ , denoted by  $A_{\mathcal{P}}^1(S')$ , is the set of all states  $s \in S_{\mathcal{P}}$ , such that there exists an input  $\sigma$  satisfying  $\delta_{\mathcal{P}}(s, \sigma) \subseteq S'$ .*

In other words, the protagonist can enforce a visit to  $S'$  from a state  $s \in A_{\mathcal{P}}^1(S')$  by applying an input  $\sigma$  regardless of the following adversary's choice.

**Definition 31.** (*Adversary's direct attractor*). *The adversary's direct attractor of  $S'$ , denoted by  $A_{\mathcal{S}}^1(S')$ , is the set of all states  $s \in S_{\mathcal{P}}$ , such that there exists a state  $s' \in \delta_{\mathcal{P}}(s, \sigma) \cap S'$  for each input  $\sigma \in \Sigma^s$ .*

In other words, the adversary can enforce a visit to  $S'$  from a state  $s \in A_{\mathcal{S}}^1(S')$  regardless of which input  $\sigma$  has been chosen by the protagonist.

The protagonist's attractor of  $S'$  can be computed iteratively via computation of converging sequence  $A_{P_0}^*(S') \subseteq A_{P_1}^*(S') \subseteq A_{P_2}^*(S') \subseteq \dots$ , where  $A_{P_0}^*(S') = S'$  and



$A_{P_{i+1}}^*(S') = A_P^1(A_{P_i}^*(S'))$ . Intuitively  $A_{P_i}^*(S')$  is the set from which a visit to the set  $S'$  can be enforced by the protagonist in at most  $i$  steps. The adversary's attractor is computed analogously.

By solving the Rabin game outlined above we generate a control strategy  $(W_{\mathcal{P}_0}, \pi_{\mathcal{P}})$  for  $\mathcal{P}$ . In order to complete the solution to Problem 5, we adapt  $(W_{\mathcal{P}_0}, \pi_{\mathcal{P}})$  as a control strategy  $(Q_0, \Omega)$  for  $\mathcal{T}$ . Although the control function  $\pi_{\mathcal{P}}$  was memory-less,  $\Omega$  is history dependent and takes the form of a feedback control automaton  $\mathcal{C} = (S, S_0, Q, \tau, \pi, \Sigma)$ , where the set of states  $S$  and initial states  $S_0$  are inherited from  $\mathcal{R}$ , the set of inputs  $Q$  is the set of states of  $\mathcal{T}$ , and the memory update function  $\tau : S \times Q \rightarrow S$  and output function  $\pi : S \times Q \rightarrow \Sigma$  are defined as

$$\begin{aligned} \tau(s, q) &\in \delta_{\mathcal{R}}(s, o(q)) \text{ if } (q, s) \in W_{\mathcal{P}}, \tau(s, q) = \perp \text{ otherwise} \\ \pi(s, q) &= \pi_{\mathcal{P}}((q, s)) \text{ if } (q, s) \in W_{\mathcal{P}}, \pi(s, q) = \perp \text{ otherwise} \end{aligned}$$

The set of initial states  $Q_0$  of  $\mathcal{T}$  is given by  $\alpha(W_{\mathcal{P}_0})$ , where  $\alpha : S_{\mathcal{P}} \rightarrow Q$  is the projection from states of  $\mathcal{P}$  to  $Q$ . The control function  $\Omega$  is given by  $\mathcal{C}$  as follows: for a sequence  $q_0 \dots q_n$ ,  $q_0 \in Q_0$ , we have  $\Omega(q_0 \dots q_n) = \sigma$ , where  $\sigma = \pi(s_n, q_n)$ ,  $s_{i+1} = \tau(s_i, q_i)$ , and  $q_{i+1} \in \delta(q_i, \pi(s_i, q_i))$ , for all  $i \in \{0, \dots, n\}$ . It is easy to see that the product automaton of  $\mathcal{T}$  and  $\mathcal{C}$  will have the same states as  $\mathcal{P}$  but contains only transitions of  $\mathcal{P}$  closed under  $\pi_{\mathcal{P}}$ . Then, all trajectories of the closed loop  $\mathcal{T}(Q_0, \Omega)$  satisfy  $\phi$  and therefore  $(Q_0, \Omega)$  is the solution to Problem 5. In Section 6.5, we will use this result together with the construction of the control transition system  $\mathcal{T}_c$  described in Section 6.3 in order to provide a solution to Problem 4.

## 6.5 LTL Control of PWA Systems

In Section 6.3 we defined the control transition system  $\mathcal{T}_c$  as a finite abstraction of the infinite  $\mathcal{T}_e$  and showed that it can be computed using polyhedral operations.

In Section 6.4, we presented an approach for controlling finite transition systems (such as  $\mathcal{T}_c$ ) from specifications given as LTL formulas (Problem 5). In this section, we show that a control strategy generated for  $\mathcal{T}_c$  can be adapted to the infinite  $\mathcal{T}_e$ , while the satisfaction of LTL formulas by the closed loop systems is preserved, which completes the solution to Problem 4.

**Definition 32.** *A control strategy  $(Q_0^c, \Omega^c)$  for  $\mathcal{T}_c$  can be translated into a control strategy  $(Q_0, \Omega)$  for  $\mathcal{T}_e$  as follows. The initial set  $Q_0^c \subseteq Q_c$  gives the initial set  $Q_0 = \bigcup_{l \in Q_0^c} \mathcal{X}_l \subseteq Q_e$ . Given a finite sequence of states  $q_0 \dots q_k$  where  $q_0 \in Q_0$ , the control function is defined as  $\Omega(q_0 \dots q_k) = \Omega^c(o_e(q_0) \dots o_e(q_k))$ .*

**Proposition 12.** *Given a control strategy  $(Q_0^c, \Omega^c)$  for  $\mathcal{T}_c$  translated as a control strategy  $(Q_0, \Omega)$  for  $\mathcal{T}_e$ ,  $\mathcal{L}_{\mathcal{T}_e}(Q_0, \Omega) \subseteq \mathcal{L}_{\mathcal{T}_c}(Q_0^c, \Omega^c)$ , which implies that if  $\mathcal{T}_c(Q_0^c, \Omega^c)$  satisfies an arbitrary LTL formula  $\phi$ , then so does  $\mathcal{T}_e(Q_0, \Omega)$ .*

*Proof.* Transition systems  $\mathcal{T}_c$  and  $\mathcal{T}_e$  have the same set of observations and therefore the same LTL formula can be interpreted over both systems. Let  $q_0 \in Q_0$  be an initial state for  $\mathcal{T}_e$ . Its observation is  $q_0^c = o_e(q_0)$ , which is a satisfying initial state for  $\mathcal{T}_c$  (*i.e.*,  $q_0^c \in Q_0^c$ ). The next input to be applied in  $\mathcal{T}_c$  is given by the control function ( $u_0^c = \Omega(q_0^c) \in \Sigma_c \subset \Sigma_e$ ) and we can guarantee that regardless which input  $u_0 \in \Sigma_e$  such that  $d(u_0, u_0^c) < \epsilon$  is applied in  $\mathcal{T}_e$ , we have  $\delta_e(q_0, u_0) \in \delta_c(q_0^c, u_0^c)$ . This shows that a finite fragment of a word in  $\mathcal{T}_e(Q_0, \Omega)$  is also a finite fragment of a word in  $\mathcal{T}_c(Q_0^c, \Omega^c)$  and the rest of the proof follows by induction.  $\square$

The overall solution to Problem 4 consists of constructing the control transition system  $\mathcal{T}_c$  (Section 6.3), finding a satisfying control strategy for  $\mathcal{T}_c$  (Section 6.4) and adapting it to the original  $\mathcal{T}_e$ , or equivalently PWA system (Definition 32), which from Proposition 12 guarantees the correctness of the solution. It is important to note that a control strategy generated using this approach is robust with respect to knowledge of the exact state of the system (*i.e.*, the control strategy depends on the observation of a state rather than the state itself). In addition, the control strategy is robust with respect to perturbations in the applied input bounded by  $\epsilon$ , which can be used as a tuning parameter.

## 6.6 Conservatism and Stuttering Behavior

In Section 6.4 we described a solution to the problem of controlling a finite and possibly nondeterministic transition system from LTL specifications (Problem 5). In order to generate a control strategy for an infinite transition system such as  $\mathcal{T}_e$  (Problem 4) we described the construction of a finite control abstraction  $\mathcal{T}_c$  in Section 6.3. However, due to *spurious trajectories* (*i.e.*, trajectories of  $\mathcal{T}_c$  not present in  $\mathcal{T}_e$ ) we cannot guarantee that a control strategy will be found for  $\mathcal{T}_c$  even if one exists for  $\mathcal{T}_e$  and therefore, the overall method is conservative. In Chapter 4, we eliminated spurious trajectories through state refinement but the states of  $\mathcal{T}_c$  cannot be refined. Indeed, a control strategy cannot differentiate between states  $x_1, x_2 \in Q_e$  when  $o(x_1) = o(x_2)$  and therefore the states of  $\mathcal{T}_c$  must satisfy  $o_c(l_1) = o_c(l_2)$  if and only if  $l_1 = l_2$  for all  $l_1, l_2 \in Q_c$ . In the following, we present an alternative approach for reducing this conservatism.

Inspired by the abstraction of *stutter* steps described in [Baier and Katoen, 2008], in this paper we characterize only a specific class of spurious trajectories, which we introduce through an example (Figure 6-2). Assume that a constant input  $uuu\dots$  produces a trajectory  $x_1x_2x_3x_4\dots$  in  $\mathcal{T}_e$  where  $o(x_1) = l_1, o(x_2) = o(x_3) = l_2, o(x_4) = l_3$  (Figure 6-2-A). The corresponding word  $l_1l_2l_2l_3\dots$  is a trajectory of  $\mathcal{T}_c$  (*i.e.*,  $l_1, l_2, l_3 \in Q_c$ ) and from the construction described in Section 6.3 it follows that  $l_2 \in \delta_c(l_1, u)$  and  $\{l_2, l_3\} \subseteq \delta_c(l_2, u)$  (Figure 6-2-B). Then, there exists a trajectory of  $\mathcal{T}_c$  that remains infinitely in state  $l_2 \in Q_c$  under input  $u$ , which is not necessarily true for  $\mathcal{T}_e$ . Such spurious trajectories do not affect the correctness of a control strategy but increase the overall conservativeness of the method. We address this by characterizing *stuttering inputs*, which guarantee that the system will leave a state eventually, rather than in a single step, and using this additional information during the construction of the control strategy for  $\mathcal{T}_c$ .

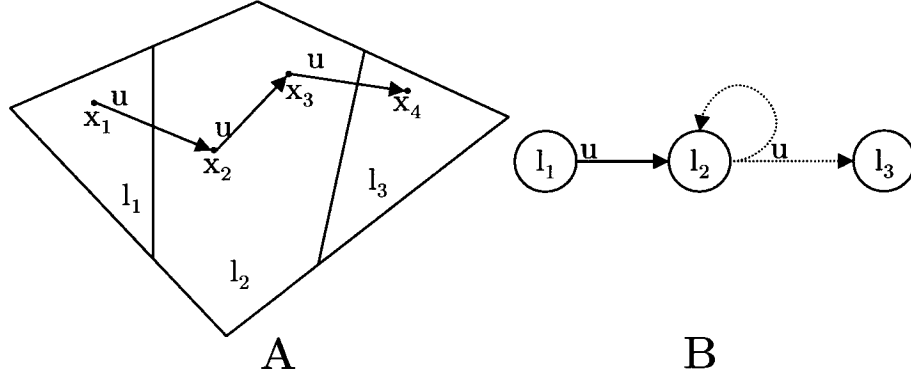


Figure 6-2: A trajectory remaining forever in state  $l_2$  exists in the finite abstraction (B), although such a behavior is not necessarily possible in the concrete system (A)

**Definition 33.** (*Stuttering inputs*). Given a state  $l \in Q_c$  and a set of states  $L' \in 2^{Q_c}$ , the set of inputs  $U_l^{L'}$  is stuttering if and only if  $l \in L'$  and for all input words  $u_0u_1\dots$ , where  $u_i \in U_l^{L'}$ , there exists a finite  $k > 1$  such that the trajectory  $x_0x_1\dots$  produced in  $\mathcal{T}_e$  by the input word satisfies  $o(x_i) = l$  for  $i = 1, \dots, k-1$  and  $o(x_k) = l' \in L', l' \neq l$ .

Using Definition 33 we identify a stuttering subset  $\Sigma_c^{ls} \subseteq \Sigma_c^l$  of the inputs available at a state  $l \in Q_c$ . Let  $u = c(U_l^{L'}) \in \Sigma_c^l$  for some  $L' \in 2^{Q_c}$  be an input of  $\mathcal{T}_c$  computed as described in Section 6.3. Then  $u \in \Sigma_c^{ls}$  if and only if  $U_l^{L'}$  is stuttering. Note that a transition  $\delta_c(l, u) = L'$  from a state  $l \in Q_c$  where  $u$  is stuttering is always nondeterministic (*i.e.*,  $|L'| > 1$ ) and contains a self loop (*i.e.*,  $l \in L'$ ) but the self loop cannot be taken infinitely in a row (*i.e.*, a trajectory of  $\mathcal{T}_e$  cannot remain infinitely in region  $\mathcal{X}_l$  under input word  $uuu\dots$ ). An input  $u \in \Sigma_c^{lu} = \Sigma_c^l \setminus \Sigma_c^{ls}$  induces a transition  $\delta_c(l, u) = L'$  where: (1) when  $L' = \{l\}$  trajectories of  $\mathcal{T}_c$  and  $\mathcal{T}_e$  produced by input word  $uuu\dots$  remain infinitely in state  $l$  and region  $\mathcal{X}_l$ , respectively, (2) when  $l \notin L'$  trajectories of  $\mathcal{T}_c$  and  $\mathcal{T}_e$  leave state  $l$  and region  $\mathcal{X}_l$ , respectively in one step under input  $u$ , (3) when  $\{l\} \subset L'$  trajectories of  $\mathcal{T}_e$  produced by input word  $uuu\dots$  can potentially remain in region  $\mathcal{X}_l$  infinitely. Although in case (3) it is also possible that trajectories of  $\mathcal{T}_e$  produced by input word  $uuu\dots$  leave region  $\mathcal{X}_l$  in finite time, we

have to be conservative in order to guarantee the correctness of the control strategy.

Note that our treatment of stuttering is different from [Baier and Katoen, 2008] in two aspects. First, we require that  $\mathcal{T}_c$  leaves a state after a finite number of transitions are taken under the same stuttering input and therefore an infinite stutter cycle is never possible. Second, we identify a set of stuttering inputs rather than constructing  $\mathcal{T}_c$  as a time abstract system. While we only characterize spurious infinite self loops (*i.e.*, cycles of length 1), in general, it is possible that cycles of arbitrary length are spurious in  $\mathcal{T}_c$ . Considering higher order cycles is computationally challenging and decreases the conservativeness of the approach only for very specific cases, while spurious self loops are commonly produced during the construction of  $\mathcal{T}_c$  and can be identified or constructed through polyhedral operations as described in Propositions 13 and 14.

**Proposition 13.** *Given a state  $l \in Q_c$  and a set of states  $L' \in 2^{Q_c}$ , input region  $U_l^{L'}$  is stuttering if and only if  $l \in L'$  and  $0 \notin \text{hull}\{(A_l - I)v_x + B_l v_u + c_l \mid \forall v_x \in \mathcal{V}(\mathcal{X}_l), \forall v_u \in \mathcal{V}(U_l^{L'})\}$ , where  $\text{hull}$  denotes the convex hull,  $\mathcal{V}(\cdot)$  is the set of vertices and  $I$  is the identity matrix.*

*Proof.* ( $\Rightarrow$ ) Let  $0 \in \text{hull}\{(A_l - I)v_x + B_l v_u + c_l \mid \forall v_x \in \mathcal{V}(\mathcal{X}_l), \forall v_u \in \mathcal{V}(U_l^{L'})\}$ . Then, there exists  $x \in \mathcal{X}_l, u \in U_l^{L'}$  such that  $A_l x + B_l u + c_l = x$  and a trajectory of the system produced by applying input sequence  $uuu\dots$  and starting at  $x$  remains forever inside  $\mathcal{X}$ . Therefore, from Definition 33,  $U_l^{L'}$  is not stuttering.

( $\Leftarrow$ ) Let  $0 \notin \text{hull}\{(A_l - I)v_x + B_l v_u + c_l \mid \forall v_x \in \mathcal{V}(\mathcal{X}_l), \forall v_u \in \mathcal{V}(U_l^{L'})\}$ . From the separating hyperplane theorem it follows that there exists  $a \in \mathbb{R}^N$  such that, for all  $z \in \text{hull}\{(A_l - I)v_x + B_l v_u + c_l \mid \forall v_x \in \mathcal{V}(\mathcal{X}_l), \forall v_u \in \mathcal{V}(U_l^{L'})\}$ ,  $a^T z > 0$ . Then, any trajectory of the system originating in  $\mathcal{X}_l$  and produced by input word  $u_1 u_2 u_3 \dots$ , where  $u_i \in U_l^{L'}$  will have a positive displacement along the direction of  $a^T$  at every step. Since  $\mathcal{X}_l$  is bounded, all trajectories will leave it in a finite number of steps and, therefore,  $U_l^{L'}$  is stuttering.  $\square$

Proposition 13 provides a computational characterization of stuttering input regions. In general, however, it is possible that an input region  $U_l^{L'}$  cannot be identified

as stuttering but a stuttering subset  $\hat{U}_l^{L'} \subset U_l^{L'}$  can be identified. Then, if such a subset is "large enough" (i.e.,  $r(\hat{U}_l^{L'}) > \epsilon$ ) it can be used in  $\mathcal{T}_c$  and allow more general control strategies. In Proposition 14 we describe the computation of such stuttering subsets.

**Proposition 14.** *Given an arbitrary  $a \in \mathbb{R}^N$ , the input region  $\hat{U}_l^{L'} = \{u \in U_l^{L'} \mid \forall v \in \mathcal{V}(\mathcal{X}_l), a^T B_l u > -a^T (A_l - I_N)v - c_l\}$ , where  $l \in L'$  is always stuttering.*

*Proof.*

$$\begin{aligned} \hat{U}_l^{L'} &= \{u \in U_l^{L'} \mid \forall v_x \in \mathcal{V}(\mathcal{X}_l), a^T B_l u > -a^T (A_l - I)v_x - c_l\} = \\ &= \{u \in U_l^{L'} \mid \forall v_x \in \mathcal{V}(\mathcal{X}_l), a^T ((A_l - I)v_x + B_l u + c_l) > 0\} \Rightarrow \\ &\Rightarrow 0 \notin \text{hull}\{(A_l - I)v_x + B_l v_u + c_l \mid \forall v_x \in \mathcal{V}(\mathcal{X}_l), \forall v_u \in \mathcal{V}(\hat{U}_l^{L'})\}, \end{aligned}$$

which, from Proposition 13, guarantees that  $\hat{U}_l^{L'}$  is stuttering.  $\square$

Although Proposition 14 is valid for an arbitrary  $a \in \mathbb{R}^N$ , the volume of the stuttering subset  $\hat{U}_l^{L'} \subset U_l^{L'}$  depends on  $a$ . Since only "large enough" input regions are considered in  $\mathcal{T}_c$  (see Algorithm 8),  $a$  should be chosen in such a way that the radius  $r(\hat{U}_l^{L'})$  is maximized. This problem is beyond the scope of this work but a possible (suboptimal) solution involves the uniform sampling of rotation groups as discussed in [Mitchell, 2007]. In our current implementation discussed in Section 6.8, only the characterization provided in Proposition 13 is used to identify stuttering input sets.

The algorithm from [Horn, 2005] from Section 6.4 can be adapted to handle the additional information about stuttering inputs captured in  $\mathcal{T}_c$ , while the correctness and completeness of the control strategy computation for the product automaton  $\mathcal{P}$  is still guaranteed.  $\mathcal{P}$  is constructed as in Section 6.4 and therefore it naturally inherits the partitioned input set  $\Sigma_c^l = \Sigma_c^{ls} \cup \Sigma_c^{lu}$  for each state  $l \in Q_c$ . Going back to the Rabin game interpretation of the control problem discussed in Section 6.4, we need to account for the fact that the adversary cannot take transitions under the same

stuttering input infinitely many times in a row. As a result, the construction of the control strategy is still performed using Horn's algorithm and only the computations of the direct attractors (Definitions (30) and (31)) are modified as follows.

Let  $l \in Q_c$  and  $u \in \Sigma_c^{ls}$  be a state and a stuttering input of  $\mathcal{T}_c$  (Definition 33). We are interested in *edge*  $(s, u, s')$  of transition  $\delta_{\mathcal{P}}(s, u) = S'$ , where  $\alpha(s) = l$  and  $s' \in S'$ . Edge  $(s, u, s')$  is called *u-nontransient* edge if  $\alpha(s) = \alpha(s') = l$  and *transient* otherwise. Note that, even though  $(l, u, l)$  is a self loop in  $\mathcal{T}_c$ ,  $(s, u, s')$  is not necessarily a self loop in  $\mathcal{P}$ . In addition, since there is at most one self loop at a state  $l \in Q_c$  and  $\mathcal{R}$  is deterministic, there is at most one *u-nontransient* edge leaving state  $s$ .

We refer to a sequence of edges  $(s_1, u_1, s_2)(s_2, u_2, s_3) \dots (s_{n-1}, u_{n-1}, s_n)$ , where  $s_i \neq s_j$  for any  $i, j \in \{1, \dots, n\}$  as a *simple path*, and to a simple path  $(s_1, u_1, s_2) \dots (s_{n-1}, u_{n-1}, s_n)$  followed by  $(s_n, u_n, s_1)$  as a *cycle*. We can observe that any sequence of *u-nontransient* edges (*i.e.* a run of the product automaton, or its finite fragment) is of one of the following shapes: a cycle (called a *u-nontransient cycle*), a lasso shape (a simple path leading to a *u-nontransient cycle*), or a simple path ending at a state where the input  $u$  is not available at all. Informally, the existence of a stuttering self loop in a state  $l$  under input  $u$  in  $\mathcal{T}_c$  means that this self loop cannot be followed infinitely many times in a row. Similarly, any *u-nontransient cycle* in the product graph cannot be followed infinitely many times in a row without leaving it. This leads us to the new definitions of protagonist's and adversary's direct attractor.

**Definition 34.** (*Protagonist's direct attractor*). *The protagonist's direct attractor of  $S'$ , denoted by  $A_{\mathcal{P}}^1(S')$ , is the set of all states  $s \in S_{\mathcal{P}}$ , such that there exists an input  $u$  satisfying*

- (1)  $\delta_{\mathcal{P}}(s, u) \subseteq S'$ , or
- (2)  $s$  lies on a *u-nontransient cycle*, such that each state  $s'$  of the cycle satisfies that  $s'' \in S'$  for all transient edges  $(s', u, s'')$

In other words, the protagonist can enforce a visit to  $S'$  also by following a  $u$ -nontransient cycle finitely many times and eventually leaving it to  $S'$ .

**Definition 35.** (*Adversary's direct attractor*). *The adversary's direct attractor of  $S'$ , denoted by  $A_S^1(S')$ , is the set of all states  $s \in S_{\mathcal{P}}$ , such that for each input  $u$  there exists a state  $s'$  such that*

- (1)  $s' \in \delta_{\mathcal{P}}(s, u) \cap S'$ , and
- (2)  $s'$  does not lie on a  $u$ -nontransient cycle

In other words, the adversary cannot enforce a visit to  $S'$  via an edge of a  $u$ -nontransient cycle. This edge can be taken only finitely many times in row and eventually different edge under input  $u$  has to be chosen.

By identifying stuttering inputs during the construction of the control transition system  $\mathcal{T}_c$  (Propositions 13 and 14) and modifying the approach from Section 6.4 to handle this additional information during the construction of a control strategy for  $\mathcal{T}_c$  (Definitions (34) and (35)), we can reduce the conservatism associated with the overall method. Even so, our solution to Problem 4 remains conservative but it is important to note that the only source of conservativeness is the construction of  $\mathcal{T}_c$  - the solution to the LTL control problem for  $\mathcal{T}_c$  is complete.

## 6.7 Complexity

As our proposed solution to Problem 4 consists of (1) the construction of the control transition system  $\mathcal{T}_c$  and (2) the generation of a control strategy for  $\mathcal{T}_c$ , the overall computational complexity is the cumulative complexity of the two parts. The computation of  $\mathcal{T}_c$  involves enumerating all subsets of  $L$  at any element of  $L$ , which gives  $O(|L| \cdot 2^{|L|})$  iterations in the worst case, although in practice this can be reduced (see Equation (6.16)). At each iteration, polyhedral operations are performed, which scale exponentially with  $N$ , the size of the continuous state space. The character-



ization of stuttering inputs described in this chapter checks each element from  $\Sigma_c$  through polyhedral operations.

It is also possible to reduce the size of  $\mathcal{T}_c$  after it is initially constructed without sacrificing solutions. More “nondeterminism” available at a state does not result in more winning strategies for the algorithm described in Section 6.4, while at the same time unnecessarily increases the complexity of the method. Formally, let  $u_1 = c(U_l^{L'_1})$  and  $u_2 = c(U_l^{L'_2})$  where  $L'_1, L'_2 \in 2^Q, L'_1 \subseteq L'_2$  be inputs of  $\mathcal{T}_c$  available at state  $l \in Q_c$  (i.e.,  $\{u_1, u_2\} \subseteq \Sigma_c^l$ ). If input  $u_2$  is used in a control strategy, then the specification is satisfied regardless of which state  $l' \in L'_2$  is visited in the next step. Clearly, the same holds for input  $u_1$  since  $L'_1$  is a subset of  $L'_2$  but keeping both inputs is unnecessary. Therefore, at each state  $l \in Q_c$  we set  $\Sigma_c^{ls} = \Sigma_c^{ls} \setminus u_2$  if  $u_1, u_2 \in \Sigma_c^{ls}$  or  $\Sigma_c^{lu} = \Sigma_c^{lu} \setminus u_2$  if  $u_1, u_2 \in \Sigma_c^{lu}$  when the property described above holds.

The overall complexity of the control strategy synthesis by Horn is  $\mathcal{O}(k!n^k)$ , where  $n$  is the size of the product automaton and  $k$  is the number of pairs in the Rabin condition of the product automaton. The modifications in the computation of the direct attractor we made in order to adapt the algorithm to deal with stuttering behavior do not change the overall complexity. Note that, in general, Rabin games are NP-complete, so the exponential complexity with respect to  $k$  is not surprising. However, LTL formulas are usually translated into Rabin automata with very few tuples in their acceptance condition.

## 6.8 Implementation and Case Study

The method described in this paper was implemented in MATLAB as the software package `conPAS2`, where all polyhedral operations were performed using the MPT toolbox [Kvasnica et al., 2004]. The tool takes as input a PWA system (as defined in

Equation (6.2)) and an LTL formula and produces a set of satisfying initial regions and a feedback control strategy for the system (see Section 6.5). The tool is freely downloadable from our web site at <http://hyness.bu.edu/software>.

For the sake of presentation, we analyzed a planar PWA system ( $N = M = 2$  in Equation (6.2)) with 36 polytopes (Figure 6.3), where only the labels of the polytopes are shown). The system is similar to the one used as a case study in Chapter 4 and its trajectories go towards one of two possible stable equilibria located in regions  $\mathcal{X}_{10}$  and  $\mathcal{X}_{27}$  (see Figure 6.3(a)).

We are interested in finding initial states and a control strategy satisfying the following specification:

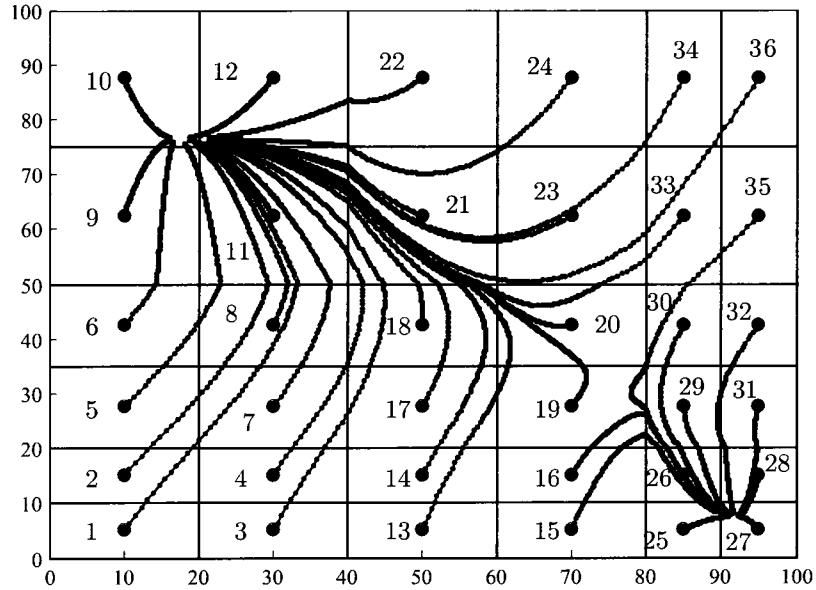
$$\phi = \diamond \square 10 \wedge \square \neg (17 \vee 18 \vee 19 \vee 20) \quad (6.17)$$

or ”eventually visit region  $\mathcal{X}_{10}$  and remain there forever and always avoid regions  $\mathcal{X}_{17}$ ,  $\mathcal{X}_{18}$ ,  $\mathcal{X}_{19}$ , and  $\mathcal{X}_{20}$ ”. Note that this specification cannot be translated into a deterministic Büchi automaton, and therefore the method from [Kloetzer and Belta, 2008a] cannot be used to generate a control strategy.

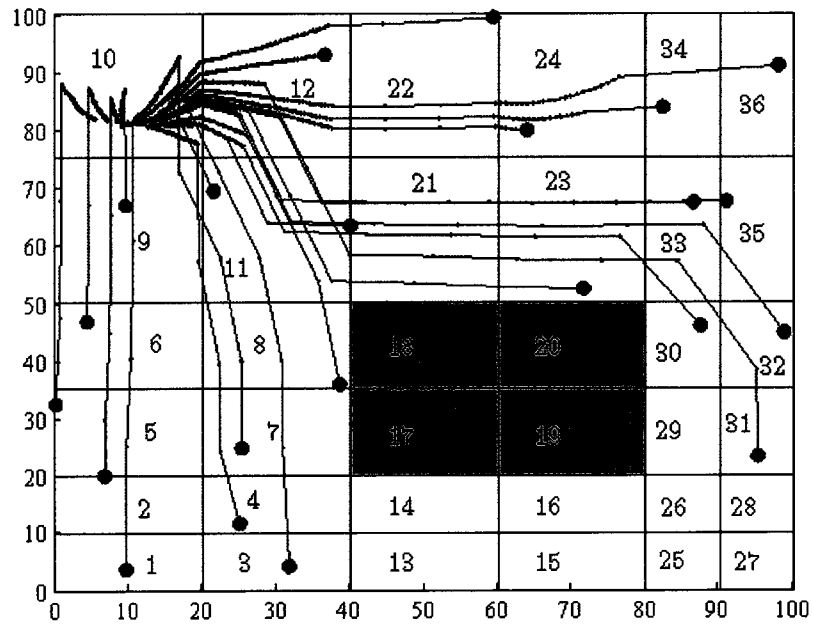
A control transition system  $\mathcal{T}_c$  with 36 states was constructed. Out of the total 396 nonempty input regions found (denoted by  $U_i^U$  in Section 6.3), 274 were ”large enough” (the radii of their inscribed spheres were larger than  $\epsilon = 0.05$ ) to be considered for a robust control strategy and included in  $\mathcal{T}_c$ . After reducing the size of  $\mathcal{T}_c$  (*i.e.*, removing unnecessary nondeterministic transitions as explained in Section 6.7) only 74 input regions were included out of which 25 were deterministic and 31 were identified as stuttering. The specification  $\phi$  (Equation (6.17)) translates into a deterministic Rabin automaton with 4 states and 1 pair in its acceptance condition. The computation of  $\mathcal{T}_c$  required under a minute and the construction of the control strategy required an additional 3 minutes on a 3.4 GHz, Intel Pentium 4 machine

with 1GB of memory. The satisfying initial region identified by `conPAS2` is shown in light gray in Figure 6.3(b)). Starting from random initial conditions, trajectories of the closed loop system were simulated (Figure 6.3(b)), where at each step applied inputs were corrupted by noise bounded by  $\epsilon$ . All simulated trajectories avoid the unsafe regions (shown in dark gray in Figure 6.3(b)) and satisfy the specification, thereby demonstrating the correctness and robustness of the control strategy.

For this particular case study, the target region  $\mathcal{X}_{10}$  of specification  $\phi$  is reachable only through transitions under stuttering inputs in  $\mathcal{T}_c$ . Therefore, a satisfying control strategy can be identified only from region  $\mathcal{X}_{10}$ , unless stuttering behavior is considered. The additional computation described in Section 6.6 allowed us to expand the satisfying initial set from  $\mathcal{X}_{10}$  only to the entire region shown in light gray in Figure 6.3(b).



(a) Trajectories of the uncontrolled PWA system go towards one of two possible stable equilibria located in regions  $\mathcal{X}_{10}$  and  $\mathcal{X}_{27}$  (initial states are shown as small circles).



(b) Trajectories of the closed loop PWA system originating anywhere in the satisfying region (light gray) satisfy the specification and eventually reach and remain in region  $\mathcal{X}_{10}$ , while always avoiding regions  $\mathcal{X}_{17}$ ,  $\mathcal{X}_{18}$ ,  $\mathcal{X}_{19}$ , and  $\mathcal{X}_{20}$  (dark gray).

Figure 6-3: Results from the formal synthesis of a control strategy for a PWA system

## Chapter 7

# Applications to Synthetic Biology

In previous chapters, we developed methods for analysis, parameter synthesis, and control of piecewise affine (PWA) systems from rich temporal logic specifications. Since PWA systems can approximate nonlinear dynamics with arbitrary accuracy, they provide a good formalism for modeling genetic regulatory networks (GRNs), which can be studied using the formal methods developed in this work.

As discussed in Section 1.2, piecewise affine (PWA) models of GRNs can be obtained by an identification procedure from input output data, collected experimentally from the system. Such an approach is suitable in cases, where a system that has been previously designed and constructed must be further validated or tuned. An analysis procedure can also be incorporated at an earlier phase of the design process, before the actual system has been built, by constructing a model from data, characterizing separate system components. This is perhaps more practical, since it allows potential designs to be filtered *a priori* and can prevent the costly experimental construction of devices that don't work as required.

In this chapter, we use an approach that combines the two strategies. We use experimental data available from [Hooshangi et al., 2005] to obtain a system model and then deduce properties of the different components of the system based on this model. We treat those components as parts that can be recombined into novel devices, and explore a subset of the space of such potential devices designs. We apply the methods developed in this dissertation to construct finite abstractions of

the different PWA device models and analyze them formally, using specifications expressed as linear temporal logic (LTL) formulas.

Our results indicate that requiring a specification to be satisfied by all trajectories of the model is, in general, too restrictive, but analysis can be used for model invalidation. The procedure we developed in Chapter 4 relies on the computation of both a satisfying region (*i.e.* a region of initial conditions, from which all trajectories of the model are guaranteed to satisfy the specification) and a violating region (*i.e.* a region, from which all trajectories are guaranteed to satisfy the negation of specification and, therefore, are violating). To guarantee that a model satisfies a specification, the satisfying region must cover the entire state space of the system. As it will become clear in this chapter, such a guarantee is too strong and cannot be obtained even for simple systems. However, the existence of violating trajectories can be guaranteed, whenever a nonempty violating region is identified. This can be very useful when evaluating a (potentially large) set of possible device designs. A design can be considered “good” if our analysis procedure reveals a (large) satisfying region and an empty or small violating region. A design is “bad”, whenever a substantial violating region is found. Such an approach can be used to automatically explore the large space of potential device designs that can be constructed from characterized parts, available in online databases and repositories.

The methods we developed in previous chapters consider separately the satisfaction of specifications by trajectories of the system originating in different regions. Therefore, additional consideration must be taken when formulating certain specifications, such as the existence of multiple stable equilibria, and interpreting the results. Intuitively, bistability can be expressed as the existence of two invariant regions  $A$  and  $B$  (*i.e.*  $\phi = A \rightarrow \Box A \wedge B \rightarrow \Box B$ ). However, such a specification will be satisfied from some initial states even when only one of the regions is an invariant.

For example, consider a case where region  $A$  is an invariant of system trajectories but region  $B$  is not. Our method will report that there exist a region satisfying specification  $\phi$  (namely, region  $A$ ), which might lead to the erroneous conclusion that the system is bistable. Even when both  $A$  and  $B$  are invariants, the relative volume of the satisfying region obtained using our approach does not represent a region satisfying a bistability specification but rather the sum of the regions satisfying two separate monostability specifications. To avoid such confusion, it is better to consider specifications such as bistability by treating them as separate monostability specifications. This leads to more informative results, since it allows the sizes of the attractor regions for the separate equilibria to be compared. In addition, such an approach also leads to efficiency improvements due to the smaller formulas used. We follow this approach for the case studies considered in this chapter.

We implement our analysis procedure described in Chapter 4 as a tool for studying synthetic gene networks. To make its use more intuitive, this implementation takes as input a specification expressed as an LTL formula and a continuous time description of the system, which is discretized internally. The formula guided refinement strategies described in Chapter 4 are not currently implemented. However, the characterization of stuttering behavior described in Chapter 6 is provided as an option to decrease the conservatism of the method. If this additional computation is performed, the specification language is restricted to  $LTL-X$  (Linear Temporal Logic without the “Next” operator) in order to guarantee the correctness of the results.

The rest of this chapter is organized as follows. In Section 7.1, we discuss the details of the PWA models we consider, which are used as approximations of continuous time ODE models. In Section 7.2, we present the results of the analysis of two simple PWA models, inspired from the genetic toggle switch [Gardner et al., 2000] and the

repressilator [Elowitz and Leibler, 2000]. In Section 7.3, we construct a PWA model of a synthetic transcriptional cascade [Hooshangi et al., 2005] based on a previously identified model [Braun et al., 2005] and use this information to construct a set of device designs based on the use of parts. In Section 7.4, we explore this space of combinatorial device designs using the analysis procedures we developed as part of this work.

Throughout this chapter, we use the terms “toggle switch” and “repressilator” to describe circuit topologies similar to the ones from [Gardner et al., 2000] and [Elowitz and Leibler, 2000], respectively, rather than to refer to the actual systems from those works. For clarity of presentation, we use simpler models (*i.e.* more drastic approximations) than what would be required for real applications, but the goal of the case studies presented in this chapter is simply to demonstrate how our methods can be applied to the field of synthetic biology.

## 7.1 PWA Models of GRNs

To illustrate the construction of PWA models used in this work, we consider a cross inhibition, toggle switch topology, consisting of two genes, expressing repressor proteins  $R_1$  and  $R_2$  (Figure 7.2(a)). Repressor  $R_1$  binds to promoter  $P_1$  to down-regulate the expression of  $R_2$  and vice versa. We assume that a large number of well mixed molecules are present in the system and transcription factor binding and unbinding to promoters proceeds much faster than transcription or translation of genes [McAdams and Shapiro, 1995]. Then, the gene regulation function (GRF), relating repressor concentration to transcription rate from the regulated promoter, can be described by a Hill type function [Hill, 1913] (see Figure 7.1(a)). The nonlinearity of GRFs leads to systems of nonlinear differential equation as models of gene network, which cannot be easily analyzed. Instead, we consider piecewise affine approximations of



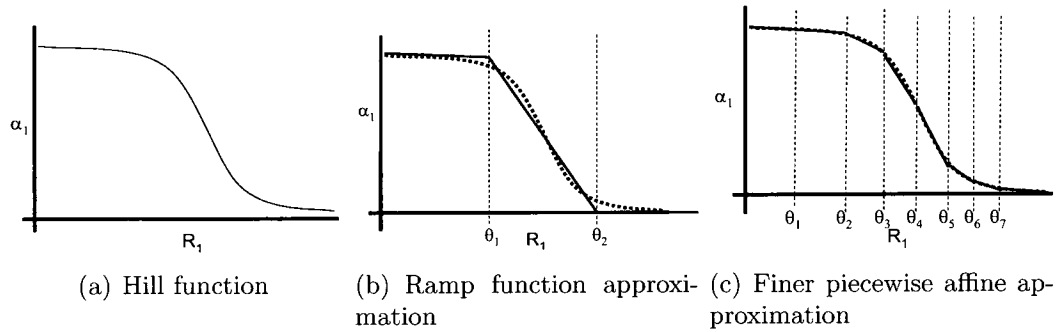


Figure 7.1: Gene regulatory function and approximations. In Figures 7.1(b) and 7.1(c) the dashed lines represent the Hill function and the solid lines its piecewise affine approximations.

GRFs.

The simplest such approximation (not considering the more restrictive, piecewise constant step function) is a ramp function, which is defined by two threshold values, inducing three regions of different dynamics (see Figure 7.1(b)). At low repressor concentrations (*i.e.*  $R_1 \leq \theta^1$ ) the regulated gene is fully expressed (*i.e.*  $\alpha_1$ , the rate of transcription from promoter  $P_1$  is maximal), at high repressor concentrations (*i.e.*  $R_1 > \theta^1$ ) expression is only basal and the response between the two thresholds is graded. In order to obtain more accurate approximations of a GRF, more break-points can be introduced resulting in general piecewise affine regulation functions and a larger number of regulation regions (see Figure 7.1(c)). The problem of approximating a function using linear segments is related to methods for piecewise linear interpolation, which date back to ancient times [Meijering, 2002] but may result in an unnecessary large number of regulation regions. More recent results allow accurate piecewise linear approximations with less break points to be obtained automatically using available algorithms [Ferrari-trecate et al., 2001].

Due to the simplicity of the resulting models, we use ramp approximations of GRFs for the case studies discussed in this chapter. In addition, we only consider

promoters regulated from a single repressor but the effects from multiple repressor/activator proteins or external inducers can be modeled through PWA functions of higher dimension. By modeling transcription and translation as a single step, a continuous time PWA model of the toggle switch network (Figure 7.2(a)) can be written as

$$\begin{aligned}\dot{R}_1 &= \alpha_2^0 + \alpha_2^m r^-(R_2, \theta_2^1, \theta_2^2) - \gamma_1 R_1 \\ \dot{R}_2 &= \alpha_1^0 + \alpha_1^m r^-(R_1, \theta_1^1, \theta_1^2) - \gamma_2 R_2,\end{aligned}\tag{7.1}$$

where  $R_1$  and  $R_2$  denote the concentrations of the two repressor proteins<sup>1</sup> and  $\gamma_1$  and  $\gamma_2$  are their respective degradation rates.  $\alpha_i^0$  is the basal and  $\alpha_i^m$  is the maximal rate of expression from promoter  $P_i$  and  $r^-$  is a ramp function of the concentrations of repressors  $R_i$  and threshold values  $\theta_i^1, \theta_i^2$ , and ranges between 0 and 1. The partitioning of the state space of the system is given by the thresholds of the regulation functions. In order to obtain discrete time piecewise affine models, which can be analyzed using the methods developed in this dissertation, the continuous dynamics in each region are discretized using a time step that is small compared to the dynamics of the system. Such models can be easily extended into higher dimensions to capture larger networks (*e.g.* the model for a repressilator system (Figure 7.2(b)) includes three state variables  $R_1, R_2$ , and  $R_3$ ) or to capture explicitly dynamics of other species, such as mRNA.

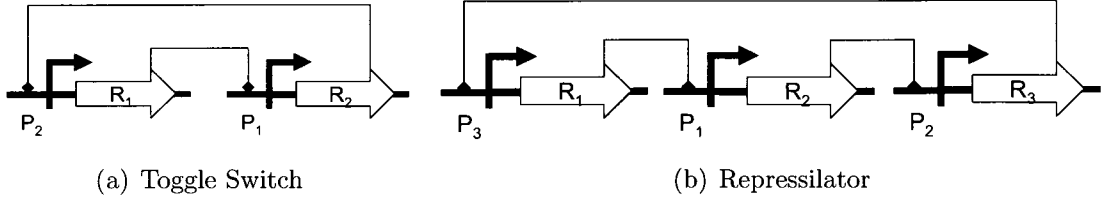


Figure 7.2: Gene regulatory network topologies used in our case studies (square arrowheads indicate repression).

## 7.2 Analysis of a toggle switch and a repressilator

We construct a simple toggle switch system (Figure 7.2(a)) with the following parameters:

$$\begin{aligned} \alpha_1^0 &= 0.15, \alpha_1^m = 1.55, \theta_1^1 = 40, \theta_1^2 = 80, \gamma_1 = 0.01 \\ \alpha_2^0 &= 0.17, \alpha_2^m = 0.92, \theta_2^1 = 20, \theta_2^2 = 50, \gamma_1 = 0.02 \end{aligned} \quad (7.2)$$

The maximum allowed concentration for both  $R_1$  and  $R_2$  is 100 (*i.e.* the system is defined on the rectangle  $\mathcal{X} = \text{hull}(\{[0, 0], [100, 0], [100, 100], [0, 100]\})$ ), which is partitioned into 9 regions by the thresholds. A discrete time PWA model is obtained from (7.1) with parameters (7.2) through discretization with time step  $T = 20$ , as described in Section 7.1. To introduce uncertainty in our model, we allow each parameters to vary in a range, specified as a percentage of the values from (7.2).

Depending on their initial conditions, trajectories of the system go towards one of two possible stable equilibria (Figure 7.3). When parameter uncertainty is introduced in the model, trajectories from the same initial condition can go to both equilibria (Figure 7.3(b)), *i.e.* the system is nondeterministic. As described earlier, we study the regions of attractions for the two stable equilibria to asses the bistability of the system. Using the system's thresholds, the following linear predicates are defined:

<sup>1</sup>here, we use  $R_1$  and  $R_2$  both as names of proteins and to denote their respective concentrations

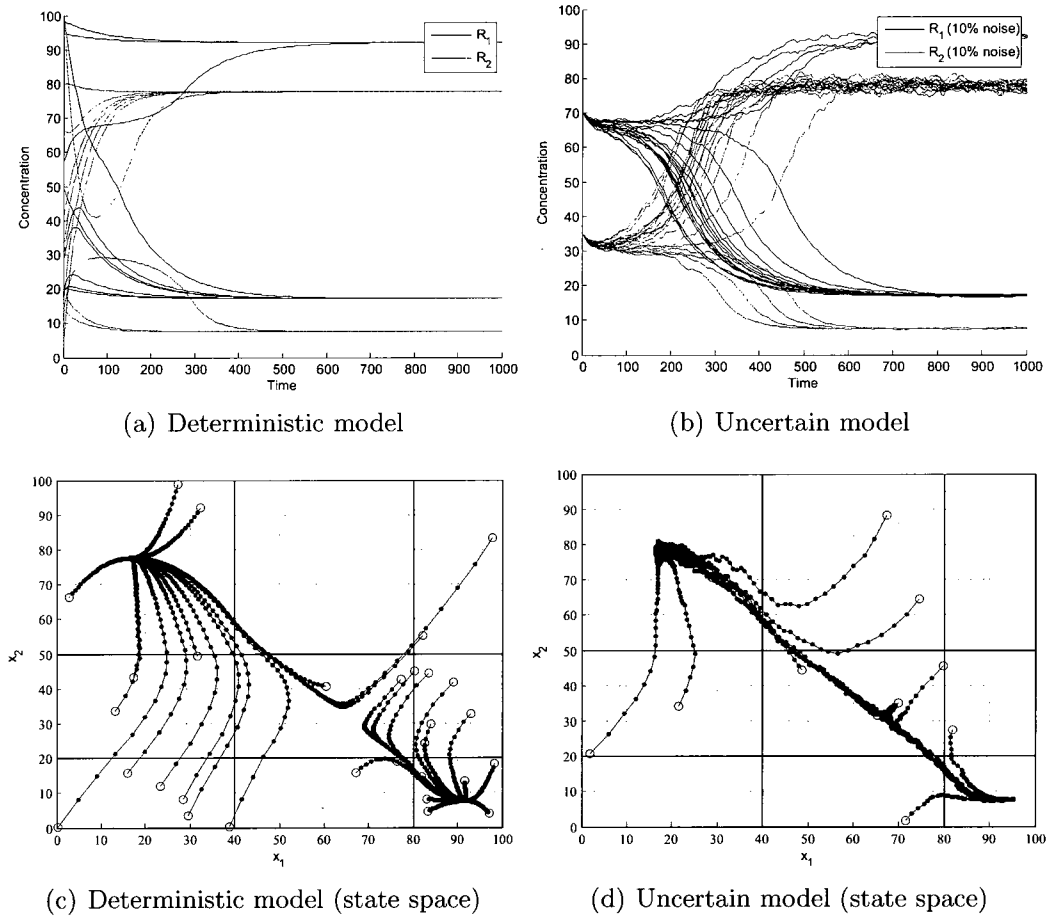


Figure 7-3: Numerical simulations of the deterministic and 10% parameter uncertainty toggle switch models. Trajectories in both state space and time are shown.

Model	$\phi_1$	$\neg\phi_1$	time (figure)	$\phi_2$	$\neg\phi_2$	time (figure)
deterministic	19.5%	78.4%	15 sec (7.4(a))	78.4%	19.5%	14 sec (7.4(b))
1% noise	6.8%	65.4%	2 min (7.4(c))	55.4%	13.4%	3 min (7.4(d))
10% noise	0%	49.5%	11 min (7.4(e))	43.9%	6.6%	18 min (7.4(f))

Table 7.1: Relative volumes and computation times of satisfying and violating regions for the toggle switch model with different amounts of parameter uncertainty.

$\pi_1 := R_1 > 80$ ,  $\pi_2 := R_1 < 40$ ,  $\pi_3 := R_2 > 50$ , and  $\pi_4 := R_2 < 20$ . We consider specifications  $\phi_1 = \diamond\Box(\pi_1 \wedge \pi_4)$  and  $\phi_2 = \diamond\Box(\pi_2 \wedge \pi_3)$ . In other words,  $\phi_1$  is satisfied if all trajectories of the system eventually reach and stabilize in the region, where the concentrations of  $R_1$  are high (above 80) and the concentrations of  $R_2$  are low (below 20). Similarly,  $\phi_2$  is satisfied if all trajectories of the system eventually reach and stabilize in the region, where  $R_1$  is below 40 and  $R_2$  is above 50. Exploring both specifications separately allows us to identify the attractor regions of the two stable equilibria of the system and compare their sizes.

The models were analyzed using the method described in Chapter 4, with a refinement limit  $\epsilon = 1$ . The results from the analysis of the three models (deterministic, 1% noise, and 10% noise) with specifications  $\phi_1$  and  $\phi_2$  are summarized in Table 7.1 and the satisfying and violating regions are presented graphically in Figure 7.2. For the deterministic model, satisfying states were identified for both specifications (Figures 7.4(a) and 7.4(b)), which suggest that the two equilibria are stable. The attractor region for the equilibrium where  $R_2$  concentrations are high and  $R_1$  concentrations are low is larger. As expected, when parameter uncertainty is introduced in the model, smaller satisfying and violating regions can be identified (Figures 7.4(c) and 7.4(d)). When 10% uncertainty in the parameters is introduced, a satisfying region cannot be identified for specification  $\phi_2$  (Figures 7.4(e) and 7.4(f)) and, therefore, bistability cannot be guaranteed.

We construct a symmetric repressilator model (Figure 7.2(b)) with the following

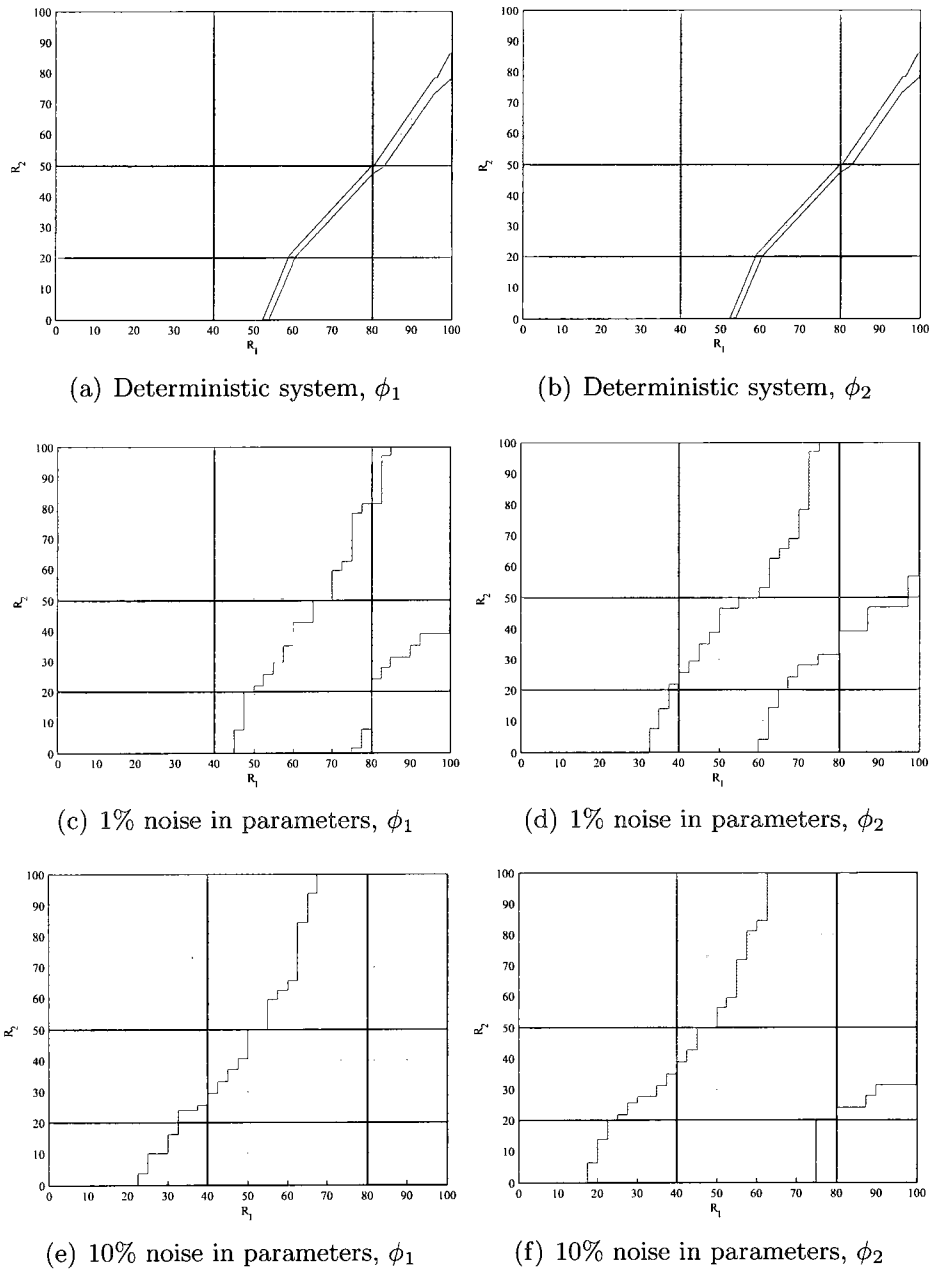


Figure 7-4: Results from the analysis of fixed parameter (deterministic), and uncertain parameter toggle switch model for specifications  $\phi_1$  (eventually, the concentrations of  $R_1$  and  $R_2$  stabilize in a high and a low state, respectively) and  $\phi_2$  (eventually the concentrations of  $R_1$  and  $R_2$  stabilize in a low and a high state, respectively). Satisfying regions are shown in green (lighter gray) and violating regions are shown in red (darker gray).

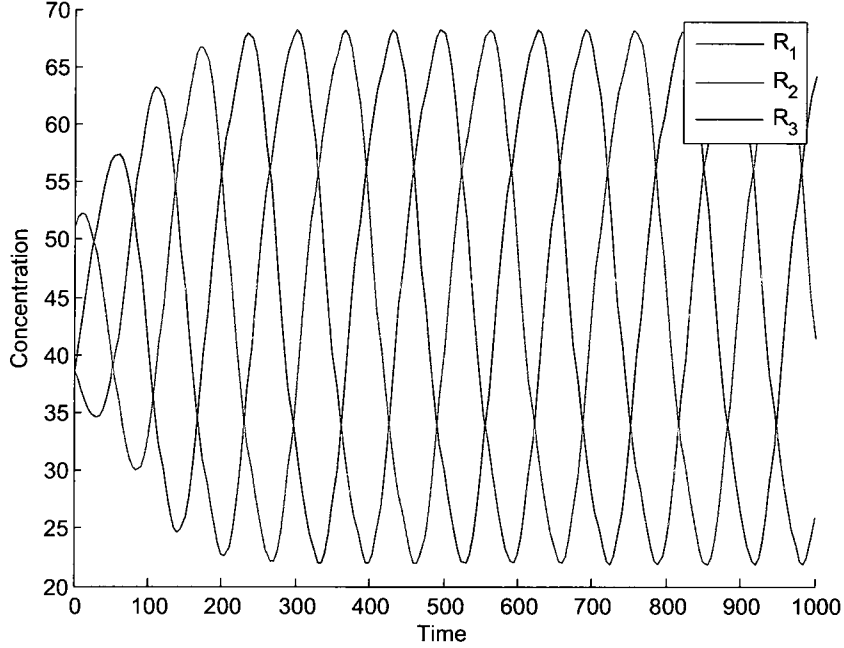


Figure 7.5: Numerical simulation of the deterministic repressilator model.

parameters, identical for all  $i = 1, \dots, 3$ :

$$\alpha_i^0 = 0.02, \alpha_i^m = 1.6, \theta_i^1 = 30, \theta_i^2 = 60, \gamma_i = 0.02 \quad (7.3)$$

The maximum allowed concentration for  $R_1$ ,  $R_2$ , and  $R_3$  is 100 and the system is a cube partitioned by the thresholds into 27 regions. As before the discrete time PWA model is obtained by discretizing the continuous model as described in Section 7.1. To introduce uncertainty in the model, each parameter is allowed to vary in a 1% range around the value from (7.3). Trajectories of the repressilator system oscillate (Figure 7.2 and we focus specifically on characterizing the existence of oscillatory behavior in the concentrations of  $R_3$ . We define linear predicates  $\pi_1 := R_3 > 60$  and  $\pi_2 := R_3 < 30$ . We define LTL formula  $\phi = \Box(\Diamond\pi_1 \wedge \Diamond\pi_2)$ , which specifies that concentrations of  $R_3$  always oscillate, reaching values over 60 and below 30 (note that  $\pi_1$  and  $\pi_2$  cannot be satisfied at the same time).

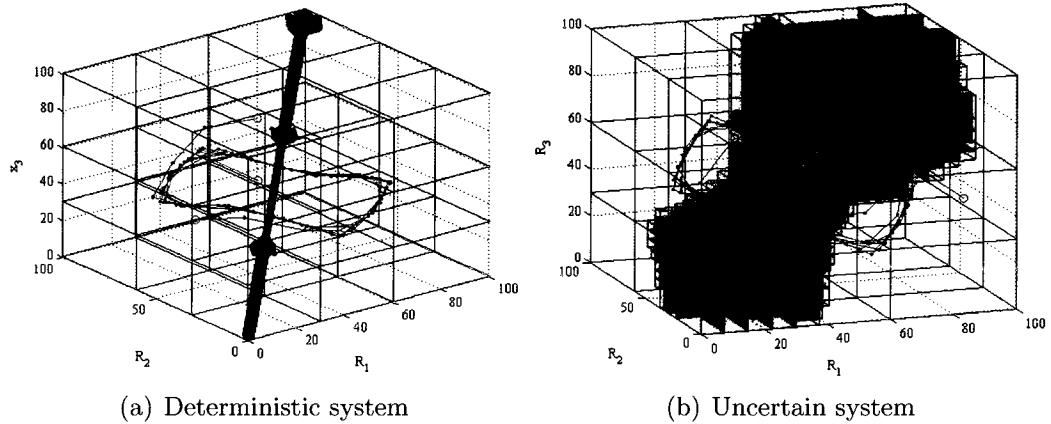


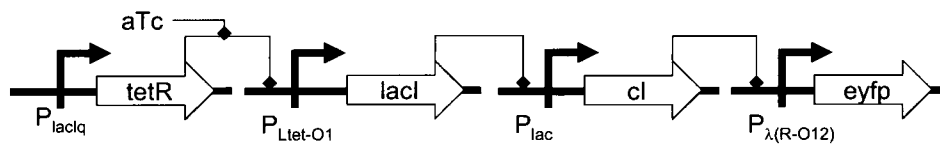
Figure 7.6: Results from the analysis of fixed parameter (deterministic), and uncertain parameter (1% noise) repressilator model for specification  $\phi = \square(\diamond\pi_1 \wedge \diamond\pi_2)$ . The satisfaction of the specification can be guaranteed for trajectories of the system originating anywhere but the region shown in black.

The models were analyzed using the method described in Chapter 4, with a refinement limit  $\epsilon = 1$ . The results of the analysis of the repressilator system are presented graphically in Figure 7.6. The relative volumes of the satisfying regions for the deterministic and uncertain systems were 99.8% and 69%, and the computation times were 11 min and 3h, respectively. No violating regions were identified for either model. As expected, the identified satisfying region decreases when uncertainty is introduced in the model.

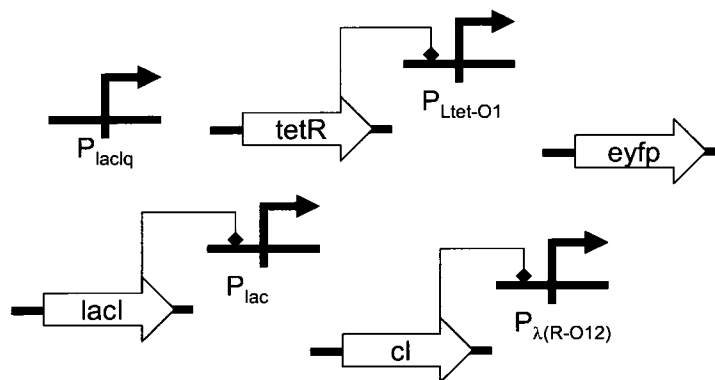
### 7.3 Constructing Devices from Parts

To further demonstrate the use of the methods developed in this dissertation, we study a PWA model of a transcriptional cascade (Figure 7.7(a)) that has been previously constructed [Hooshangi et al., 2005]. Parameters of a nonlinear ODE model have been previously identified from experimental data [Braun et al., 2005] and a piecewise multiaffine approximation has been constructed and studied in [Batt et al., 2008], while a more accurate piecewise multiaffine model approximation was consid-





(a) Transcriptional Cascade



(b) A set of genetic parts

Figure 7-7: The original transcriptional cascade used in this case study and the set of parts it is composed of.

ered in [Batt et al., 2007b]. The previously developed models are specified in continuous time but the dynamics in each region of the state space can be discretized. The framework developed in this dissertation cannot handle multiaffine dynamics directly but the previous model is multiaffine only in the concentrations of the external inducer  $aTc$ . We consider separately the cases when  $aTc$  is absent from the system and present in saturating concentrations (*i.e.* the repression of promoter  $P_{LtetO1}$  by  $TetR$  is completely lifted). This results in two separate piecewise affine models that describe the dynamics of the cascade with and without  $aTc$ . Similarly, additional models can be constructed to describe the system for intermediate concentrations of  $aTc$ , but such conditions are not considered for this case study. To construct the cascade model, we use the following parameters:

$$\begin{aligned}
\alpha_{P_{lacIq}}^0 &= 0, \alpha_{P_{lacIq}}^m = 221.78, \\
\alpha_{P_{Ltet-O1}}^0 &= 3.28605, \alpha_{P_{Ltet-O1}}^m = 903.86, \\
\alpha_{P_{lac}}^0 &= 3.39794, \alpha_{P_{lac}}^m = 389.961, \\
\alpha_{P_{\lambda(R-O12)}}^0 &= 4.3154, \alpha_{P_{\lambda(R-O12)}}^m = 4000, \\
\theta_{TetR}^1 &= 25, \theta_{TetR}^2 = 100, \gamma_{TetR} = 0.013045 \\
\theta_{LacI}^1 &= 100, \theta_{LacI}^2 = 2000, \gamma_{LacI} = 0.014094 \\
\theta_{CI}^1 &= 1500, \theta_{CI}^2 = 4000, \gamma_{CI} = 0.015013 \\
\gamma_{eyfp} &= 0.0122
\end{aligned} \tag{7.4}$$

Note that no thresholds are provided for  $eyfp$ , since it is not a repressor. Using ramp functions, we model the transcriptional cascade in Equation (7.5), as described in Section 7.1. Note that when  $aTc$  is present in the system, we consider a model where the expression from promoter  $P_{Ltet-O1}$  is  $\alpha_{P_{Ltet-O1}}^0 + \alpha_{P_{Ltet-O1}}^m$  (*i.e.* regulation from  $tetR$  is lifted). In Section 7.4.1, we present results from the analysis of this model.

$$\begin{aligned}
\dot{TetR} &= \alpha_{P_{lacIq}}^m - \gamma_{TetR} TetR \\
\dot{LacI} &= \alpha_{P_{Ltet-O1}}^0 + \alpha_{P_{Ltet-O1}}^m r^- (TetR, \theta_{TetR}^1, \theta_{TetR}^2) - \gamma_{LacI} LacI \\
\dot{CI} &= \alpha_{P_{lac}}^0 + \alpha_{P_{lac}}^m r^- (LacI, \theta_{LacI}^1, \theta_{LacI}^2) - \gamma_{CI} CI \\
\dot{eyfp} &= \alpha_{P_{\lambda(R-O12)}}^0 + \alpha_{P_{\lambda(R-O12)}}^m r^- (CI, \theta_{CI}^1, \theta_{CI}^2) - \gamma_{eyfp} eyfp
\end{aligned} \tag{7.5}$$

The transcriptional cascade (Figure 7.7(a)) is composed of a number of separate parts: promoter and repressor pairs that can be used to construct different devices (Figure 7.7(b)). We ignore the constitutive promoter  $P_{lacIq}$  and, assuming that the concentrations of repressors  $tetR$ ,  $lacI$ , and  $cI$  can be measured directly, ignore the reporter  $eyfp$ . This leaves a set of genetic parts (3 repressor/promoter pairs) that we use to construct and model new devices and analyze the results.

We focus on the toggle switch and repressilator topologies studied in Section 7.2 and construct all devices with those topologies, possible from the available parts (note that due to the cyclic nature of the networks, only the relative order of the repressors matters). The set of device designs is shown in Figure 7-8. To construct device models, we follow the approach described in Section 7.1. As an example, we show the model for the  $tetR/cI$  toggle switch device (Figure 7.8(e)) in Equation (7.6). Note that for all devices except  $tetR/cI$ , we consider separate models for the case when  $aTc$  is present and absent. We analyze the models of all devices in Section 7.4.

$$\begin{aligned}
\dot{LacI} &= \alpha_{P_{\lambda(R-O12)}}^0 + \alpha_{P_{\lambda(R-O12)}}^m r^- (CI, \theta_{CI}^1, \theta_{CI}^2) - \gamma_{LacI} LacI \\
\dot{CI} &= \alpha_{P_{lac}}^0 + \alpha_{P_{lac}}^m r^- (LacI, \theta_{LacI}^1, \theta_{LacI}^2) - \gamma_{CI} CI
\end{aligned} \tag{7.6}$$

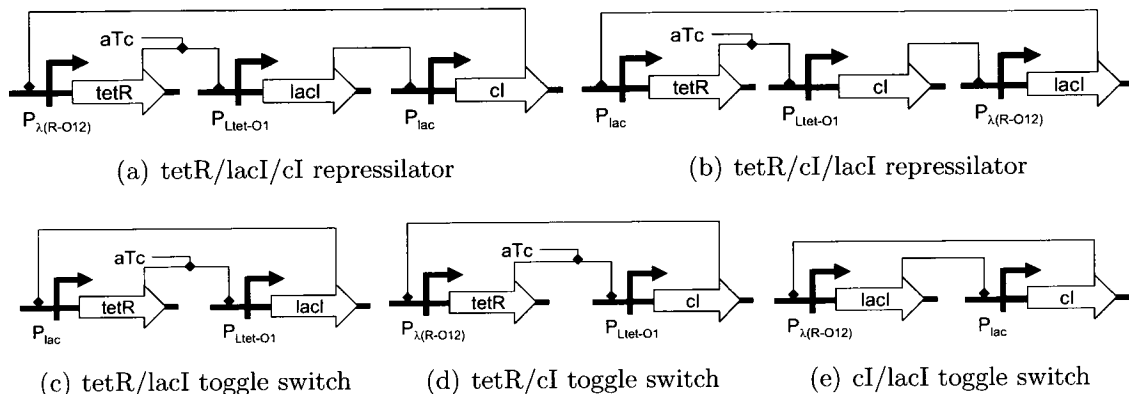


Figure 7.8: A set of devices with a repressilator or a toggle switch topology, constructed from the parts from 7.7(b).

## 7.4 Formal Analysis of Device Models

In the previous section, we described the construction of a set of models from parts. Starting from the original transcriptional cascade model, we constructed two separate implementations of a repressilator topology, and three separate toggle switch circuits. In this section, we apply our analysis procedure (Chapter 4) to study those system models. For all results discussed in this section we do not impose a refinement limit for our algorithm as before but, instead, limit the number of iterations to 50.

### 7.4.1 Cascade results

Trajectories of the transcriptional cascade model (Figure 7.7(a)) reach an equilibrium, where the concentration of *eyfp* is low when no *aTc* is present in the system (Figure 7.9(a)), and high if *aTc* is available (Figure 7.9(b)). To study the two equilibria, we define the following linear predicates  $\pi_1 := \text{eyfp} < 1000$  and  $\pi_2 := \text{eyfp} > 30000$ . We define LTL formulas  $\phi_1 = \diamond\Box\pi_1$  and  $\phi_2 = \diamond\Box\pi_2$ . The results of the analysis are summarized in Table 7.2.

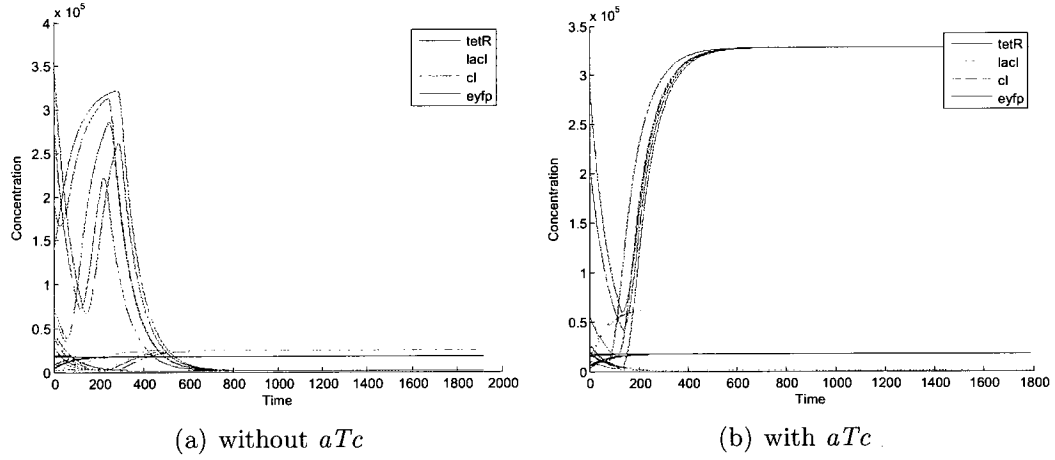


Figure 7-9: Numerical simulations of the transcriptional cascade model.

$aTc$	$\phi_1$	$\neg\phi_1$	time	$\phi_2$	$\neg\phi_2$	time
absent	0.007%	0%	2h 40min	0%	0.007%	3h
present	0%	99.8%	1.5 sec	99.7%	0%	24 min

Table 7.2: Relative volumes and computation times of satisfying and violating regions for the cascade model.

#### 7.4.2 Repressilator results

For the two repressilator topologies, which we denote as  $tetR/lacI/cI$  (Figure 7.8(a)) and  $tetR/cI/lacI$  (Figure 7.8(b)), we want to study the existence of oscillatory behavior. Specifically, we only consider oscillations in the concentration of one of the proteins. From numerical simulations, it appears that trajectories of the  $tetR/lacI/cI$  system do not oscillate but instead settle in an equilibrium where the concentration of  $cI$  is high when no  $aTc$  is present in the system (Figure 7.10(a)), and low if  $aTc$  is available (Figure 7.10(b)). Similarly, for the  $tetR/cI/lacI$  system it appears that the concentration of  $lacI$  is high when no  $aTc$  is present (Figure 7.10(c)), and low if  $aTc$  is available (Figure 7.10(d)). For systems  $tetR/lacI/cI$  and  $tetR/cI/lacI$ , we explore the possibilities that the concentrations of  $cI$  and  $lacI$ , respectively, oscillate or settle in either of the two equilibria. It is important to note that in the presence of  $aTc$ ,

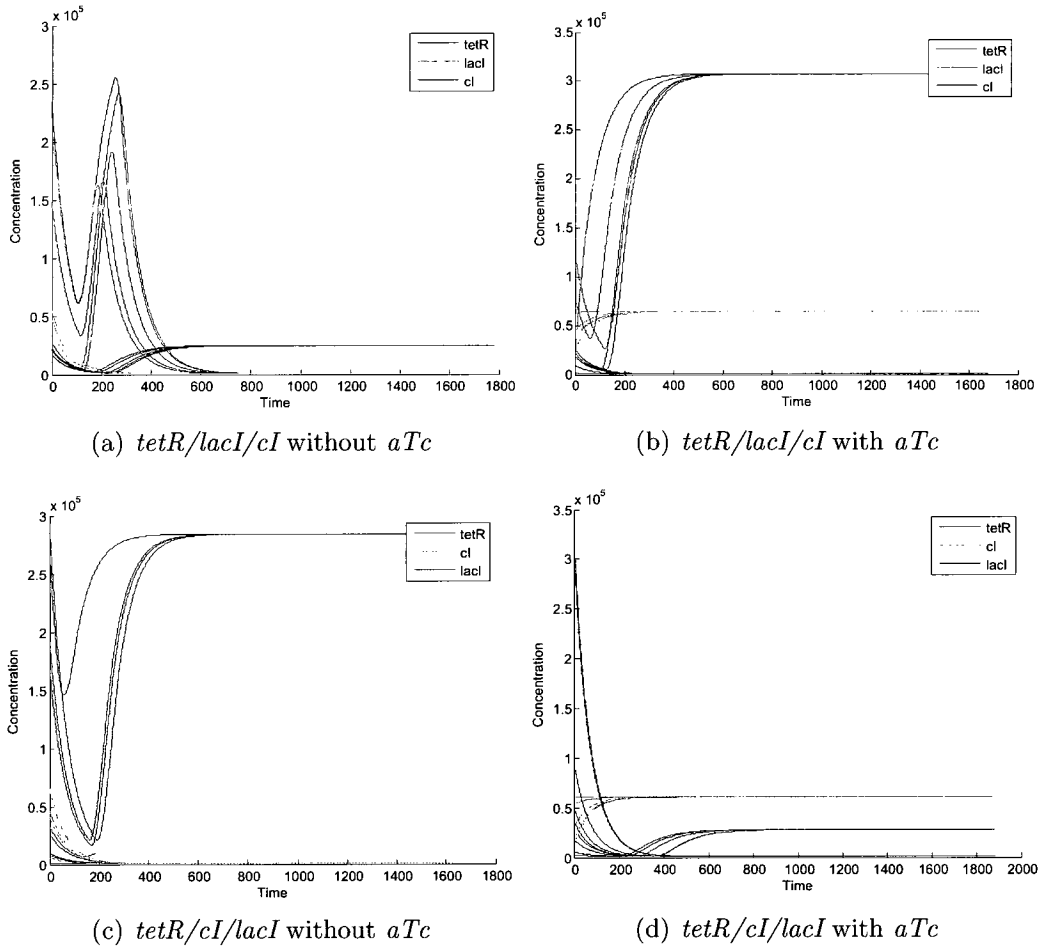


Figure 7.10: Numerical simulations of the two repressilator models.

$aTc$	$\phi_1$	$\neg\phi_1$	time	$\phi_2$	$\neg\phi_2$	time	$\phi_3$	$\neg\phi_3$	time
absent	0%	100%	1.5 sec	0%	100%	2.5sec	0%	100%	2.5sec
present	99.8%	0%	5 sec	0%	99.8%	1.5 sec	0%	99.8%	1.5 sec

Table 7.3: Relative volumes and computation times of satisfying and violating regions for the  $tetR/lacI/cI$  repressilator model.

$aTc$	$\phi_4$	$\neg\phi_4$	time	$\phi_5$	$\neg\phi_5$	time	$\phi_6$	$\neg\phi_6$	time
absent	0%	100%	2.5 sec	2.45%	0%	35 min	0%	99.96%	1.5 sec
present	99.96%	0%	1.5 sec	0%	99.7%	1.5 sec	0%	99.96%	1.5 sec

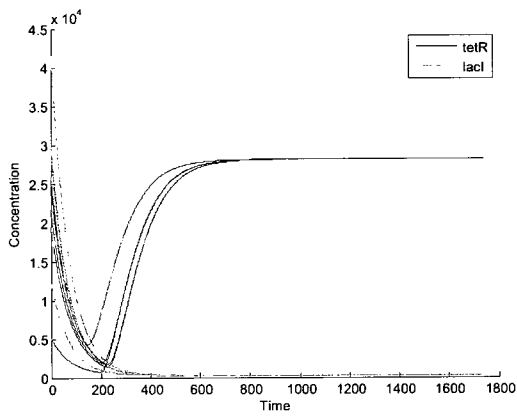
Table 7.4: Relative volumes and computation times of satisfying and violating regions for the  $tetR/cI/lacI$  repressilator model.

the feedback repressilator topology is broken and the systems reduce to transcriptional cascades. Therefore, for those systems we can only hope to achieve oscillatory behavior in the absence of  $aTc$ .

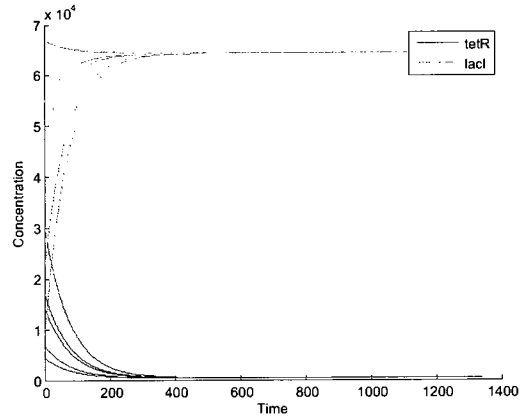
For  $tetR/lacI/cI$ , we define the following linear predicates  $\pi_1 := cI < 1000$  and  $\pi_2 := cI > 20000$ . We define LTL formulas  $\phi_1 = \diamond\Box\pi_1$ ,  $\phi_2 = \diamond\Box\pi_2$ , and  $\phi_3 = \Box(\diamond\pi_1 \wedge \diamond\pi_2)$ . Similarly, for  $tetR/cI/lacI$ , we define predicates  $\pi_3 := lacI < 1000$  and  $\pi_4 := lacI > 250000$  and formulas  $\phi_4 = \diamond\Box\pi_3$ ,  $\phi_5 = \diamond\Box\pi_4$ , and  $\phi_6 = \Box(\diamond\pi_3 \wedge \diamond\pi_4)$ . The analysis results for systems  $tetR/lacI/cI$  and  $tetR/cI/lacI$  are summarized in tables 7.3 and 7.4, respectively.

### 7.4.3 Toggle switch results

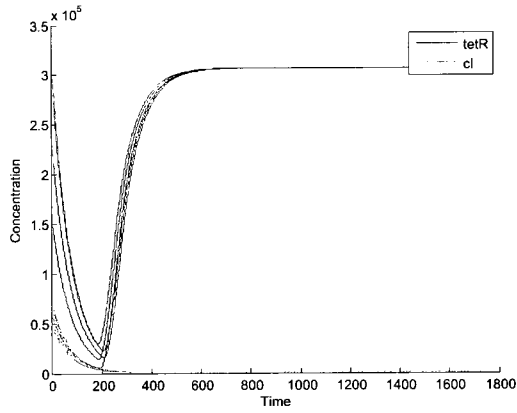
We denote the three possible devices with a toggle switch topology, constructed from the parts in Figure 7.7(b) as  $tetR/lacI$  (Figure 7.8(c)),  $tetR/cI$  (Figure 7.8(d)), and  $cI/lacI$  (Figure 7.8(e)). Using the analysis methods we have developed, we explore whether the systems have two stable equilibria, and compare their regions of attraction. As previously discussed, we express the existence of multiple equilibria using separate LTL formulas. As for the repressilator topologies discussed in the previous section, in the presence of  $aTc$ , the feedback toggle switch topology is broken



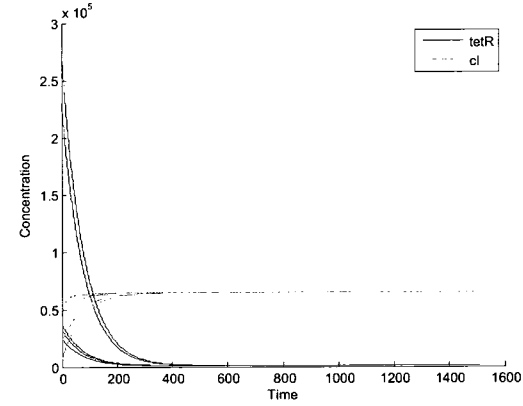
(a) *tetR/lacI* without *aTc*



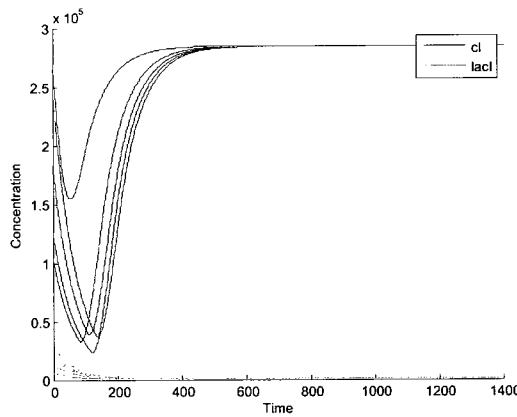
(b) *tetR/lacI* with *aTc*



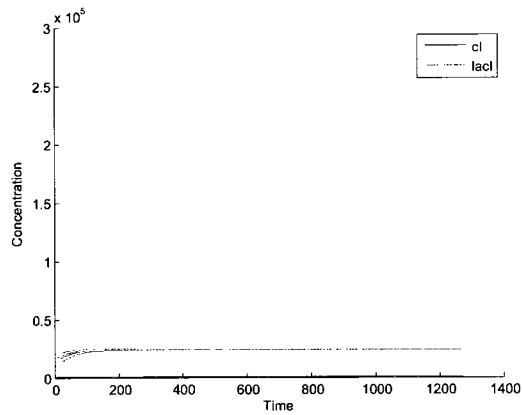
(c) *tetR/cI* without *aTc*



(d) *tetR/cI* with *aTc*



(e) *cI/lacI* from random initial conditions



(f) *cI/lacI* from specific initial conditions

Figure 7-11: Numerical simulations of the three toggle switch models.



for the first two systems and they reduce to transcriptional cascades. Therefore, for those devices we can only hope to achieve bistability in the absence of  $aTc$ . Only one model is constructed for the  $cI/lacI$  device as there is no input from  $aTc$ .

Simulated trajectories of the  $tetR/lacI$  system reaches an equilibrium where the concentration of  $lacI$  is low in the absence of  $aTc$  (Figure 7.11(a)) and high otherwise (Figure 7.11(b)). Similarly, trajectories of the  $tetR/cI$  system reaches an equilibrium where the concentration of  $cI$  is low in the absence of  $aTc$  (Figure 7.11(c)) and high otherwise (Figure 7.11(d)). For the  $cI/lacI$  there is not  $aTc$  input initial simulations from random initial conditions reveal an equilibrium, where the concentration of  $lacI$  is low. For all three systems, results from numerical simulations suggest that only a single stable equilibrium is present but for the  $tetR/lacI$  and  $tetR/cI$  systems, this equilibrium can be switched using the external input. We verify and explore further those results using the analysis procedures developed in this dissertation.

As for the toggle switch systems discussed in Section 7.2, we want to identify the regions of attraction for the stable equilibria separately and compare their sizes. For the  $tetR/lacI$  system, we define linear predicates  $\pi_1 := lacI > 60000$ ,  $\pi_2 := tetR > 25000$ ,  $\pi_3 := lacI < 250$ , and  $\pi_4 := tetR < 250$ . We are interested in analyzing the system for specifications  $\phi_1 = \diamond\Box(\pi_1 \wedge \pi_4)$  and  $\phi_2 = \diamond\Box(\pi_2 \wedge \pi_3)$ . For the  $tetR/cI$  system, we define linear predicates  $\pi_5 := cI > 60000$ ,  $\pi_6 := tetR > 300000$ ,  $\pi_7 := cI < 250$ , and  $\pi_8 := tetR < 500$  and formulas  $\phi_3 = \diamond\Box(\pi_5 \wedge \pi_8)$  and  $\phi_4 = \diamond\Box(\pi_6 \wedge \pi_7)$ . Finally, for the  $cI/lacI$  system we define  $\pi_9 := cI > 20000$ ,  $\pi_{10} := lacI > 250000$ ,  $\pi_{11} := cI < 250$ , and  $\pi_{12} := lacIR < 500$  and formulas  $\phi_5 = \diamond\Box(\pi_9 \wedge \pi_{12})$  and  $\phi_6 = \diamond\Box(\pi_{10} \wedge \pi_{11})$ . Because the formulas for the different systems are qualitatively similar, we simplify the notation and consider only formulas  $\phi_a$  and  $\phi_b$  (*i.e.*  $\phi_a$  is  $\phi_1$  for  $tetR/lacI$  but  $\phi_3$  for  $tetR/cI$ ). The results of the analysis are summarized in table 7.5.

model	$\phi_a$	$\neg\phi_a$	time	$\phi_b$	$\neg\phi_b$	time
<i>tetR/lacI</i> without <i>aTc</i>	0%	100%	1.5 sec	0.37%	0%	12 min
<i>tetR/lacI</i> with <i>aTc</i>	0%	100%	1 sec	0%	100%	1 sec
<i>tetR/cI</i> without <i>aTc</i>	0%	100%	1 sec	100%	0%	4 sec
<i>tetR/cI</i> with <i>aTc</i>	99.9%	0%	1 sec	0%	99.9%	1 sec
<i>cI/lacI</i>	0.4%	14.2%	2 min	12.65%	0.57%	2 min

Table 7.5: Relative volumes and computation times of satisfying and violating regions for the three toggle switch models.

The analysis results of the *cI/lacI* system reveal that a second stable equilibrium might exist. To confirm this, additional trajectories were simulated from initial conditions, where the concentrations of *lacI* were much higher than those of *cI*. The simulations confirm the existence of a second stable equilibrium state (Figure 7.11(f)).

## 7.5 Discussion

In Section 7.2, we studied a bistable toggle switch and an oscillating repressilator genetic network. The results demonstrated that the characteristic behavior of the two systems can be captured as LTL specifications and validated using the analysis procedures described in Chapter 4. In addition, we were able to study the sizes of the satisfying and violating regions for each specification in order to gain more information about the systems.

In Section 7.3, we described how the previously available model of a transcriptional cascade can be used to derive the models of two repressilator and three toggle switch circuits, constructed from the same set of parts. Based on the results from Section 7.2, we expected that a repressilator topology would produce systems that oscillate and a toggle switch topology would produce bistable systems. In Section 7.4, we used our analysis procedure to test *in silico*, whether such design requirements were indeed satisfied.

First, we analyzed the unmodified transcriptional cascade model. For the case when there is no  $aTc$ , our results identify only a very small region of initial conditions for the system, from which low  $eyfp$  concentration (below 1000) is eventually and stably reached. When  $aTc$  is present, we can guarantee that for a substantial region of initial conditions, the concentration of  $eyfp$  will eventually reach and stabilize at high values (over 30000). This is consistent with the expected behavior of the transcriptional cascade.

For the  $tetR/lacI/cI$  repressilator circuit, oscillations of the concentration of  $cI$  between values above 20000 and below 1000 are disproved from a large region of initial conditions. Similarly, oscillations of the concentration of  $lacI$  between values above 250000 and below 1000 are disproved for the  $tetR/cI/lacI$  circuit. Those results hold both in the presence and absence of  $aTc$ . For the first system, concentration of  $cI$  are likely to stabilize to low values when  $aTc$  is present but trajectories are not likely to reach and stabilize in a region of either high (above 20000), or low (below 1000) concentrations of  $cI$  when  $aTc$  is absent. While from simulations it appears that  $cI$  stabilizes to values above 20000 in the absence of  $aTc$  the stability of the region cannot be guaranteed, possibly as a result of other trajectories originating there. For the second system, concentration of  $lacI$  again stabilize to low values when  $aTc$  is present and in its absence, a small region from the system reaches and remains at high concentrations of  $lacI$  is discovered. Our results indicate that, as constructed, the repressilator systems do not produce the required oscillations and must be tuned further to get this behavior.

For the  $tetR/lacI$  and  $tetR/cI$  toggle switch systems, the existence of two attracting invariant regions cannot be confirmed by our analysis procedure. For the  $tetR/lacI$  neither of the regions tested was identified as stable or attracting. For the  $tetR/cI$  system, the results indicate that all trajectories of the system reach and

remain in a region of high *tetR* and low *cI* concentrations in the absence of *aTc*, and high *cI* and low *tetR* concentrations in the presence of *aTc*. For the *cI/lacI* toggle switch system, our analysis revealed that two attracting invariant sets exist. Further numerical simulations confirmed that the system can settle in two different stable equilibria. Only one of the two equilibria was revealed initially through numerical simulation, due to the small size of the attractor region of the other.

By using our analysis procedure, we were able to invalidate the two repressilator device designs as they did not lead to the expected behavior. In addition, we were able to correctly identify the bistable behavior of the *tetR/cI* system, that was missed initially when exploring the system through simulations. As the set of biological parts available in various repositories increases, device designs must be explored automatically from high level specifications. Our analysis procedures can be applied to computational models of synthetic gene networks to select potentially good device design and filter out bad ones, which can reduce the experimental work required to produce devices that work correctly.

## Chapter 8

# Conclusions and Future Work

The main focus of this work has been the development of a framework that allows properties of synthetic gene networks to be specified formally using rich specifications, resembling natural language. Methods for the fully automatic analysis of realistic models of such systems were developed. Our approach was based on the construction of finite abstractions of infinite systems and expanded the known class of hybrid systems, for which finite quotients can be computed efficiently. Our framework led to the implementation of several software tools for analysis, parameter synthesis, and control of piecewise affine systems from temporal logic specifications. Applications of our analysis procedure to the field of synthetic biology were demonstrated through computational experiments.

The model of a previously constructed transcriptional cascade was used to describe the dynamics of its separate components. Those components were computationally recombined in different ways to construct novel devices, which were tested against specifications of bistability and oscillatory behavior using our methods. We showed that such an approach can be used to find device design that fail to operate as required, which can save costly experimental validation. In addition, we showed that our method can uncover behavior that is missed if only numerical simulations are used to study the models.

As the complexity of devices constructed as part of synthetic biology increases and more intricate functionality is required, tools allowing the automatic analysis of

models against rich specifications will be needed to guide design efforts. A number of biological part databases are currently being developed and there is an ongoing effort to characterize the available parts (*i.e.* provide measurement data that describes their behavior). Similarly to the applications demonstrated in this thesis, such information can be used to explore the large space of potential device designs *in silico*, by constructing mathematical models and analyzing them from high level specifications. This approach can allow researchers to experimentally construct only a small set of devices that are likely to work as required and validate them experimentally. Focusing the experimental efforts to a smaller number of constructs can save costly and time consuming experiments.

Currently available tools for biodesign automation allow parts from repositories to be combined automatically into new devices, with the help of liquid handling robots. As those methods improve, it will become possible to construct a large number of devices with minimal effort. Our methods can add the extra layer of validation that limits the construction efforts to a smaller set of devices that are more likely to work as required. As more parts are characterized and more devices are constructed, the quality of mathematical models will improve and analysis procedures such as the ones developed as part of this thesis will hold more predictive power. There are several extensions that can be implemented in our framework to improve the results further.

The separate methods for analysis, parameter synthesis and control developed in this thesis can be combined into one single procedure. Our analysis method can be used first to identify regions of initial conditions, from which all trajectories of a system satisfy a specification. For non-satisfying regions, our parameter synthesis procedure can tune the system and guarantee its correctness. If non-satisfying regions still remain, control strategies can be found to affect the system from those states

only (the controllers implementing those strategies can come online only when the systems enters a specific region). In this way, the region from which satisfaction of a specification can be guaranteed might be expanded further than if just a single procedure was used.

The most significant extension of the methods described in this thesis would involve transforming them from a nondeterministic into a probabilistic framework. Our current approach does not consider the likelihood of an execution but requires that all trajectories of the system satisfy the specification. As a result, our methods are conservative and a single run that violates the specification is sufficient to invalidate the model from a whole set of initial conditions. This is especially true when parameter uncertainty is considered. Intuitively, there is some distribution over the set of possible parameters and some trajectories are more likely than others. In order to account for such effects, the probability of individual executions must be captured.

A probabilistic framework would allow devices to be compared based on their probability of satisfying a specification. In our current approach, we compare devices based on the relative size of the satisfying (violating) region (the region from which the satisfaction (violation) of the formula is guaranteed). However, even when the sizes of the satisfying region of two devices are similar, their probabilities of satisfying the specification might be very different. Maximizing the probability of satisfying a specification can also be used to tune devices using various search strategies, including genetic algorithms. In addition, probabilistic temporal logics can be used to capture less restrictive specifications. Instead of requiring that *the system oscillates*, which will be violated even if a very unlikely non-oscillating run exists, the slightly weaker *with probability over 90%, the system oscillates* seems more appropriate for biological systems.

There exist tools that can decide whether a finite probabilistic model satisfies

a specification expressed as a probabilistic temporal logic formula. However, the stochastic models used for biological systems are usually infinite, so a strategy for abstraction, similar to the ones used in this thesis in a nondeterministic setting, must be developed.

Additional extensions to the framework described in this thesis can address the complexity of the resulting methods, which limits their application to relatively small genetic networks. Larger systems can be broken up into modules, which are analyzed separately. Methods for guaranteeing the correctness of a system from guarantees of the correctness of its separate modules have been developed in the field of formal verification. For probabilistic systems, methods for statistical model checking consider a subset of possible executions and provide only statistical guarantees of correctness but can handle much larger systems.

Probabilistic versions of the methods from this thesis that can handle larger systems can become valuable tools for the analysis of biological systems. Through integration with other biodesign automation platforms, such tools can provide a complete design solution for the field of synthetic biology. This can decrease the time and cost associated with projects and allow researchers to attempt the construction of more challenging designs, which can lead to more advanced applications.



## References

- Alur, R., Belta, C., Ivancic, F., Kumar, V., Mintz, M., Pappas, G., and Schug, J. (2001). Hybrid modelling and simulation of biomolecular networks. In Di Benedetto, M. and Sangiovanni-Vincentelli, A., editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 19–32. Springer Berlin / Heidelberg.
- Alur, R., Courcoubetis, C., and Dill, D. (1990). Model-checking for real-time systems. In *IEEE Symposium on Logic in Computer Science*, pages 414–425, Philadelphia, PA.
- Alur, R., Courcoubetis, C., Henzinger, T. A., and Ho, P. H. (1993). Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Grossman, R., Nerode, A., Ravn, A., and Rischel, H., editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer Berlin / Heidelberg.
- Alur, R., Dang, T., and Ivancic, F. (2002). Reachability analysis of hybrid systems via predicate abstraction. In Tomlin, C. and Greenstreet, M., editors, *Hybrid Systems: Computation and Control*, volume 2289 of *Lecture Notes in Computer Science*, pages 758–819. Springer Berlin / Heidelberg.
- Alur, R. and Dill, D. L. (1994). A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235.
- Alur, R., Henzinger, T., and Sontag, E., editors (1996). *Hybrid Systems III: Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*. Springer-Verlag.
- Alur, R., Henzinger, T. A., Lafferriere, G., and Pappas, G. J. (2000). Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88:971–984.
- Anderson, J. C., Clarke, E. J., Arkin, A. P., and Voigt, C. A. (2006). Environmentally controlled invasion of cancer cells by engineered bacteria. *Journal of Molecular Biology*, 355:619–627.
- Antoniotti, M. and Mishra, B. (1995). Discrete event models + temporal logic = supervisory controller: Automatic synthesis of locomotion con-

- trollers. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, pages 1441–1446, Nagoya, Aichi, Japan.
- Antoniotti, M., Park, F., Policriti, A., Ugel, N., and Mishra, B. (2003). Foundations of a query and simulation system for the modeling of biochemical and biological processes. In *Proceedings of the Pacific Symposium of Bio-computing (PSB03)*, pages 116–127.
- Antunes, M., Ha, S., Tewari-Singh, N., Morey, K., Trofka, A., Kugrens, P., Deyholos, M., and Medford, J. (2006). A synthetic de-greening gene circuit provides a reporting system that is remotely detectable and has a re-set capacity. *Journal of Plant Biotechnology*, 4:605–622.
- Aziz, A., Shiple, T., Singhal, V., Brayton, R., and Sangiovanni-Vincentelli, A. (2002). Formula-dependent equivalence for compositional CTL model checking. *Formal Methods in System Design*, 21:193–224.
- Baier, C. and Katoen, J.-P. (2008). *Principles of Model Checking*. The MIT Press.
- Barmish, B. and Sankaran, J. (1979). The propagation of parametric uncertainty via polytopes. *IEEE Transactions on Automatic Control*, 24:346–249.
- Barnat, J., Brim, L., and Ročkai, P. (2009). DiVinE 2.0: High-Performance Model Checking. In *Proceedings of the International Workshop on High Performance Computational Systems Biology (HiBi)*, pages 31–32. IEEE Computer Society Press.
- Basu, S. and Kumar, R. (2006). Quotient based approach to control of nondeterministic discrete-event systems with  $\mu$ -calculus specification. In *American Control Conference*, Minneapolis, MN.
- Batt, G., Belta, C., and Weiss, R. (2007a). Model checking liveness properties of genetic regulatory networks. In Grumberg, O. and Huth, M., editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 4424 of *Lecture Notes in Computer Science*, pages 323–338. Springer Berlin / Heidelberg.
- Batt, G., Belta, C., and Weiss, R. (2008). Temporal logic analysis of gene networks under parameter uncertainty. *IEEE Transactions on Circuits and Systems and IEEE Transactions on Automatic Control (joint special issue on Systems Biology)*, 53:215–229.

- Batt, G., Ropers, D., de Jong, H., Geiselman, J., Mateescu, R., Page, M., and Schneider, D. (2005). Validation of qualitative models of genetic regulatory networks by model checking : Analysis of the nutritional stress response in *Escherichia coli*. *Bioinformatics*, 21(Suppl.1):i19–i28.
- Batt, G., Yordanov, B., Weiss, R., and Belta, C. (2007b). Robustness Analysis and Tuning of Synthetic Gene Networks. *Bioinformatics*, 23(18):2415–2422.
- Belta, C., Esposito, J., Kim, J., and Kumar, V. (2005). Computational techniques for analysis of genetic network dynamics. *The International Journal of Robotics Research*, 24(2–3):219–235.
- Belta, C., Finin, P., Habets, L., A, H., Imielinski, M., V.Kumar, and Rubin, H. (2004). Understanding the bacterial stringent response using reachability analysis of hybrid systems. In Alur, R. and Pappas, G. J., editors, *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 107–116. Springer Berlin / Heidelberg.
- Belta, C. and Habets, L. (2004). Constructing decidable hybrid systems with velocity bounds. In *43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas.
- Belta, C., Schug, J., Dang, T., Kumar, V., Pappas, G. J., Rubin, H., and Dunlap, P. V. (2001). Stability and reachability analysis of a hybrid model of luminescence in the marine bacterium *Vibrio fischeri*. In *40th IEEE Conference on Decision and Control*, Orlando, FL.
- Bemporad, A., Garulli, A., Paoletti, S., and Vicino, A. (2003). A greedy approach to identification of piecewise affine models. In Maler, O. and Pnueli, A., editors, *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 97–112. Springer Berlin / Heidelberg.
- Bemporad, A., Giovanardi, L., and Torrisi, F. (2000). Performance driven reachability analysis for optimal scheduling and control of hybrid systems. In *Proceedings of the 39th IEEE Conference Decision and Control*, pages 969–974.
- Bergmann, F. and Sauro, H. (2006). SBW-a modular framework for systems biology. *Proceedings of the 2006 Winter Simulation Conference*, pages 1637–1645.
- Bernot, G., Comet, J.-P., Richard, A., and Guespin, J. (2004). Application of formal methods to biological regulatory networks: Extending Thomas’

- asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3):339–347.
- Bianco, A. and de Alfaro, L. (1995). Model checking of probabilistic and nondeterministic systems. In Thiagarajan, P., editor, *Foundations of Software Technology and Theoretical Computer Science*, volume 1026 of *Lecture Notes in Computer Science*, pages 499 – 513. Springer Berlin / Heidelberg.
- Biofab (2009). The biofab:international open facility advancing biotechnology (biofab). online at <http://www.biofab.org/>.
- Blinov, M. L., Faeder, J. R., Goldstein, B., and Hlavacek, W. S. (2004). BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20(17):3289–91.
- Bouajjani, A., Fernandez, J.-C., and Halbwachs, N. (1991). Minimal model generation. In Clarke, E. and Kurshan, R., editors, *Computer-Aided Verification*, volume 531 of *Lecture Notes in Computer Science*, pages 197–203. Springer Berlin / Heidelberg.
- Brajnik, G. and Clancy, D. (1998). Focusing qualitative simulation using temporal logic: theoretical foundations. *Annals of Mathematics and Artificial Intelligence*, 22(1-2):59–86.
- Braun, D., Basu, S., and Weiss, R. (2005). Parameter estimation for two synthetic gene networks: A case study. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 769–772.
- Brutlag, D. L., Galper, A. R., and Millis, D. H. (1991). Knowledge-based simulation of DNA metabolism: prediction of enzyme action. *Computer Applications in the Biosciences*, 7(1):9–19.
- Cai, Y., Hartnett, B., Gustafsson, C., and Peccoud, J. (2007). A syntactic model to design and verify synthetic genetic constructs derived from standard biological parts. *Bioinformatics*, 23(20):2760–2767.
- Canton, B., Labno, A., and Endy, D. (2008). Refinement and standardization of synthetic biological parts and devices. *Nature Biotechnology*, 26(7):787–93.
- Cases, I. and Lorenzo, V. D. (2005). Genetically modified organisms for the environment: stories of success and failure and what we have learned from them. *International Microbiology*, 8(3):213–222.

- Chabrier-Rivier, N., Chiaverini, M., Danos, V., Fages, F., and Schächter, V. (2004). Modeling and querying biomolecular interaction networks. *Theoretical Computer Science*, 325(1):25–44.
- Chandran, D., Bergmann, F. T., and Sauro, H. M. (2009). TinkerCell: modular CAD tool for synthetic biology. *Journal of Biological Engineering*, 3:19.
- Chutinan, A. and Krogh, B. (1998). Computing approximating automata for a class of linear hybrid systems. In Antsaklis, P., Lemmon, M., Kohn, W., Nerode, A., and Sastry, S., editors, *Hybrid Systems V*, volume 1567 of *Lecture Notes in Computer Science*, pages 637–637. Springer Berlin / Heidelberg.
- Chutinan, A. and Krogh, B. H. (2001). Verification of infinite-state dynamic systems using approximate quotient transition systems. *IEEE Transactions on Automatic Control*, 46(9):1401–1410.
- Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., and Tacchella, A. (2002). NuSMV 2: An opensource tool for symbolic model checking. In Brinksma, E. and Larsen, K., editors, *Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 241–268. Springer Berlin / Heidelberg.
- Clarke, E., Fehnker, A., Han, Z., Krogh, B., Ouaknine, J., Stursberg, O., and Theobald, M. (2003). Abstraction and counterexample-guided refinement in model checking of hybrid systems. *International Journal of Foundations of Computer Science (IJFCS)*, 14(4):583–604.
- Clarke, E. M. M., Peled, D., and Grumberg, O. (1999). *Model checking*. MIT Press.
- Cooling, M. T., Rouilly, V., Misirli, G., Lawson, J. R., Yu, T., Hallinan, J., and Wipat, A. (2010). Standard virtual biological parts: a repository of modular modeling components for synthetic biology. *Bioinformatics*, 26(7):925–31.
- Courcoubetis, C. and Yannakakis, M. (1995). The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907.
- Davoren, J. M., Coulthard, V., Markey, N., and Moor, T. (2004). Non-deterministic temporal logics for general flow systems. In Alur, R. and Pappas, G. J., editors, *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 107–121. Springer Berlin / Heidelberg.

- de Jong, H. (2002). Modeling and simulation of genetic regulatory systems. *Journal of Computational Biology*, 9(1):69–105.
- de Jong, H., Geiselmann, J., Hernandez, C., and Page, M. (2003a). Genetic network analyzer : Qualitative simulation of genetic regulatory networks. *Bioinformatics*, 19(3):336–344.
- de Jong, H., Gouzé, J.-L., Hernandez, C., Page, M., Sari, T., and Geiselmann, J. (2003b). Hybrid modeling and simulation of genetic regulatory networks: A qualitative approach. In Maler, O. and Pnueli, A., editors, *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 267–282. Springer Berlin / Heidelberg.
- Del Vecchio, D., Ninfa, A. J., and Sontag, E. D. (2008). Modular cell biology: retroactivity and insulation. *Molecular Systems Biology*, 4(161):161.
- Densmore, D., Van Devender, A., Johnson, M., and Sritanyaratana, N. (2009). *A platform-based design environment for synthetic biological systems*. ACM Press, New York, New York, USA.
- Eker, S., Knapp, M., Laderoute, K., Lincoln, P., and Talcott, C. (2002). Pathway logic: Executable models of biological networks. *Electronic Notes in Theoretical Computer Science*, 71.
- Elowitz, M. B. and Leibler, S. (2000). A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–8.
- Emerson, E. A. (1985). Automata, tableaux and temporal logics (extended abstract). In *Proceedings of the Conference on Logic of Programs*, pages 79–88.
- Emerson, E. A. (1990). Temporal and modal logic. In van Leeuwen, J., editor, *Handbook of Theoretical Computer Science: Formal Models and Semantics*, volume B, pages 995–1072. North-Holland Pub. Co./MIT Press.
- Endy, D. (2005). Foundations for engineering biology. *Nature*, 438(7067):449–53.
- Fages, F., Soliman, S., and Chabrier-Rivier, N. (2004). Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. *Journal of Biological Physics and Chemistry*, 4(64.73).
- Fainekos, G. E., Kress-Gazit, H., and Pappas, G. J. (2005). Hybrid controllers for path planning: a temporal logic approach. In *IEEE Conference on Decision and Control*, pages 4885–4890, Seville, Spain.

- Ferrari-trecate, G., Muselli, M., Liberati, D., and Morari, M. (2001). A learning algorithm for piecewise linear regression. In *In Marinaro, M. and Tagliaferri, R. (eds.) Neural Nets: WIRN Vietri-01*, pages 114–119. Springer.
- Ferrari-Trecate, G., Muselli, M., Liberati, D., and Morari, M. (2003). A clustering technique for the identification of piecewise affine systems. *Automatica*, 39(2):205–217.
- Fiser, A., Do, R. K. G., and Sali, A. (2000). Modeling of loops in protein structures. *Protein Science*, 9(9):1753–73.
- Francois, P. and Hakim, V. (2004). Design of genetic networks with specified functions by evolution in silico. *Proceedings of the National Academy of Sciences of the United States of America*, 101:580–585.
- Frehse, G., Jha, S., and Krogh, B. (2008). A counterexample-guided approach to parameter synthesis for linear hybrid automata. In Egerstedt, M. and Mishra, B., editors, *Hybrid Systems: Computation and Control*, volume 4981 of *Lecture Notes in Computer Science*, pages 187–200. Springer Berlin / Heidelberg.
- Funahashi, A., Morohashi, M., Kitano, H., and Tanimura, N. (2003). Celldesigner: a process diagram editor for gene-regulatory and biochemical networks. *Biosilico*, 1:159–162.
- Gardner, T., Cantor, C., and Collins, J. (2000). Construction of a genetic toggle switch in *Escherichia coli*. *Nature*, 403:339–342.
- Gasteiger, E., Hoogland, C., Gattiker, A., Duvaud, S., Wilkins, M. R., Appel, R. D., and Bairoch, A. (2005). Protein Identification and Analysis Tools on the ExPASy Server. In Walker, J. M., editor, *The Proteomics Protocols Handbook*, pages 571–607. Humana Press.
- Gastin, P. and Oddoux, D. (2001). Fast ltl to büchi automata translation. In Berry, G., Comon, H., and Finkel, A., editors, *Computer Aided Verification*, volume 2102 of *Lecture Notes in Computer Science*, pages 53–65. Springer Berlin / Heidelberg.
- Geyer, T., Torrisi, F., and Morari, M. (2003). Efficient mode enumeration of compositional hybrid systems. In Maler, O. and Pnueli, A., editors, *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 216–232. Springer Berlin / Heidelberg.

- Ghosh, R., Tiwari, A., and Tomlin, C. (2003). Automated symbolic reachability analysis; with application to delta-notch signaling automata. In Maler, O. and Pnueli, A., editors, *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 233–248. Springer Berlin / Heidelberg.
- Ghosh, R. and Tomlin, C. J. (2004). Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modelling: Delta-Notch protein signalling. *Systems Biology*, 1(1):170–183.
- Gibson, D. G., Glass, J. I., Lartigue, C., Noskov, V. N., Chuang, R. Y., Algire, M. A., Benders, G. A., Montague, M. G., Ma, L., Moodie, M. M., Merryman, C., Vashee, S., Krishnakumar, R., Assad-Garcia, N., Andrews-Pfannkoch, C., Denisova, E. A., Young, L., Qi, Z. Q., Segall-Shapiro, T. H., Calvey, C. H., Parmar, P. P., Hutchison, C. A., Smith, H. O., and Venter, J. C. (2010). Creation of a Bacterial Cell Controlled by a Chemically Synthesized Genome. *Science*, 52.
- Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361.
- Gillespie, D. T. and Mangel, M. (1981). Conditioned Averages in Chemical Kinetics. *Journal of Chemical Physics*, 75:704–709.
- Girard, A. (2007). Approximately bisimilar finite abstractions of stable linear systems. In Bemporad, A., Bicchi, A., and Buttazzo, G., editors, *Hybrid Systems: Computation and Control*, volume 4416 of *Lecture Notes in Computer Science*, pages 231–244. Springer Berlin / Heidelberg.
- Glass, L. (1975). Classification of biological networks by their qualitative dynamics. *Journal of Theoretical Biology*, 54:85–107.
- Goler, J. (2004). BioJADE: a design and simulation tool for synthetic biological systems. online at <http://dspace.mit.edu/handle/1721.1/7115>. AITR-2004-003.
- Grieder, P. (2004). *Efficient computation of feedback controllers for constrained systems*. Ph.d. dissertation, ETH Zürich.
- Hedley, W. J., Nelson, M. R., Bellivant, D. P., and Nielsen, P. F. (2001). A short introduction to CellML. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 359(1783):1073–1089.



- Heemels, W. P. M. H., Schutter, B. D., and Bemporad, A. (2001). Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091.
- Henzinger, T. and Sastry, S., editors (1998). *Hybrid Systems: Computation and Control*, volume 1386 of *Lecture Notes in Computer Science*. Springer.
- Henzinger, T. A., Kopke, P. W., Puri, A., and Varaiya, P. (1998). What is decidable about hybrid automata? *Journal of Computer and System Sciences*, 57:94–124.
- Hill, A. V. (1913). The combinations of haemoglobin with oxygen and with carbon monoxide. *Biochemical Journal*, 7(5):471–480.
- Holzmann, G. (1997). The model checker SPIN. *IEEE Transactions on Software Engineering*, 25(5):279–295.
- Holzmann, G. (2004). *The SPIN Model Checker, Primer and Reference Manual*. Addison-Wesley, Reading, Massachusetts.
- Hooshangi, S., Thiberge, S., and Weiss, R. (2005). Ultrasensitivity and noise propagation in a synthetic transcriptional cascade. *Proceedings of the National Academy of Sciences of the United States of America*, 102(10):3581.
- Horn, F. (2005). Streett Games on Finite Graphs. In the *2nd Workshop on Games in Design and Verification (GDV)*.
- Hucka, M., Finney, A., Sauro, H., Bolouri, H., Doyle, J. C., Kitano, H., Arkin, A. P., Bornstein, B., Bray, D., Cornish-Bowden, A., Cuellar, A. A., Dronov, S., Gilles, E. D., Ginkel, M., Gor, V., Goryanin, I. I., Hedley, W. J., Hodgman, T. C., Hofmeyr, J.-H., Hunter, P. J., Juty, N. S., Kasberger, J. L., Kremling, A., Kummer, U., Le Novère, N., Loew, L. M., Lucio, D., Mendes, P., Minch, E., Mjolsness, E. D., Nakayama, Y., Nelson, M. R., Nielsen, P. F., Sakurada, T., Schaff, J. C., Shapiro, B. E., Shimizu, T. S., Spence, H. D., Stelling, J., Takahashi, K., Tomita, M., Wagner, J., and Wang, J. (2003). The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531.
- Jiang, S. and Kumar, R. (2006). Supervisory control of discrete event systems with CTL\* temporal logic specifications. *SIAM Journal on Control and Optimization*, 44(6):2079–2103.
- Juloski, A., Wieland, S., and Heemels, W. P. M. H. (2005). A bayesian approach to identification of hybrid systems. *IEEE Transactions on Automatic Control*, 50(10):1520–1533.

- Kanellakis, P. C. and Smolka, S. A. (1990). CCS expressions, finite-state processes, and three problems of equivalence. *Information and Computation*, 86:43–68.
- Karaman, S., Sanfelice, R. G., and Frazzoli, E. (2008). Optimal control of mixed logical dynamical systems with linear temporal logic specifications. In *Proceedings of the 47th IEEE Conference on Decision and Control (CDC)*, pages 2117–2122.
- Kauffman, S. A. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22:437–467.
- Kelly, J. R., Rubin, A. J., Davis, J. H., Ajo-Franklin, C. M., Cumbers, J., Czar, M. J., de Mora, K., Gliberman, A. L., Monie, D. D., and Endy, D. (2009). Measuring the activity of BioBrick promoters using an in vivo reference standard. *Journal of Biological Engineering*, 3:4.
- Klein, J. and Baier, C. (2006). Experiments with deterministic  $\omega$ -automata for formulas of linear temporal logic. *Theoretical Computer Science*, 363(2):182–195.
- Kloetzer, M. and Belta, C. (2006a). A fully automated framework for control of linear systems from ltl specifications. In Hespanha, J. and Tiwari, A., editors, *Hybrid Systems: Computation and Control*, volume 3927 of *Lecture Notes in Computer Science*, pages 333–347. Springer Berlin / Heidelberg.
- Kloetzer, M. and Belta, C. (2006b). Ltl planning for groups of robots. In *IEEE International Conference on Networking, Sensing, and Control*, Ft. Lauderdale, FL.
- Kloetzer, M. and Belta, C. (2008a). Dealing with non-determinism in symbolic control. In *Hybrid Systems: Computation and Control*, volume 4981 of *Lecture Notes in Computer Science*, pages 287–300. Springer Berlin / Heidelberg.
- Kloetzer, M. and Belta, C. (2008b). A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297.
- Knight, T. (2003). Idempotent vector design for standard assembly of bio-bricks. *MIT Synthetic Biology Working Group*.
- Kuipers, B. (1981). Qualitative simulation. *Artificial intelligence*, 29:289–388.

- Kvasnica, M. (2008). *Efficient Software Tools for Control and Analysis of Hybrid Systems*. PhD thesis, Swiss Federal Institute of Technology (ETH) Zurich.
- Kvasnica, M., Grieder, P., and Baotić, M. (2004). Multi-Parametric Toolbox (MPT).
- Lafferriere, G., Pappas, G., and Yovine, S. (1999a). A new class of decidable hybrid systems. In Vaandrager, F. and van Schuppen, J., editors, *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 137–151. Springer Berlin / Heidelberg.
- Lafferriere, G., Pappas, G. J., and Sastry, S. (2000). O-minimal hybrid systems. *Mathematics of Control, Signals and Systems*, 13(1):1–21.
- Lafferriere, G., Pappas, G. J., and Yovine, S. (1999b). Reachability computation for linear hybrid systems. In *Proceedings of the 14th IFAC World Congress*, pages 7–12, Beijing, P.R.C.
- LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press.
- Le Novère, N., Bornstein, B., Broicher, A., Courtot, M., Donizelli, M., Dharuri, H., Li, L., Sauro, H., Schilstra, M., Shapiro, B., Snoep, J. L., and Hucka, M. (2006). BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Research*, 34(Database issue):D689–91.
- Le Novère, N., Finney, A., Hucka, M., Bhalla, U. S., Campagne, F., Collado-Vides, J., Crampin, E. J., Halstead, M., Klipp, E., Mendes, P., Nielsen, P., Sauro, H., Shapiro, B., Snoep, J. L., Spence, H. D., and Wanner, B. L. (2005). Minimum information requested in the annotation of biochemical models (MIRIAM). *Nature Biotechnology*, 23(12):1509–15.
- Levine, M. and Tjian, R. (2003). Transcription regulation and animal diversity. *Nature*, 424(6945):147–51.
- Lin, J.-N. and Unbehauen, R. (1992). Canonical piecewise-linear approximations. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 39(8):697–699.
- Lloyd, C. M., Lawson, J. R., Hunter, P. J., and Nielsen, P. F. (2008). The CellML Model Repository. *Bioinformatics*, 24(18):2122–3.
- Loizou, S. G. and Kyriakopoulos, K. J. (2004). Automatic synthesis of multiagent motion tasks based on LTL specifications. In *IEEE Conference on Decision and Control*, volume 1, pages 153–158.

- Longabaugh, W. J., Davidson, E. H., and Bolouri, H. (2005). Computational representation of developmental genetic regulatory networks. *Developmental Biology*, 283(1):1 – 16.
- Lynch, N. and Krogh, B. H., editors (2000). *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin.
- McAdams, H. H. and Shapiro, L. (1995). Circuit simulation of genetic networks. *Science (New York, N.Y.)*, 269(5224):650–6.
- Meijering, E. (2002). A chronology of interpolation: From ancient astronomy to modern signal and image processing. In *Proceedings of the IEEE*, volume 90, pages 319–342.
- Memon, M. A. (2003). Computational logic: Linear-time vs. branching-time logics. Graduate Course Notes, Simon Fraser University, Burnaby, Canada.
- Mestl, T., Plathe, E., and Omholt, S. W. (1995). Periodic solutions in systems of piecewise-linear differential equations. *Dynamics and stability of systems*, 10(2):179–193.
- Milner, R. (1989). *Communication and concurrency*. Prentice-Hall, Englewood Cliffs, NJ.
- Mitchell, J. C. (2007). Discrete uniform sampling of rotation groups using orthogonal images. *SIAM Journal of Scientific Computing*, 30(1):525–547.
- Monteiro, P. T., Ropers, D., Mateescu, R., Freitas2, A. T., and de Jong, H. (2008). Temporal logic patterns for querying dynamic models of cellular interaction networks. *Bioinformatics*, 24(16):i227–i233. 10.1093/bioinformatics/btn275.
- Motzkin, T., H.Raiffa, Thompson, G., and R.M.Thrall (1953). The double description method. In Kuhn, H. and Tucker, A., editors, *Contributions to Theory of Games*, volume 2. Princeton University Press.
- Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S. (1993). An approach to the description and analysis of hybrid systems. In Grossman, R., Nerode, A., Ravn, A., and Rischel, H., editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 149–178. Springer Berlin / Heidelberg.
- Olivier, B. G. and Snoep, J. L. (2004). Web-based kinetic modelling using JWS Online. *Bioinformatics*, 20(13):2143–4.

- Pappas, G. J. (2003). Bisimilar linear systems. *Automatica*, 39(12):2035–2047.
- Peccoud, J., Blauvelt, M. F., Cai, Y., Cooper, K. L., Crasta, O., DeLalla, E. C., Evans, C., Folkerts, O., Lyons, B. M., Mane, S. P., Shelton, R., Sweede, M. A., and Waldon, S. A. (2008). Targeted development of registries of biological parts. *PloS one*, 3(7):e2671.
- Piterman, N. and Pnueli, A. (2006). Faster solutions of rabin and streett games. In *Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 275–284, Washington, DC, USA. IEEE Computer Society.
- Puri, A. and Varaiya, P. (1994). Decidability of hybrid systems with rectangular differential inclusions. In Dill, D., editor, *Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*, pages 95–104. Springer Berlin / Heidelberg.
- Purnick, P. E. M. and Weiss, R. (2009). The second wave of synthetic biology: from modules to systems. *Nature Reviews. Molecular Cell Biology*, 10(6):410–22.
- Quottrup, M. M., Bak, T., and Izadi-Zamanabadi, R. (2004). Multi-robot motion planning: A timed automata approach. In *IEEE International Conference on Robotics and Automation*, pages 4417–4422, New Orleans, LA.
- Rialle, S., Felicori, L., Dias-Lopes, C., Pérès, S., El Atia, S., Thierry, A. R., Amar, P., and Molina, F. (2010). BioNetCAD: design, simulation and experimental validation of synthetic biochemical networks. *Bioinformatics*, 26(18):2298–304.
- Riedel, M. D. (2010). The Next Frontier in Design Automation : Bio-Design Automation (BDA). *DAC.COM Knowledge Center Article*.
- Rizk, A., Batt, G., Fages, F., and Soliman, S. (2008). On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In Heiner, M. and Uhrmacher, A., editors, *Computational Methods in Systems Biology*, volume 5307 of *Lecture Notes in Computer Science*, pages 251–268. Springer Berlin / Heidelberg.
- Ro, D.-K., Paradise, E. M., Ouellet, M., Fisher, K. J., Newman, K. L., Ndungu, J. M., Ho, K. A., Eachus, R. A., Ham, T. S., Kirby, J., Chang, M. C. Y., Withers, S. T., Shiba, Y., Sarpong, R., and Keasling, J. D.

- (2006). Production of the antimalarial drug precursor artemisinic acid in engineered yeast. *Nature*, 440:940–943.
- Rodrigo, G., Carrera, J., and Jaramillo, A. (2007). Asmparts: assembly of biological model parts. *Systems and Synthetic Biology*, 1(4):167–70.
- Roll, J., Bemporad, A., and Ljung, L. (2004). Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40(1):37–50.
- Rosenfeld, N., Young, J., Alon, U., Swain, P., and Elowitz, M. (2007). Accurate prediction of gene feedback circuit behavior from component properties. *Molecular Systems Biology*, 3(143).
- Rosenfeld, N., Young, J. W., Alon, U., Swain, P. S., and Elowitz, M. B. (2005). Gene regulation at the single-cell level. *Science (New York, N. Y.)*, 307(5717):1962–5.
- Safra, S. (1988). On the complexity of omega-automata. In *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 319–327.
- Safra, S. (1989). *Complexity of automata on infinite objects*. PhD thesis, The Weizman Institute of Science, Rehovot, Israel.
- Salis, H. M., Mirsky, E. a., and Voigt, C. a. (2009). Automated design of synthetic ribosome binding sites to control protein expression. *Nature Biotechnology*, 27(10):946–50.
- Savage, D. F., Way, J., and Silver, P. A. (2008). Defossilizing fuel: how synthetic biology can transform biofuel production. *ACS Chemical Biology*, 3:13–16.
- Schlipf, T., Buechner, T., Fritz, R., Helms, M., and Koehl, J. (1997). Formal verification made easy. *IBM Journal of Research and Development*, 41(4-5):567–576.
- Shults, B. and Kuipers, B. (1997). Proving properties of continuous systems: Qualitative simulation and temporal logic. *Artificial Intelligence*, 92(1-2):91–130.
- Sontag, E. (1981). Nonlinear regulation: The piecewise linear approach. *IEEE Trans. Automat. Contr.*, 26(2):346–358.
- Tabuada, P. (2006). Symbolic control of linear systems based on symbolic subsystems. *IEEE Transactions on Automatic Control*, 51(6):1003–1013.

- Tabuada, P. and Pappas, G. (2003). Model checking LTL over controllable linear systems is decidable. In Maler, O. and Pnueli, A., editors, *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 498–513. Springer-Verlag.
- Tabuada, P. and Pappas, G. (2006). Linear time logic control of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 51(12):1862–1877.
- Thomas, R. (1991). Regulatory networks seen as asynchronous automata: a logical description. *Journal of Theoretical Biology*, 153:1–23.
- Thomas, W. (2002). Infinite games and verification. In Brinksma, E. and Larsen, K., editors, *Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 58–65. Springer Berlin / Heidelberg.
- Tiwari, A. and Khanna, G. (2002). Series of abstractions for hybrid automata. In Tomlin, C. and Greenstreet, M., editors, *Hybrid Systems: Computation and Control*, volume 2289 of *Lecture Notes in Computer Science*, pages 425–438. Springer Berlin / Heidelberg.
- Tůmová, J., Yordanov, B., Belta, C., Černá, I., , and Barnat, J. (2010). A symbolic approach to controlling piecewise affine systems. In *Proceedings of the 49th IEEE Conference on Decision and Control*, pages 4230–4235, Atlanta, GA.
- Tuttle, L. M., Salis, H., Tomshine, J., and Kaznessis, Y. N. (2005). Model-driven designs of an oscillating gene network. *Biophysical Journal*, 89:3873–3883.
- Vaandrager, F. and van Schuppen, J., editors (1999). *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*. Springer.
- Vardi, M. (1999). Probabilistic linear-time model checking: An overview of the automata-theoretic approach. In Katoen, J.-P., editor, *Formal Methods for Real-Time and Probabilistic Systems*, volume 1601 of *Lecture Notes in Computer Science*, pages 265–276. Springer Berlin / Heidelberg.
- Vardi, M. (2001). Branching vs. linear time: Final showdown. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 2031 of *Lecture Notes in Computer Science*, pages 1–22. Springer-Verlag, London, UK.

- Vardi, M. Y. and Wolper, P. (1986). An automata-theoretic approach to automatic program verification. In *Proceedings of the IEEE Symposium on Logic in Computer Science (LICS)*, pages 332–344, Washington, DC, USA. IEEE Computer Society.
- Vidal, R., Soatto, S., Ma, Y., and Sastry, S. (2003). An algebraic geometric approach to the identification of a class of linear hybrid systems. In *Proceedings of the 42nd IEEE Conference on Decision and Control*.
- Villalobos, A., Ness, J. E., Gustafsson, C., Minshull, J., and Govindarajan, S. (2006). Gene Designer: a synthetic biology tool for constructing artificial DNA segments. *BMC Bioinformatics*, 7:285.
- Wolper, P., Vardi, M., and Sistla, A. (1983). Reasoning about infinite computation paths. In et al., E. N., editor, *IEEE Symposium on Foundations of Computer Science*, pages 185–194, Tucson, AZ.
- Yordanov, B. and Belta, C. (2008a). Formal analysis of piecewise affine systems under parameter uncertainty with application to gene networks. In *Proceedings of the American Control Conference*, pages 2767–2772.
- Yordanov, B. and Belta, C. (2008b). Parameter synthesis for piecewise affine systems from temporal logic specifications. In Egerstedt, M. and Mishra, B., editors, *Hybrid Systems: Computation and Control*, volume 4981 of *Lecture Notes in Computer Science*, pages 542–555. Springer Berlin / Heidelberg.
- Yordanov, B. and Belta, C. (2009). Temporal logic control of discrete-time piecewise affine systems. In *Proceedings of the 48th IEEE Conference on Decision and Control*, pages 3182–3187, Shanghai, China.
- Yordanov, B. and Belta, C. (2010). Formal analysis of discrete-time piecewise affine systems. *IEEE Transactions on Automatic Control*, 55(12):2834–2841.
- Yordanov, B., Belta, C., and Batt, G. (2007). Model checking discrete time piecewise affine systems: application to gene networks. In *European Control Conference (ECC)*, Kos, Greece.
- Yordanov, B., Tůmová, J., Belta, C., Černá, I., , and Barnat, J. (2010). Formal analysis of piecewise affine systems through formula-guided refinement. In *Proceedings of the 49th IEEE Conference on Decision and Control*, pages 5899–5904, Atlanta, GA.



Zhang, G. and Ignatova, Z. (2009). Generic algorithm to predict the speed of translational elongation: implications for protein biogenesis. *PloS one*, 4(4):e5036.

Zuker, M. (2003). Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Research*, 31(13):3406–3415.

## Boyan Y. Yordanov

---

CONTACT INFORMATION Department of Biomedical Engineering  
 Boston University *Cell:* (508) 579-3543  
 HyNeSs Lab *Fax:* (617) 353-5548  
 15 Saint Mary's Street, Room 145 *E-mail:* yordanov@bu.edu  
 Brookline, MA 02446 *Web:* hyness.bu.edu/yordanov

RESEARCH INTERESTS I am interested in studying how the computation performed by biological systems leads to properties that make life possible. I focus on exploring genetic regulatory networks designed as part of the field of synthetic biology, where the complexity is manageable. I use mathematical models based on hybrid systems, which I analyze formally using abstraction and model checking techniques.

EDUCATION **Boston University**, Boston, Massachusetts USA

Ph.D., Biomedical Engineering, January 2011

- Dissertation: *A Formal Framework for Analysis and Design of Synthetic Gene Networks*
- Research Adviser: Professor Calin Belta

M.S., Biomedical Engineering, September 2009

- Research Adviser: Professor Calin Belta

**Clark University**, Worcester, Massachusetts USA

*Magna cum Laude*, Concentration in Bioinformatics.

B.A., Biochemistry and Molecular Biology, June 2005

- High Departmental Honors
- Thesis: *Initial Conformational Changes of Transthyretin Triggering Amyloid Formation: Molecular Dynamics Simulation under Low pH Conditions.*
- Research Adviser: Professor Shuanghong Huo

B.A., Computer Science, June 2005

- High Departmental Honors
- Thesis: *Implementation of the Pebble Game Algorithm for Flexibility Analysis of Proteins.*
- Research Adviser: Professor Li Han

- AWARDS
- Annual Center for Information and Systems Engineering (CISE) Award (Honorable Mention) 2007, 2008,
  - Whitaker Fellowship, 2005,
  - Outstanding Academic Achievement in Computer Science Award, 2004,
  - James'39 and Ada Bickman Summer Science Research Internship Award, 2004.
- JOURNAL PAPERS AND BOOK CHAPTERS
- B. Yordanov**, J. Tůmová, C. Belta, I. Černá, J. Barnat, Temporal Logic Control of Discrete-Time Piecewise Affine Systems, *in press*, 2010.
- B. Yordanov**, C. Belta, Formal Analysis of Discrete-Time Piecewise Affine Systems, In *Transactions on Automatic Control*, volume 55(12), pages 2834-2841, 2010.
- B. Yordanov**, C. Belta, Parameter Synthesis for Piecewise Affine Systems from Temporal Logic Specifications, In *Lecture Notes in Computer Science*, volume 4981, pages 542-555, 2008.
- G. Batt, **B. Yordanov**, R. Weiss, C. Belta, Robustness Analysis and Tuning of Synthetic Gene Networks, In *Bioinformatics*, volume 23(18), pages 2415-2422, 2007.
- M. Yang, **B. Yordanov**, Y. Levy, R. Brüschweiler, S. Huo, The Sequence-Dependent Unfolding Pathway Plays a Critical Role in the Amyloidogenicity of Transthyretin, In *Biochemistry*, volume 45, 2006.
- M. Yang, M. Lei, **B. Yordanov**, S. Huo, Peptide Plane Can Flip in Two Opposite Directions: Implication in Amyloid Formation of Transthyretin, In *The Journal of Physical Chemistry B*, volume 110, 2006.

- CONFERENCE PROCEEDINGS **B. Yordanov**, J. Tůmová, C. Belta, I. Černá, J. Barnat, Formal Analysis of Piecewise Affine Systems through Formula-Guided Refinement, In *Proceedings of the 49th IEEE Conference on Decision and Control, Atlanta, GA*, pages 5899-5904, 2010
- J. Tůmová, **B. Yordanov**, C. Belta, I. Černá, J. Barnat, A Symbolic Approach to Controlling Piecewise Affine Systems, In *Proceedings of the 49th IEEE Conference on Decision and Control, Atlanta, GA*, pages 4230-4235, 2010.
- B. Yordanov**, C. Belta, Temporal Logic Control of Discrete-Time Piecewise Affine Systems, In *Proceedings of the 48th IEEE Conference on Decision and Control, Shanghai, China*, 2009.
- B. Yordanov**, C. Belta, Parameter Synthesis for Piecewise Affine Systems from Temporal Logic Specifications, In *11th International Conference on Hybrid Systems: Computation and Control, St. Louis, MO*, 2008.
- B. Yordanov**, C. Belta, Formal Analysis of Piecewise Affine Systems under Parameter Uncertainty with Application to Gene Networks, In *Proceedings of the 2008 American Control Conference, Seattle, WA*, pages 2767-2772, 2008.
- B. Yordanov**, C. Belta, G. Batt, Model Checking Discrete Time piecewise Affine Systems: Application to Gene Networks, In *European Control Conference, Kos, Greece*, 2007.

CONFERENCE  
ABSTRACTS  
AND POSTERS

**B. Yordanov**, C. Belta, Formal Approaches to Analysis, Synthesis, and Control of Synthetic Gene Networks, In *Boston University Science and Engineering Research Symposium*, 2010.

**B. Yordanov**, C. Belta, Formal Analysis and Parameter Synthesis for Piecewise Affine Systems with Applications to Gene Networks, In *4th Northeast Control Workshop*, University of Maryland, College Park, MD, April 26–27, 2008.

**B. Yordanov**, C. Belta, Formal Analysis and Parameter Synthesis for Piecewise Affine Systems, In *Boston University Science and Engineering Research Symposium*, 2008.

**B. Yordanov**, C. Belta, Formal Analysis of Piecewise Affine Systems under Parameter Uncertainty with Application to Gene Networks, In *4th Annual RECOMB Satellite on Regulatory Genomics*, 2007.

**B. Yordanov**, C. Belta, G. Batt, Model Checking Discrete-Time Piecewise Affine Systems: Applications to Gene Networks, In *Boston University Science and Engineering Research Symposium*, 2007.

G. Batt, **B. Yordanov**, R. Weiss, and C. Belta, Robustness Analysis and Tuning of Synthetic Gene Networks. In *Journées Ouvertes Biologie, Informatique et Mathématiques (JOBIM)*, Marseille, France, 2007

G. Batt, **B. Yordanov**, R. Weiss, C. Belta, Robustness Analysis and Tuning of Synthetic Gene Networks. In *Proc. Synthetic Biology 3.0*, 2007.

G. Batt, **B. Yordanov**, R. Weiss, C. Belta, Robustness Analysis and Tuning of Synthetic Gene Networks with Parameter Uncertainties. In *Engineering Principles in Biological Systems*, Cold Spring Harbor, NY, 2006

SOFTWARE  
TOOLS

**B. Yordanov** and C. Belta, FaPAS: Formal Analysis of Piecewise Affine Systems under Parameter Uncertainty.

**B. Yordanov** and C. Belta, ParSyPAS: Parameter Synthesis for Piecewise Affine Systems.

**B. Yordanov** and C. Belta, conPAS: Temporal Logic Control of Discrete-Time Piecewise Affine Systems.

**RESEARCH  
EXPERIENCE**

Graduate Research Assistant, 2006 - 2010:

HyNeSs Lab, Boston University, Boston MA  
PI: Professor Calin Belta

Undergraduate Research Assistant, 2004 - 2005:

Biochemistry Dept., Clark University, Worcester MA,  
PI: Professor Shuanghong Huo

Undergraduate Research Assistant, 2004 - 2005:

Computer Science Dept., Clark University, Worcester MA,  
PI: Professor Li Han

**TEACHING  
EXPERIENCE**

Teaching Assistant

Computer Science Dept., Clark University, Worcester MA,  
Introductory computer science courses (CSCI101, CSCI102)

Teaching Assistant

Bioinformatics, Clark University, Worcester MA,  
Introductory Bioinformatics course (BINF101)

**ADDITIONAL  
TRAINING**

Responsible Conduct of Research, 2010

Laboratory Safety Training, 2008