Algorithmic Algebraic Model Checking: Hybrid Automata and Systems Biology

by

Venkatesh Pranesh Mysore

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

May 2006

Bud Mishra

To Amma and Appa

—

Also dedicated to the memory of "Balsu Sir" Prof. Balasubramanian

Acknowledgments

I always thought that the *Acknowledgements* section typically read the same, and often found the words artificial, contrived and forced, as if to satisfy a requirement. However, it was only when I began thinking about writing my thesis that I truly appreciated the inner calling to acknowledge. One cannot but bewilder at the improbability of events that had to all occur to make such an accomplishment as a Doctorate in Philosophy possible. I am now filled with an urge to thank the many persons and events who played a part in making this momentous achievement possible. The list is long, but certainly incomplete – so apologies in advance for all omissions.

I would first like to express my deepest and warmest gratitude to my thesis committee – Prof. Bud Mishra, Prof. Amir Pnueli and Prof. Jane Hubbard, for providing the research direction that was fundamental to the development of most of the ideas discussed in this thesis. They paid attention to my long presentations patiently, and encouraged me to work harder and with greater rigor, by providing critical and timely feedback. The breadth and depth of my research – I owe completely to their experience and wisdom. Prof. Mishra's incredible breadth of knowledge and his tremendous grasp of the state of the art of so many disconnected fields has been extremely inspiring. He played a foundational role in conceiving a multitude of ideas during our discussions, some of which have found a home in my thesis. Similarly, I am honored to have had Prof. Pnueli, an ACM Turing Award winner, in my thesis committee. His humility, simplicity, humor and reassuring spirit, not to mention his model checking courses, saw me through my Ph.D. He goaded and assisted me in much of my work, though often indirectly and implicitly. I was equally fortunate with my third advisor – Prof. Hubbard, another prolific researcher with an unassuming nature. Though I could not exploit the opportunity to learn from her extensively during my Ph.D., Prof. Hubbard was responsible for my choosing to work on metabolic networks, and indirectly ensured that I address my original Systems Biology motivation.

I would like to thank the NYU Bioinformatics Group for providing a vibrant research environment, and providing me financial support throughout my study. Prof. Mishra again deserves a special mention for maintaining such a stimulated set of individuals from diverse departments working on different problems, all under one roof, in one floor in fact. The opportunity to work full-time before my graduation and all the travel awards are gratefully acknowledged. New York University's Computer Science Department has also been extremely gracious in providing me research and teaching assistantships whenever required. It is a privilege that is so often taken for granted. I am also grateful to them for having given me this opportunity to pursue and realize my scientific ambitions in such a prestigious academic institution, and for having given me a chance to interact with the eminent faculty of the Courant Institute of Mathematical Sciences, and to benefit from their remarkable array of courses. Another scholar pivotal to the development of the algebraic model checking ideas was Prof. Carla Piazza of the University of Udiné. Her sharp critiques of my research drafts were often the seeds of newer and better thought-through ideas. Dr. Marco Antoniotti was also instrumental in cementing my interest in model checking. Prof. Vijay Saraswat's guidance and encouragement towards the end of the thesis work was particularly motivating. On a more informal note, all members of the NYU Bioinformatics Group, particularly Ofer Gill, Mathias Heynman and Giuseppe Narzisi, deserve a mention for their research input, random discussions, group meetings and of course, many a hearty laugh. Again, Dr. Salvatore "Toto" Paxia's perennial technical assistance cannot go unacknowledged.

On a very personal note, I'd like to thank my two greatest role models – my father Dr. M.B. Pranesh and my mother Smt. Lakshmi Pranesh. As no apotheosis will suffice, I will just remark that I owe everything completely to them. I am extremely fortunate to have the blessings of my grandmother Smt. Subhadra Venkateswaran. On the practical day-to-day level, my wife Meenakshi deserves no ordinary praise for her astounding patience and tolerance, for being a perennial source of love, wit, joy and excitement, and most importantly, for making my years as a graduate student the most memorable time of my life. Rutgers University gave us a lovely home, and NJ-Transit & PATH ensured I got to work on time! I would also like to thank my entire family, especially my brothers Mr. Madukar & Mr. Jayanth, my sister-in-law Mrs. Kavita, my divine niece Ms. Janani Sumitra Jayanth, my in-laws Sri. Venkatesan & Smt. Jayashree Venkatesan, my uncle Dr. Raghunath and my sister-in-law Prof. Archana for their continuous supply of happiness, moral support, backing, guidance and encouragement, for their advice, interest, inspiration, e-mail forwards and not to forget, their never-fading confidence in me! I could not have been luckier on the familial front, and all my family members are veritably responsible for everything that I am. Clichéd as such statements might be, they are nevertheless genuine and heartfelt.

Finally, I feel incredibly lucky to have received such a fantastic education all my life. From my foundational years at St.John's Besant Nagar to my high-schooling at P.S.Senior, my memories of my teachers in Madras continue to brim with warmth and respect. Without any doubt, it was the IIT coaching classes offered by Prof. Balasubramanian, Prof. Ananthan, Prof. Santhanam, Prof. Govindarajan and Prof. K.S. Ramachandran that helped me realize my full potential, permanently enhancing my academic life. The four undergrad years that followed at IIT Kharagpur were easily the most relaxed, liberating and rewarding. However, it was Prof. Thomas Anantharaman, then at the University of Wisconsin at Madison, who was solely responsible for my choosing to continue for my Ph.D. As his research assistant during my Mas-

ters, I was overwhelmed by his brilliance and humility. It was he who recommended that I continue my Ph.D. in his colleague's lab at NYU.

Abstract

The field of Systems Biology strives to hasten our understanding of the fundamental principles of life by adopting a global systems-level approach for the analysis of cellular function and behavior. One central problem is modeling, simulation and analysis of biochemical pathways, leading to the characterization of innate and emergent properties. *Hybrid Automata* have established themselves as an ideal framework for capturing the dynamics of such networks of interacting biochemical species. The reactions are captured in terms of the concentrations of the biochemicals, with their evolution governed primarily by the laws of chemical kinetics. Discrete states capture regimes of cellular behavior with different biochemical species and reactions predominating, while transitions capture the biochemical conditions under which a new set of flow equations operate. Our goal in this thesis is to aid Systems Biology research by improving the current understanding of hybrid automata, by developing techniques for symbolic rather than numerical analysis of the dynamics of biochemical networks modeled as hybrid automata, and by honing the theory to two classes of problems: kinetic mass action based simulation in genetic regulatory / signal transduction pathways, and pseudo-equilibrium simulation in metabolic networks.

Hybrid automata become decidable for the reachability problem, i.e., amenable to useful symbolic analysis, only when their expressive freedom is appropriately curtailed. We first inspect and refine the boundary between decidable and undecidable subclasses of hybrid automata. We provide new constructions to characterize the minimal extensions necessary for decidable classes like the 2-dim Piecewise Constant Derivative (PCD) system to become equivalent to the "open" 2-dim Hierarchical PCD (HPCD) class, and for the "open" 2-dim HPCD class to become undecidable. How-

ever, the low dimensionality constraint that must be met for this class to be decidable makes it unsuitable for modeling chemical reactions. Our quest for semi-decidable subclasses leads us to polynomial hybrid systems, which we expand to define the "semi-algebraic" subclass. This is the widest hybrid automaton subclass amenable to rigorous symbolic temporal reasoning, and is sufficiently expressive to capture most Systems Biology. We begin with the bounded reachability problem, and then show how the dense-time temporal logic Timed Computation Tree Logic (TCTL) can be model-checked by exploiting techniques from real algebraic geometry, primarily real quantifier elimination. We also prove the undecidability of reachability in the real Turing Machine formalism. To help overcome the double exponential complexity of real quantifier elimination, we then develop approximation strategies for semi-algebraic hybrid systems by extending bisimulation partitioning, rectangular grid-based approximation and polytopal approximation. We identify well-behaved subclasses and develop new optimizations. We also document the simplifications resulting from discretizing time.

Having developed a rigorous well-defined class of hybrid automata amenable to algebraic model checking, we return to the Systems Biology domain. We develop a uniform algebraic framework for modeling biochemical and metabolic networks. We extend the flux balance analysis based approach for equilibrium characterization. We thus translate these real-world problems into a form that can exploit the techniques we previously developed for semi-algebraic hybrid systems. We present some preliminary results using a prototypical tool Tolque. It is a symbolic algebraic dense time model-checker for semi-algebraic hybrid automata, which uses Qepcad for quantifier elimination.

The techniques developed in this thesis present a theoretically grounded mathematically sound platform for powerful symbolic temporal reasoning over biochemical networks and other semi-algebraic hybrid automata. It is hoped that by building upon this thesis, along with the development of computationally efficient quantifier elimination algorithms and the integration of different computer algebra tools, scientific software systems will emerge to fundamentally transform the way biochemical networks (and other hybrid automata) are investigated and understood.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Systems Biology

Advances in biotechnology and genetic engineering have enabled biochemists and geneticists to perform a vast array of experiments rapidly, resulting in the generation of a huge amount of biological data [228, 148]. Organizations such as the Gene Ontology Consortium [131], the Systems Biology Markup Language team [154], the Kyoto Encyclopedia of Genes and Genomes (KEGG) [165], and the Alliance for Cellular Signaling [3] face the gargantuan task of organizing the information in the best possible way [172]. The task of managing these databases (for a list see [53]) is in itself a huge computational challenge, as discussed in the reviews by Navarro et al. [228] and Deville et al. [105]. Several software systems and standards have been developed for creating, updating and managing this information about cellular pathways, events and behaviors (e.g., Pathway Tools [167], Patika [104], CellML [198]).

However, there is an increasing lag between the rate of *data collection* and the rate of *information extraction*. One of the biggest challenges facing the scientific community now is to excavate the biological truths which lie obfuscated in the distributed results of diverse experiments. There are two ways in which computer scientists can aid this revolution in Biology. The first line of research involves performing statistical analysis, pattern recognition and other machine learning operations on the data. The second set of tasks that Computer Scientists have at hand is simulation-based analysis, i.e., model and simulate biochemical systems using computers to test hypotheses, validate predictions and suggest experiments for Biochemists to carry out. The relatively new terms *Bioinformatics* and *Systems Biology*, originally denoting these two categories respectively [157, 171, 106], are now being used ambiguously, further befuddled by "Computational Biology", "Computational Chemistry" and occassionally, "Theoretical Biology".

Kirschner [170], who remarks that "Scientific fields, like species, arise by descent with modification", gives us a comprehensive definition of Systems Biology:

> "*Systems biology is the study of the behavior of complex biological organization and processes in terms of the molecular constituents. It is built on molecular biology in its special concern for information transfer, on physiology for its special concern with adaptive states of the cell and organism, on developmental biology for the importance of defining a succession of physiological states in that process, and on evolutionary biology and ecology for the appreciation that all aspects of the organism are products of selection, a selection we rarely understand on a molecular level. Systems biology attempts all of this through quantitative measurement, modeling, reconstruction, and theory.*"

Liu [197] describes Systems Biology as being "integrative biology with the ultimate goal of being able to predict de novo biological outcomes given the list of the components involved" ([230] is an exemplary instance), and emphasizes the computational requirements: the ability to digitalize biological output, computational power for analysis, and algorithms for integrating heterogeneous data. Several insightful articles critically evaluating the definition, scope and potential of "Systems Biology" are continuing to appear [231, 194, 253, 284, 93, 197, 69, 68]. Uniformly, it is hoped that eventually there will be computational systems that will have access to all published biological truths (possibly encoded in their models), allowing Biologists to experiment in an entirely new way (see [169] for a glimpse into the future). Facts not patent may be extracted through the analysis of simulations, verification by wet-lab experiments, and iterative model refinement (see *Figure* 3.1 and *Figure* 3.2).

The Problem with Biological Problems

Biological problems do not always invite mathematical treatment. The reason is twofold: (1) There seems to be no inherent formal structure to many of the phenomena, beyond some basic cause-and-effect rules; (2) The experimental data, despite being superfluous and redundant in some cases, is almost always insufficient and incomplete.

The technological advancements have propelled the discovery of biochemical truths at the micro and macro level, all leading up to the conclusion that the assumption about the lack of inherent formal structure is not warranted. However, since this attitude change is relatively recent[1], much of the current body of biological facts remains analytically uninterpreted in its entirety. Another dimension of the problem surfaces due to the dynamic nature of cellular behavior, i.e., the time-variation in cellular composition and function, in addition to growth, mitosis, meiosis and motility. Clearly, a mathematically well grounded approach for expressing the current hypothe-

[1] "That we are at a crossroads in how to explore biology is not at all clear to many." [170]

ses formally, and checking their mutual consistency automatically is needed to refine, revise and improve our understanding of biology[2]. Further, the logical consistency of the hypothesized "axioms of biology" needs to be verified at all time instants of the cellular life cycle. Thus, tools like model checking of temporal logic properties are imperative, if this formalization of a "theoretical biology" is to be attempted.

1.2 Modeling and Analyzing Biochemical Systems

As Aderem[2] points out, "Network biology is in its infancy – future needs range from the development of new theoretical methods to characterize network topology, to insights into the dynamics of motif clusters and biological function". Analysis of the dynamical properties of complex systems has been performed before in control theory applications, VLSI circuit verification, program verification, robotics, etc. In this thesis, we explore how biological processes could be subject to similar analyses [101, 156] and develop the theoretical framework necessary for representing biochemical networks, for describing their temporal properties and for verifying them algebraically. Though the focus is on solidifying the theoretical component, we ensure that we develop decidable algorithms and build a software tool that implements some of the key concepts of our approach.

Modeling using Hybrid Automata

A dynamical system is defined by a set of mathematical rules, usually ordinary differential equations, describing the continuous evolution of the system's variables with time. When there are different sets of rules operating at different sets of conditions

[2] In this thesis, "Biology" stands only for cellular and subcellular biochemical processes, and deals only with the dynamical behavior of interacting biochemicals in a specific pathway.

(the notion of multiple "discrete states"), or when there are variables that change instantaneously or discontinuously (the notion of "discrete variables" and "discrete transitions"), the system is set to exhibit both continuous and discrete dynamical properties. Such a system is a "hybrid" dynamical system, and its formal representation is termed a hybrid automaton [147, 285].

Example 1.2.1 *Consider the repressor mechanism common in molecular biology. A protein P is produced at a rate p and consumed at a rate c. Its repressor protein R is always produced at a rate r. When the repressor concentration R exceed a certain "cutoff" level R_c, it begins to repress the production of protein P. As a result of this consumption in the repressing reaction, the repressor concentration eventually drops below a different cutoff R_{c_2}, and the system resumes production of protein P. This simple abstraction of a common biochemical machinery is modeled using a two-state hybrid automaton in* Figure *1.1.*

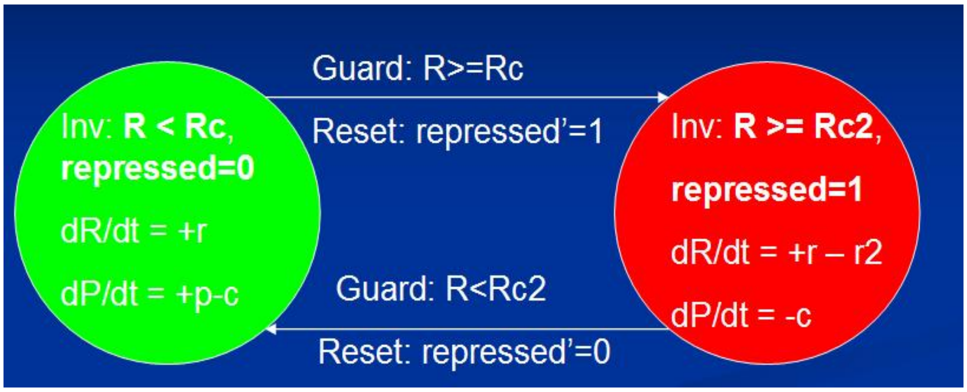

Figure 1.1: Repressor Hybrid Automaton

Hybrid automata are a very natural way of capturing networks of interacting biochemicals [6, 213]. As seen in *Fig.* 1.1, the different discrete states correspond to different regimes of behavior where different species predominate, leading to different approximations. The flow equations follow from the laws of chemical kinetics, while the discrete transitions capture the specific conditions leading to a state change, with a possible reset of variables. Hybrid automata have been very popular in many engineering disciplines like VLSI, control systems and robotics.

Model-Checking Temporal Logic Properties

The biggest advantage of formally capturing a dynamical system such as a biochemical network as a hybrid system is that the formal reasoning tools developed for hybrid automata may be exploited. In particular, the technique for proving that a hybrid automaton satisfies a certain temporal property may be automated for certain subclasses. The language for formally expressing such queries is temporal logic, and the procedure for automatically answering the queries is termed model checking[203, 87, 216]. In addition, if the model checking procedure can answer queries involving symbolic[3] parameters, it becomes an extremely powerful means of reasoning about the behavior of a system.

1.3 Motivation and Purpose

Mathematical analysis of network structure, classification of networks into functional modules, and *de novo* network design will be the keys in understanding Biology through network analysis [4]. In this thesis, an approach to the modeling and analysis of biochemical processes is presented. The ideas are developed in logical sequence and are integrated under the umbrella of "Algorithmic Algebraic Model Checking" (AAMC) [239, 222, 221, 225].

[3] In this thesis, "symbolic" and "algebraic" denote the retaining of variables and parameters as symbols rather than numbers.

Over the last decade or so, hybrid automata have emerged as one of the most suitable formalisms for modeling biochemical networks [102]. Hybrid automata have been investigated from a theoretical computer science perspective for over two decades [11, 14, 8, 15, 12]. The only way of analyzing a general hybrid automaton is via numerical simulation. Researchers have constrained hybrid automata in different ways to enable efficient analysis. In some cases, even parametric analysis becomes possible. However, it is not clear what the most general subclass is that is amenable to symbolic analysis.

1.3.1 The Theory of Hybrid Automata

A fundamental temporal property of a system is reachability: "is a certain end state reachable from a certain initial state?". Decidability of reachability is a tangible way of understanding how unwieldy a subclass is. If it is decidable, it means that any reachability query that can ever be asked about any member of the subclass can always be answered, i.e., algorithmically verified. If it is undecidable, it means that we might encounter queries which we may never be able to answer. Semi-decidability means that there exists a decision procedure which will terminate when the query is true, but may never terminate if the query is false.

What is Decidable?

We first try to see if we can identify a decidable hybrid automaton subclass that is expressive enough to capture biochemical systems. The literature quickly reveals that simple two and three dimensional hybrid automata can become undecidable [38]. While one could argue that the artificial constructs that lead to undecidability can never occur in natural biochemical networks, we recall our original goal: to identify the most general theoretical solution, rather than study an efficient subclass. Our first contribution in this thesis is the sharper characterization of decidable subclasses of hybrid systems, by refining the boundary between decidable and undecidable subclasses [223]. We present new restrictions and extensions of the open subclass Hierarchical Piecewise Constant Derivative (HPCD) systems[39], which characterize its proximity to decidability and undecidability.

What is Semi-Decidable?

Having found HPCDs unsuitable for biochemical systems (due to their low dimensionality requirement), we review more general subclasses that are decidable. Linear systems are not suitable because quadratic terms are predominant in kinetic mass action based flow equations. O-minimal systems [185] are excellent in terms of continuous dynamics, but their discrete transitions are extremely restrictive. Then we encounter polynomial hybrid systems [121, 122] that seem most appropriate. We find that it has already been shown how many temporal logic queries over polynomial systems can be solved by transforming them into real quantifier elimination problems, which are decidable. We rigorously extend this class in an attempt to characterize the broadest subclass that is amenable to symbolic analysis.

The second major contribution of this thesis is the introduction of a new, broad, mathematically well-founded subclass, that has the complexity necessary to capture most Biochemistry accurately. In this subclass "Semi-Algebraic Hybrid Automata" [239], we restrict the expressions appearing in the automaton description to be semi-algebraic, i.e., Boolean combinations of polynomial equations and inequalities. This allows the utilization of the sophisticated mathematical machinery already developed in the field of *Real Algebraic Geometry* [212, 214]. We show that bounded reachability in non-zeno[4] semi-algebraic hybrid automata is decidable [239]. As an aside, we discuss the Blum-Shub-Smale model[66] of real computation, and prove that the hybrid automaton reachability problem is undecidable even in that more powerful framework. We then evaluate semi-algebraic hybrid automata in the context of methods in literature that rely on quantifier elimination and other symbolic techniques.

Endless Possibilities

Having characterized a suitable subclass, we then try to assess its "analyzability". There are several planes along which the reachability analysis can be extended: Is it possible to solve all model checking queries over some temporal logic? Is it possible to handle dense time, or is time discretization an implicit part of the algorithm developed for the reachability problem? Does the algebraic procedure support an enhanced query vocabulary with non-standard operators and expressions? We address these motivating questions by showing that the dense-time logic Timed Computation Tree Logic (TCTL), enhanced with new expressions, is semi-decidable in general, and decidable in the bounded-time non-zeno case [222]. We talk about the relevance of these assumptions in the biological context, and discuss related literature. This work represents a significant step forward in terms of the applicability of symbolic methods, and establishes semi-algebraic hybrid automata as an extremely powerful framework.

Approximating Decidability

Having shown that a powerful technique for general hybrid automata exists, the issue that naturally emerges is its practicability and computational complexity. Since several approximation approaches have been developed for simpler subclasses, we

[4] A "zeno" dynamical system is one where infinite computation can happen in zero time

investigate whether any of them be applied to semi-algebraic hybrid automata. We show how several existing techniques for approximation like bisimulation partitioning, using rectangular grids, using polytopes, and time discretization can be applied to the semi-algebraic domain by non-trivial extension [221]. We also identify some well-behaved subclasses and present optimizations of these approximate methods.

We show how bisimulation partitioning can be applied to semi-algebraic hybrid automata. In the process, we also see how linearity leads to decidability, while monotonicity guarantees approximate decidability. Another popular technique for approximation is over / under approximating by rectangular grids or polyhedra. Here, rather than trying to simplify the continuous flow equations, we try to see if we can reduce the complexity of the semi-algebraic sets obtained at each iteration. In the case of grids, we keep track of unions of numerical intervals, while in polyhedra, we bound the set using linear terms (lines, planes, etc.). Another more trivial kind of approximation is time discretization, where points sampled at periodic time-points are used to interpret a temporal query. We discuss how the time-step of integration can be made large, without disallowing the intermediate discrete transitions. We also point out how discretization can be applied in conjunction with the theory developed earlier and in combination with other approximation methods. We conclude by discussing when and where the different approximation strategies are likely to work, what sort of speed-up is possible, etc. We also point out other techniques that remain to be extended to the semi-algebraic domain.

1.3.2 Reducing Biology to a Dynamical System

Having developed the theory, we return to our motivating problems in Systems Biology. While the basic case of biochemicals interacting as per kinetic mass action has already been extended to the algebraic domain, we find that metabolic networks, which have special properties, have not been handled algebraically. We develop a general algebraic framework for biochemical and metabolic networks [225]. We then show how metabolic networks, which have special properties, can be characterized algebraically. We show how fast and slow reactions can be handled effectively, and also show how the equilibrium description can be extracted from both the kinetic mass action description and flux balance analysis.

1.3.3 Tolque: A Prototype

We conclude with a brief look at a prototypical implementation: Tolque [222, 224]. It is a dense-time symbolic algebraic model-checker for semi-algebraic hybrid systems, that uses Qepcad[151] for quantifier elimination. The double-exponential complexity of quantifier elimination [90] results in Tolque being impractical for large problems. However, the ability to address diverse problems algebraically and uniformly becomes evident in the examples. We demonstrate the tool's performance on the one-cell and two-cell Delta-Notch protein-interaction network, and some other simple examples.

1.4 Organization of the Thesis

In this thesis, we suggest a possible confluence of the theory of hybrid automata and the techniques of algorithmic algebra to create a computational basis for systems biology. We start by discussing our basis for this choice, as we also recognize its power and limitations. Admittedly, the work described here is built on biological foundations that can be faulted for being simple and abstract. We address those issues and present approximation schemes for simplifying the procedure. We conclude by discussing how to extend the methodology proposed here, and use a prototypical implementation to analyze the issues in scaling up.

The consistent and cogent techniques that are described in this thesis lay the foundation of a new research topic: Algorithmic Algebraic Model Checking. In the next two chapters, we review the fundamentals of the three fields of study that are relevant to AAMC: Systems Biology, Hybrid Systems and Model Checking. In particular, *Chapter* 2 introduces *Hybrid Automata* and *Temporal Logic*, while *Chapter* 3 provides the relevant biochemistry, along with a critical review of the various schemes for modeling and analyzing biochemical systems. Having surveyed such a diverse set of sciences, we then present in *Chapter* 4 an analysis of the open subclasses of hybrid automata. We start with decidable 2-dimensional Piecewise Constant Derivative systems (PCDs) and discuss relevant research about extending them to create the open subclass 2-dim Hierarchical PCD. Our primary contribution here is the refinement of the proximity of the HPCD class to decidability and undecidability. Effectively, a clearer picture of the computational power of hybrid automata emerges.

This chapter lays a good foundation for *Chapter* 5, where we introduce a new subclass – "Semi-Algebraic Hybrid Systems". Semi-algebraic hybrid automata represent the broadest subclass of hybrid systems that are amenable to rigorous symbolic mathematical analysis. We summarize the results from real algebraic geometry which enable these computations. We discuss the bounded reachability problem over semi-algebraic hybrid automata, and also prove the undecidability of reachability in the Real Turing Machine formalism. In *Chapter* 6, we solve the algebraic dense-time model-checking problem over this class. TCTL is used to pose algebraic temporal logic queries which are then reduced to iterations of the *one-step until* operator. That operator is shown to be decidable, and hence most fragments of TCTL become semi-decidable. Further, we show how the vocabulary of TCTL can be expanded without any modification to the computation procedure. In *Chapter* 7, we address the double exponential computational complexity of quantifier elimination by exploring approximation schemes. We show how other techniques applicable for simpler subclasses can be extended to the semi-algebraic domain. We characterize well-behaved subclasses and develop new optimizations. We also document the simplifications that result from discretizing the hybrid system analysis. Proofs omitted for clarity and brevity in chapters 4 and 7 are provided in the *Appendix*[5]. Having worked out the theory necessary for model checking hybrid automata efficiently, we return to our motivating problems in Systems Biology in *Chapter* 8. We describe the procedure for modeling a generic biochemical system, and then present in detail the algebraic techniques for the subclass of metabolic networks. We show how the multi-time-scale and pseudo-equilibrium properties of these networks can be exploited to perform efficient algebraic analysis. We demonstrate how the algebraic equilibrium description can be obtained from direct kinetic mass action models and also from flux balance analysis. In *Chapter* 9, we present the prototypical implementation "Tolque". It is a symbolic algebraic dense time model checker that can solve semi-algebraic TCTL queries over semi-algebraic hybrid automata. We document some examples popular in the formal verification and in the Systems Biology communities. We conclude with a discussion in *Chapter* 10, where a critical analysis of the contributions of the thesis is presented. We discuss the several lines of future work that merit attention, and summarize our perspective.

[5]Abbreviations frequently used in the thesis have been tabulated in *Table* .1 in *Section C* of the *Appendix*

Chapter 2

Background: Model Checking Hybrid Automata

The technical background necessary for this thesis falls under several broad categories: (1) *Systems Biology* provides the motivation, introducing us to the problems we are interested in solving; (2) *Hybrid Automata* provide the theoretical framework necessary for modeling these problems of interest; (3) *Temporal Logic* is the means of mathematically expressing the properties of the hybrid automata that we want to prove or disprove; (4) *Model Checking* is the process of verifying the temporal logic query over the hybrid automaton model of the original Biological problem; (5) *Real Algebraic Geometry* becomes necessary to perform the model checking for the new expressive subclass of hybrid sutomata that we introduce in this thesis.

In this chapter, we only introduce the fundamentals of temporal logic, model checking and hybrid automata. The other concepts will be introduced in the appropriate chapters, so as to better motivate their relevance to the thesis [1].

[1] The interested reader can peruse *Sections* 5.1.2 and 8.4.1for an introduction to computational real algebraic geometry

2.1 Hybrid Systems

A continuous dynamical system involves a set of variables evolving over time as per a set of laws. Typically, these laws are differential equations. In addition to these continuous dynamics, many dynamical systems have discrete properties. Further, the system evolves as per different continuous dynamics in different modes of operation. To capture this confluence of continuous and discrete dynamics, the general framework of hybrid automata [2] was defined [147, 285].

A hybrid automaton can be used to approximate a complex non-linear system in terms of a model that is partly discrete and partly continuous. It can be understood as a directed graph of discrete states and transitions, which allows arbitrary: (1) "invariant" expressions dictating when the system can remain within a discrete state; (2) continuous dynamics (e.g., differential equations / inclusions) in the "flow" expressions, in each discrete state (continuous evolution with time); (3) conditions controlling when each transition can be taken, in the "guard" expressions; (4) expressions describing the change in the values of the variables in the "reset" relations, during each discrete state transition (instantaneous discrete evolution). A computation of a hybrid automaton is a series of continuous evolution steps of arbitrary time-length each, interspersed with an arbitrary number of zero time-length discrete transition steps, where no continuous or dicrete step violates any of the constraints of the automaton. Hybrid automata may be formally described thus:

Definition 2.1.1 Hybrid Automata. *[239, 222]* *A k-dimensional* hybrid automaton *is a 7-tuple, $H = (Z, V, E, Init, Inv, Flow, Jump)$, consisting of the*

[2] A hybrid *automaton* typically refers to the mathematical model, while a hybrid *system* typically refers to the physical dynamical system that happens to exhibit both continuous and discrete dynamics.

following components:

- $Z = \{Z_1, \ldots, Z_k\}$ *a finite set of variables ranging over the reals* $\mathbb{R}$*;* $\dot{Z} = \{\dot{Z}_1, \ldots, \dot{Z}_k\}$ *denotes the first derivatives with respect to the time* $t \in \mathbb{R}$ *during continuous change;* $Z' = \{Z'_1, \ldots, Z'_k\}$ *denotes the set of values at the end of a discrete change;*

- (V, E) *is a directed graph; the vertices of* V *are called* control modes*, the edges of* E *are called* control switches*;*

- *Each vertex* $v \in V$ *is labeled by "initial", "invariant" and "flow" labels:* $\mathit{Init}(v)$*,* $\mathit{Inv}(v)$*, and* $\mathit{Flow}(v)$*; the labels* $\mathit{Init}(v)$ *and* $\mathit{Inv}(v)$ *are constraints whose free variables are in* Z*; the label* $\mathit{Flow}(v)$ *is a constraint whose free variables are in* $Z \cup \dot{Z}$*;*

- *Each edge* $e \in E$ *is labeled by "jump" conditions:* $\mathit{Jump}(e)$*, which is a constraint whose free variables are in* $Z \cup Z'$*.* □

Let H be a hybrid automaton of dimension k. For any given control mode $v \in V$, we denote with $\Phi(v)$ the set of functions from $\mathbb{R}^+$ to $\mathbb{R}^k$ satisfying the constraints in $\mathit{Flow}(v)$. In addition, for any given $r \in \mathbb{R}^k$, we use $\mathit{Init}(v)(r)$ ($\mathit{Inv}(v)(r)$ and $\mathit{Flow}(v)(r)$) to denote the Boolean value obtained by pairwise substitution of r with Z in $\mathit{Init}(v)$ ($\mathit{Inv}(v)$ and $\mathit{Flow}(v)$, respectively). Similarly, for any given r, $s \in \mathbb{R}^k$, we use $\mathit{Jump}(e)(r, s)$ to denote the Boolean value obtained by pairwise substitution of r with Z and s with Z' in $\mathit{Jump}(e)$. The semantics of hybrid automata can now be given in terms of execution traces as in the definition below.

Definition 2.1.2 Semantics of Hybrid Automata. *[239, 222]* *Let* $H = (Z, V, E, \mathit{Init}, \mathit{Inv}, \mathit{Flow}, \mathit{Jump})$ *be a hybrid automaton of dimension* k*.*

A location ℓ *of* H *is a pair* $\langle v,\ r\rangle$, *where* $v \in V$ *is a state and* $r \in \mathbb{R}^k$ *is an assignment of values to the variables of* Z. *A location* $\langle v,\ r\rangle$ *is said to be* admissible *if* $Inv(v)(r)$ *is satisfied.*

The continuous reachability transition relation, $\xrightarrow[\mathcal{C}]{t}$, *between admissible locations is defined as follows:*

$$\langle v, r\rangle \xrightarrow[\mathcal{C}]{t} \langle v, s\rangle$$
$$\textit{iff} \qquad \exists f \in \Phi(v)\Big(f(0) = r \ \wedge\ f(t) = s \ \wedge\ \forall t' \in [0, t](Inv(v)(f(t')))\Big).$$

The discrete reachability transition relation, $\xrightarrow[\mathcal{D}]{0}$, *between admissible locations is defined as follows:*

$$\langle v, r\rangle \xrightarrow[\mathcal{D}]{0} \langle u, s\rangle \qquad \textit{iff} \qquad \langle v, u\rangle \in E \ \wedge\ Jump(\langle v, u\rangle)(r, s)$$

A trace *of* H *is a sequence* $\ell_0, \ell_1, \ldots, \ell_n, \ldots$ *of admissible locations such that*

$$\forall i \geq 0 \ \ \ell_i \xrightarrow[\mathcal{C}]{t_i} \ell_{i+1} \ \vee\ \ell_i \xrightarrow[\mathcal{D}]{0} \ell_{i+1}. \quad \Box$$

Effectively, a hybrid automaton may be understood as a set of variables evolving as per the "flow" equations (continuous dynamics) in the initial state. At a certain point in time, certain "guard" conditions are satisfied; the hybrid system now has the choice of taking a discrete transition to a new state, where the variables will evolve as per a new set of flow equations. As a result of such a transition, the variables of the system may get reset to new values, described by the "reset" relation. Alternatively, the system can choose to continue in the current discrete state, and at any later time instant, take one of the many discrete transitions whose guards are all satisfiable. However, the system cannot endlessly remain in the current discrete state: it is forced to jump out of it when it no longer satisfies the state's "invariant". This broad definition imposes no restrictions on the nature of the functions that can appear in the flow, invariant, guard or reset relations. Clearly, the only possible means of "analysis" in this case is numerical simulation.

The natural course of research has been identifying subclasses of general hybrid systems that are amenable to efficient analysis. Subclasses like timed automata, rectangular automata, piece-wise constant derivative systems and hierarchical piece-wise constant derivative systems are elaborated in *Sec.* 4.2. Other well-studied subclasses like multirate automata, linear systems, time-invariant systems, O-minimal systems and polynomial systems have been discussed in detail in *Sec.* 5.1.1. Semi-algebraic hybrid automata, the new class introduced in this thesis, is defined in *Sec.* 5.2. The reader may also benefit from the pictorial representations of different types of hybrid automata in *Figures* 1.1, 4.1, 4.2, 5.1, 5.2, 5.3, 9.1 and 9.2.

2.2 Temporal Logic

Model checking using temporal logic is a standard tool for analysis in diverse engineering disciplines like hardware design, embedded systems, communication protocols, computer security, and more recently biochemical pathways. *Model Checking* over Kripke structures [178] arose as a viable method for the automatic verification of artificial systems within the Electrical Engineering community.

Temporal logic (TL) with its *linear* (LTL) and *branching* (CTL) time variants has been used successfully to verify the behavior of several *discrete systems* modeling many applications from engineering [203, 87, 216]. Model checking over *hybrid systems* has also made great advances in the last decade [7, 11, 12, 266]. The discrete time logics CTL and LTL have been extended to dense time to facilitate characterization of real time and hybrid systems. For linear time temporal logics, the extensions include Metric Temporal Logic (MTL) [14], Timed Propositional Temporal Logic (TPTL) [15], Real-Time Temporal Logic (RTTL) [232], Explicit-Clock Temporal Logic (ECTL) [140] and Metric Interval Temporal Logic (MITL) [13]. The branching time extensions include Real-Time Computation Tree Logic (RTCTL) [116] and Timed Computation Tree Logic (TCTL) [7].

Since TCTL and CTL will be referenced in the course of this thesis, they have been summarized in *Sec.* 6.2 and *Sec.* 7.5.1 respectively. A summary of relevant model-checking / formal-verification approaches for hybrid automata may be found in *Sec.* 5.5, *Sec.* 6.4 and *Sec.* 7.6, with a detailed survey of tools in *Sec.* 9.2.

Chapter 3

Modeling and Analyzing Biochemical Systems

In this chapter, we attempt to understand the problem of modeling and analyzing biochemical systems. This problem is central to the field of Systems Biology. This chapter represents the motivation behind this thesis, rather than being an "application" example. We establish the following points: (1) Systems Biology is a very promising field that uses the mathematical and computational sciences to extract a systems-level understanding of the cellular and subcellular processes underlying all life; (2) Modeling, simulating and analyzing biochemical networks is one of the fundamental methodologies employed by Systems Biologists; (3) *Hybrid Automata* present an excellent platform for capturing the dynamical behavior of biochemical networks; (4) Symbolic rather than numerical methods hold the key to unlocking deeper and more fundamental biological principles, and could potentially shed some light on whether there indeed exists a universal "theory of life". We introduce cellular biology and biochemistry, and then present the different modeling frameworks in existence. We then detail some real biochemical examples before discussing the types of analyses that can be performed.

3.1 The Biology of Cells

We first introduce the three broad categories into which the vast array of cellular biochemical processes may be organized: Genetic Regulation, Metabolism and Signal Transduction.

3.1.1 Genetic Regulation

The oft repeated "central dogma of biology" states that biochemical information in cells is encoded primarily in the Deoxyribo Nucleic Acid (DNA) molecules. DNA gets *transcribed* into messenger Ribo Nucleic Acid (mRNA), and the mRNA then gets *translated* into proteins at the ribosomes. Genetic regulation is the process of modulation of the expression of the relevant genes at the correct locations and times. Proteins, the products of genes, themselves partake in this genetic regulatory process, thus giving rise to complex interaction networks. As transcription factors, some of these proteins interact with regions of the DNA to effect these changes. The binding of the transcription machinery and the transcription factors to the DNA involves complex protein-DNA-protein interactions, where, more often than not, the structural modification of the DNA (such as euchromatin and heterochromatin regions) and the protein have to be considered.

The rate of gene transcription, the post-transcriptional mechanisms that affect mRNA half-life / stability and the formation of the mRNA-ribosome complex are other aspects of genetic regulation. Similarly, there are post-translational mechanism for protein modification such as phosphorylation of key residues, multimerization, chaperone-guided complex formation, protein-folding control, and genetic control by small interfering RNA.

3.1.2 Signal Transduction

The cell responds to external signals through receptors, which may be on its surface or in its cytoplasm. The signal is transmitted to the interior through messengers which induce the desired response to the external signal. Typically, a ligand binds to a trans-membrane receptor whose conformation subsequently changes. This change is detected by proteins bound to it (usually on the cytoplasmic side), or is manifested as a change in the receptor's chemical properties. Subsequently, second messenger molecules amplify the signal and communicate it to the target(s). Alternatively, the ligand can directly enter the cell through non-specific channels and then bind to the receptors inside the cell. Small molecules like Calcium often participate in these pathways, where most of the reactants are enzymatic proteins. The net result of the signal transduction pathway is an appropriate response by the specific subcellular component. Very often, the signaling pathway results in the nuclear localization of transcription factors, leading to the transcription (or shutting down) of corresponding genes. The binding of the signaling molecule with the receptor, the modification of the structure of the receptor and associated proteins (with the receptor sometimes acting as an enzyme) and dispatching of second messengers are the activities near the cell membrane. Receptor desensitization, internalization and regeneration are other complex sub-processes, adding to the physical aspects like binding and diffusion.

3.1.3 Metabolism

Metabolism[1] represents almost all processes that are not genetic regulatory or signal transducing. The gigantic set of biochemicals needed by the cell are continuously produced and consumed by complex enzyme catalyzed pathways. These comprise the metabolic network. They essentially govern the matter and energy cycles of a cell – the way energy and matter are obtained, transformed and consumed by living organisms. Photosynthesis for example is the process by which light energy is converted into chemical energy during sugar (e.g., glucose) formation. During respiration, the oxidation of glucose transforms the energy into Adenosine Tri-Phosphate (ATP). While the ATP-cycle and photosynthesis comprise the well-known energy metabolism, carbohydrate metabolism deals with Glycolysis and Phosphates, lipid metabolism pertains to Triacyl Glycerol and Fatty Acids and amino acid metabolism mostly refers to Glutamate and Urea.

Other Processes

There are still more aspects of cellular biology beyond this simple tri-fold characterization. These include the biophysics of DNA packaging, protein folding and DNA-protein interaction, cell adhesion, non-transcriptional regulatory pathways, cellular compartments and related spatio-temporal phenomena, cell proliferation, and cell migration.

3.2 The Chemistry of Life

We follow the introduction to Biology with an introduction to chemical kinetics.

[1] *Chapter* 8 deals with the modeling and analysis of metabolic networks.

3.2.1 Chemical Kinetics

The fundamental law of chemical kinetics states that the rate of a reaction is proportional to the concentration of the reactants. The law can be understood by thinking about the situation at a molecular level: the probability of a reaction occurring is directly proportional to the probability of the reactant molecules coming physically close together. The chances of such a meeting are clearly dependent on the number of reactant molecules present in a region, in other words, the product of the concentrations. Its implications are summarized below: [2]

Theorem 3.2.1 ***The Law of Kinetic Mass Action*** *[91, 168, 289]*

- *For the reaction $aA + bB \longleftrightarrow cC + dD$, the rate of the forward reaction $v_f \equiv k_f[A]^a[B]^b$ and the rate of the backward reaction $v_b \equiv k_b[C]^c[D]^d$, where k_f and k_b are the forward and backward rate constants respectively.*

- *The rates of the reactions are related to the rate of individual reactants via the number of molecules as: $\frac{1}{c}\dot{C} = \frac{1}{d}\dot{D} = -\frac{1}{a}\dot{A} = -\frac{1}{b}\dot{B} = (v_f - v_b)$.*

- *If the chemical system is at equilibrium at a given temperature, then the following ratio is a constant termed the equilibrium constant: $\frac{[C]^c[D]^d}{[A]^a[B]^b} = \frac{k_f}{k_b} = K_{eq}$, which is often rephrased as $k_f[A]^a[B]^b = k_b[C]^c[D]^d$.*

- *If the system is not at equilibrium, the ratio $\frac{k_f}{k_b}$ is different from the equilibrium constant K_{eq}. In such cases, the ratio is called the* reaction quotient, *designated Q.*

- *A system not at equilibrium tends to equilibrium i.e., $Q \rightarrow K_{eq}$.* $\square$

[2] The square brackets "[ ]" around the chemical species denote their concentrations. The rate of a reaction is also referred to as its "flux".

The kinetic mass action (KMA) law thus directly yields ordinary differential equations (ODEs) governing the rate of a reaction, in terms of the concentrations and rate constants.

The *stoichiometric matrix* is an alternate form of representing the system of ODEs, where all rate terms are replaced by corresponding flux variables. This representation is particularly useful when all derivatives are zero (see *Chapter* 8), thus reducing the system of polynomial equations to a linear system.

Note 3.2.1 *In nature, the rate of otherwise slow reactions is increased dramatically by the action of catalysts. Catalysts (typically enzymes) lower the activation energy of a reaction, thus increasing the number of reactant molecules likely to interact and produce product. The kinetics of enzyme mediated reactions is often expressed using specialized equations that represent the solution of the simultaneous equations involving the enzymes and the reactants.*

3.2.2 The Cell as a System of ODEs

Thus, we can write down *kinetic mass-action* equations for the time variation of the concentrations of the biochemical species in a cell, in the form of a system of *ordinary differential equations*. If one knew all the species x_i involved in any one pathway, the mass-action equations for the system could be expressed in the following form

$$\dot{x}_i = f_i(x_1, x_2, \ldots, x_n), \qquad\qquad i = 1, 2, \ldots, n.$$

Each equation above is an algebraic differential equation consisting of two algebraic terms, a positive term representing synthesis and a negative term representing degradation (see *Defn.* 8.3.1 for a more explicit formalization).

Example 3.2.1 *The transcription process can be described by equations of the* Hill *type, with its Hill coefficient n depending on the* cooperativity *among the transcription binding sites. If the concentrations of DNA and RNA are denoted by x_M, y_M, etc., and those of proteins by x_P, y_P, etc., then the relevant equations are of the form:*

$$\begin{aligned}\dot{x}_M &= -k_1 x_M + k_3 \frac{1+\theta y_P^n}{1+y_P^n} \\ \dot{x}_P &= -k_2 x_P + k_4 x_M\end{aligned}$$

where the superscripted dots denote the time-derivatives. For both RNA and DNA, the degradation is represented by a linear function; for RNA, synthesis through transcription is a highly nonlinear but a rational Hill-type function; and for proteins, synthesis through translation is a linear function of the RNA concentration. In the equation for transcription, y_P denotes the concentration of proteins involved in the transcription initiation of the DNA, k_1 and k_2 are the forward rate constants of the degradation of RNA and proteins respectively, k_3 and k_4 are the rate constants for RNA and protein synthesis and θ models the saturation effects in transcription. It is to be observed that when $n = 1$, the equation closely resembles the Michaelis-Menten *equations that describe the rate of enzyme mediated reactions at high substrate concentration.*

3.3 Modeling Biology

Bork and Serrano [69] emphasize that Systems Biology aims at a quantitative understanding of biological systems, with activities ranging "from collections of physiological data with quantified molecular parts lists to abstract mathematical modeling of biological processes". With the advent of powerful computers and the development of Genetic Engineering and Biotechnology in the last quarter of the twentieth century, a vast amount of raw data (such as the DNA sequence of different species, the time profiles of mRNA and protein expression levels, and protein structures) started becoming available. Building on the earlier research on cellular composition and function, the field of Bioinformatics sprouted to process this raw data, using ideas from statistics and machine learning; an offshoot was the field of Systems Biology, which aimed to model, simulate and analyze biochemical pathways [157, 171, 106]. It was hoped that a systems level approach would aid the elucidation of deeper truths about cellular biology. Subsequently, this categorization / nomenclature dissolved, with "Bioinformatics", "Systems Biology" and "Computational Biology" all being liberally generalized. For instance, Aderem [2] summarizes Systems Biology as being hypothesis driven, global, quantitative, iterative, integrative, and dynamic; these characteristics could well apply to traditional Bioinformatics or any of the sister-fields.

Clearly, the field of Systems Biology is still at a state where its very meaning and scope are being debated (also see *Sec.* 1.1). A reasonable picture is obtained just by observing the titles of key articles published on the topic in 2005: *on the inherent ambiguity of the categorization* – "The Meaning of Systems Biology" [170], "Systems biology: in the broadest sense of the word" [284] and "Is there biological research beyond Systems Biology? A comparative analysis of terms" [68]; *on the scope of the field* – "Fundamental issues in systems biology" [231], "Systems Biology: Its Practice and Challenges" [2] and "Systems Biology, Integrative Biology, Predictive Biology" [197]; *on its potential and future* – "Systems biology: where it's at in 2005" [194], "Systems Biology: Will it work?" [253], "Systems biology may work when we learn to understand the parts in terms of the whole" [93] and "Towards Cellular Systems in 4D" [69].

The problem of modeling, simulating and analyzing biochemical processes is without doubt one of the foci of this young and evolving field. The different players and processes in the biochemical domain may be represented at different levels of abstraction[101, 156]. Some of the major approaches are summarized below.

3.3.1 Logical Modeling

The state of the reactant is captured through a finite number of abstract-states (where intermediate expression levels are assumed to have the same behavior), and functions are used to describe the new states (concentration range) of the chemical species, given their old states. The transitions between states can be assumed to occur synchronously or (more accurately) asynchronously. In the simplest case, only two states ("on" and "off") are used, and Boolean algebra is used to describe the dynamics. Concurrent Transition Systems [80, 79] and Pathway (Rewrite) Logic [114] are good expositions of logical modeling. Kappler et al. [166] demonstrate how to extend simple Boolean networks by using ordinary differential equations to capture the concentration, while Boolean functions continue to determine the rates of the reactions. The probability of being in a state is sometimes a more reasonable measure to estimate, as in the case of Sachs et al. [250] who use Bayesian networks to model cell signaling pathways. Similarly, Shmulevich et al. [264] describe the use of probabilistic Boolean networks to model genetic regulatory networks and determine the long-term joint probabilistic behavior of a few selected genes. Platzer et al. [240] simulate the embryonic development of *C. elegans* by assuming Boolean states for the genes and synchronously updating at each time step based on an interaction matrix. Batt et al. [52] have applied model checking theory on biochemical systems modeled though qualitative simulation.

3.3.2 Differential Equations

If instead the concentrations are represented exactly in the real continuous domain, the differential equations of the dynamics directly follow from the KMA models. As a compromise between discrete and continuous representations, qualitative differential equations can be used with qualitative states corresponding to the different concentration ranges [52, 102]. Partial differential equations are necessary for spatially distributed models, e.g., reaction-diffusion equations.

3.3.3 Stochastic Master Equations

The biochemical system may be treated as stochastic, and the probability of system being in a certain state (rather than how the state of the system changes with time) may be estimated [129]. In a stochastic simulation, at each time step the next reaction and its time of occurrence are estimated, and the state of the system revised accordingly.

3.3.4 Hybrid Systems

The discrete states of the hybrid system naturally describe regimes of system behavior which are qualitatively different, in terms of which species and reactions predominate. The guards and resets of the discrete transitions allow description of the biochemical conditions under which the system-state changes. The use of hybrid automata for modeling biomolecular networks has been described by Alur et al. [6] and Mishra et al. [213]. Amonlirdviman et al. [19] demonstrated the utility of hybrid systems by modeling Drosophila planar cell polarity. Starting with the S-System formulation of Savageau and Voit[290], Antoniotti et al. [32] used an additional automaton to broaden the set of representable systems, subsequently using full-fledged hybrid automata [29]. Lincoln and Tiwari [195] detail hybrid automaton modeling of biochemical networks, while Hu et al. [153] describe stochastic hybrid system modeling of subtilin production in *Bacillus subtilis* (see *Table* 3.1 for a list of examples).

3.3.5 Other Methods

Chen and Hofestaedt [81] give a very good review of hybrid Petri nets and demonstrate its application to quantitatively model and simulate gene regulated metabolic networks. A slightly extended version - hybrid functional Petri nets have been used by Nagasaki et al. [226] to capture continuous and discrete events and transitions. Regev et al. [244] and Curti et al. [97] suggest the use of pi-calculus for biochemical modeling, while Phillips and Cardelli have proposed stochastic pi-calculus [237]. Another popular formalism is statecharts [139], where states and events that cause transitions between states are used to visually capture system dynamics that naturally allow concurrent, hierarchical and multi-scale modeling. Some applications of statechart-based modeling of biochemical networks are [113, 163, 120]. Weimar [293] uses the cellular automata approach to model enzyme catalyzed reactions. Bockmayr and colleagues have suggested the use of hybrid concurrent constraint programming for biochemical modeling [67, 117]. Faeder et al. [118] detail a general rule-based modeling approach where interaction are specified between biochemicals (the components), leading to the emergence of global behavior. Kohn and colleagues [175, 176] suggest molecular interaction maps (MIMs) as a standard for representing bioregulatory networks. While all methods hitherto discussed involve approximating at various levels of abstraction, *ab initio* modeling involves quantum mechanics based simulation at the atomic or sub-atomic level (large-scale projects include BlueGene[305], NAMD[238], GROMACS[270] and Desrad[262]).

Table 3.1: Biochemical Pathways Modeled as Hybrid Automata

Pathway	Models	Analyses
Delta & Notch [135]	neural net[205], non-linear ODEs [89], piecewise affine hybrid automaton [126]	optimization [205], simulation [89], symbolic methods [126, 222] Tolque in *Sec.* 9.3
Quorum Sensing [160]	hybrid automaton [6]	Charon [6], Simpathica [30]
Bacterial Chemotaxis [269, 62]	non-linear ODEs [269, 62], indepdent dynamics automaton[75]	simulation [269]
Purine metabolism [289]	S-System [289], XS-System [31]	simulation [289], Simpathica [31]
Sporulation in *B. subtilis*	qualitative differential equations [52], parametric hybrid system [195]	qualitative simulation & model checking [52], symbolic analysis in HybridSAL [195]
Cell polarity in *Drosophila*	partial differential equations[19], hybrid automata[19, 128]	simulation [19], symbolic analysis with MATLAB [128]

3.4 Examples of Biochemical Pathways

In this section, we try to convey the domain of application of the modeling and analyzing techniques relevant to this thesis by documenting some popular examples. For the convenience of the reader, the information about pathways and modeling efforts have been summarized in *Table* 3.1 and *Table* 3.2.

3.4.1 Delta-Notch

We first summarize the biochemical details of this complex pathway based on the review of Greenwald [135], and then present some simple mathematical models. Lin-12, Notch and glp-1 belong to the LIN-12/Notch family of receptor proteins, while apx-1, lag-2, Delta and Serrate belong to DSL family of ligand proteins. They are often referred to as just the "Delta-Notch" pathway, with the specific protein often

Table 3.2: Biochemical Pathways Modeled using other Formalisms

Pathway	Models	Analyses
Wnt Signaling [193]	non-linear ODEs [193]	control theoretic[193], simulation [193], Simpathica [215]
Repressilator [115]	non-linear ODEs [115], S-System [32]	simulation [115], Simpathica [32], Tolque in *Sec.* 9.4.2
4-Gene Artificial Circuit [137]	non-linear ODEs [137]	simulation [137]
Yeast Cell Cycle [282]	non-linear ODEs [282]	simulation [282], Simpathica [28], time-frequency analysis [45]
Mammalian Cell Cycle [175]	Molecular interaction map [175, 176], Concurrent Transition System[80, 79]	CTL–NuSMV–DMC [80, 79]
Ras / MAPK Pathway [206]	non-linear ODEs [206]	simulation [63], time-frequency analysis [45]
Immune System	Cellular Automaton[242], statecharts [164, 113]	IMMSIM [242], time-frequency analysis[45], simulation [164, 113], visual summarization [113]
RTK / EGF signaling	Pathway Logic [114], linear control system [297]	Term rewriting & model-checking [114], simulation [297]
G-Protein-coupled receptors	mechanistic Monte Carlo model [299]	Monte Carlo simulation [299]
FAK-ERK activation	Bayesian networks [250]	Bayesian analysis [250]

determined by the organism (*Caenorhabditis elegans*, *Drosophila melanogaster*, vertebrates, etc.). This receptor-ligand protein-protein interaction has been implicated in intercellular communication leading to cell fate determination. The mechanism of lateral inhibition (or lateral specification) whereby one cell expresses a ligand and tries to force adjoining cells to adopt a different cell fate has been used to explain pattern formation. The eventual fate of each cell seems to be determined by feedback mechanisms that amplify initial uneven distributions (either random or biased).

In addition to the several roles of the Delta-Notch pathway, its activity has also been documented in the AC/VU cell fate (Anchor Cell or Ventral Uterine precursor cell) decision during the *C. elegans* gonad development and the induction of the Drosophila wing margin. The complexity of these interactions is exemplified in the latter case, where two different ligands Delta (from ventral to dorsal) and Serrate (from dorsal to ventral) compete with the Notch receptors in the wing margin, under a bias imposed by a fourth protein Fringe which makes Notch prefer Delta over Serrate. Numb and Suppressor of Hairless (Su(H)) are also involved in related interactions in Drosophila, while lag-1 and emb-5 are downstream effectors that physically interact with LIN-12/Notch in *C. elegans*.

Another well-documented Notch mediated pathway in Drosophila is the specification of sensory organ precursors (SOPs), where equivalent proneural cells differentiate into one SOP and many epidermal cells. The process by which the ligand expression is repressed following receptor activation during SOP specification is thought to follow the following route:

$$Notch + Su(H) \xrightarrow{+} EnhancerofSplit - bHLH + Groucho \xrightarrow{-} Ac/Sc \xrightarrow{+} Delta,$$

where $\xrightarrow{+}$ and $\xrightarrow{-}$ denote positive and negative regulation, respectively. In addition, Notch is thought to undergo positive autoregulation (see [135] for other genes involved in the Notch pathway, their turnover, processing and trafficking).

One important aspect of the hypothesized pathway-models is that different aspects have been experimentally verified in different organisms, and the extrapolation of results across species is not always justified. Further, lateral specification is just one of the mechanisms in operation between the Delta and Notch families of proteins, that too, only in certain pathways in certain organisms (for instance see [235]).

This complete picture is disturbingly absent in the analysis literature, which is replete with attempts to explain the pattern formation aspect alone, assuming only the fundamental dynamics of the underlying biochemical interactions (and always assuming lateral specification / inhibition). Nevertheless, these approaches represent a promising starting point for developing more accurate models and performing more rigorous analyses.

Neural Network Modeling

Marnellos and Mjolsness [205] suggested modeling genes as nodes in recurrent neural nets, with connection weights corresponding to the strength of interaction. They modeled genetic regulation of the differentiation of neuroblasts and SOP cells from proneural clusters of equivalent cells. Their model did not assume even preliminary knowledge of the Delta-Notch pathway, but instead assumed a lateral inhibition mechanism.

A gene is assumed to "sum" the inputs from genes in the same cell or in neighboring cells at time t according to the equation:

$$u_a(t) = \Sigma_b T_{ab} v_b(t) + \Sigma_{i \in N} \Lambda^i \Sigma_b \hat{T}_{ab} \hat{V}_b^i(t),$$

where T is the matrix of gene interactions and the $v_b(t)$ are the gene product concentrations within the cell; $\hat{T}$ is the matrix of gene interactions with neighboring cells and the $\hat{v}_b^i(t)$ are the product concentrations in neighboring cell i; N is the set of neighboring cells and Λ^i is a factor depending on the overlap of the cell with neighboring cell i. Concentration of the product of gene a then changes according to:

$$\frac{dv_a}{dt} = R_a g(u_a(t) + h_a) - \lambda_a v_a(t),$$

where g is a sigmoid function, R_a is the rate of production of gene a's product, h_a is the threshold of activation of gene a and λ_a is the rate of decay of gene product.

The key to their approach is optimizing gene interaction strengths to fit gene expression patterns (obtained from literature) using simulated annealing. It is assumed that the optimization will return the model parameters that will lead to the corresponding biological system reaching the final state from the initial state. Their method was successfully used to analyze lateral signaling, cell delamination and the dynamics of proneural cluster resolution, and yielded interesting conclusions and predictions.

Differential Equation Modeling

Collier et al. [89] proposed a simple model to show that lateral inhibition (via feedback) was sufficient to explain much of the pattern formation. Rather than integrating details of the biochemical mechanisms, they assumed that:

> "*(1) Cells interact through Delta-Notch signaling only with cells with which they are in direct contact. (2) The rate of production of Notch activity is an increasing function of the level of Delta activity in neighboring cells. (3) The rate of production of Delta activity is a decreasing function of the level of activated Notch in the same cell. (4) Production of Notch and Delta activity is balanced by decay, described by simple exponential decay with fixed rate constants. (5) The level of activated Notch in a cell determines the cell's fate: low levels lead to adoption of the primary fate, high levels to adoption of the secondary fate.*"

The differential equations that would result from these assumptions have the form:

$$\frac{d(N_P/N_0)}{d\tau} = F(\bar{D}_P/D_0) - \mu N_P/N_0$$

$$\frac{d(D_P/D_0)}{d\tau} = G(N_P/N_0) - \rho D_P/D_0,$$

where N is the level of Notch activation, D is the level of Delta activity, τ is time, F is a continuous increasing function such as $\mu\frac{x^k}{a+x^k}$ and G is a continuous decreasing function such as $\rho\frac{1}{1+bx^h}$. μ and ρ are positive constants–the rate constants (inverse lifetimes) for decay of Notch and Delta activity respectively. $\bar{D}_P$ denotes the mean of the levels of Delta activity in the cells adjacent to cell P. Collier and colleagues [89] showed that the homogenous steady state is unstable while the heterogeneous steady state is stable under certain conditions.

Hybrid Automaton Modeling

Ghosh and Tomlin [126] proposed a simplified hybrid model to capture pattern formation via lateral inhibition. Each biological cell is modeled as a four state piecewise affine hybrid automaton, where the multiple states effectively allow the complex non-linear terms in the Collier model [89] to be simplified into affine terms. Their formal definition of the hybrid automaton is:

$$
\begin{aligned}
H_1 &= (Q_1, X_1, \Sigma_1, V_1, Init_1, f_1, Inv_1, R_1) \\
Q_1 &= q_1, q_2, q_3, q_4 \\
X_1 &= (v_D, v_N)^T \in \mathbb{R}^2 \\
\Sigma_1 &= \{u_D, u_N : u_D = -v_N, u_N = \Sigma_{i+1}^6 v_D^i\} \\
V_1 &= \phi \\
Init_1 &= Q_1 \times \{X_1 \in \mathbb{R}^2 : v_D, v_N > 0\} \\
flow_1(q,x) &= [-\lambda_D v_D; -\lambda_N v_N]^T if\, q = q_1 \\
&= [R_D - \lambda_D v_D; -\lambda_N v_N]^T if\, q = q_2 \\
&= [-\lambda_D v_D; R_N - \lambda_N v_N]^T if\, q = q_3 \\
&= [R_D - \lambda_D v_D; R_N - \lambda_N v_N]^T if\, q = q_4 \\
Inv_1 &= \{q_1, \{u_D < h_D, u_N < h_N\}\} \cup \\
& \quad \{q_2, \{u_D \geq h_D, u_N < h_N\}\} \cup \\
& \quad \{q_3, \{u_D < h_D, uN \geq h_N\}\} \cup \\
& \quad \{q_4, \{u_D \geq h_D, u_N \geq h_N\}\},
\end{aligned}
$$

where, v_D and v_N are the Delta and Notch protein concentrations, v_D^i is the Delta protein concentration in i-th neighboring cell; λ_D and λ_N are the Delta and Notch protein decay constants; R_D and R_N are the constant Delta and Notch protein production rates, respectively; h_D and h_N are the switching thresholds for Delta and Notch protein production, respectively. All four states are accessible from each other, with the guard of each transition equivalent to the destination state's invariant. Multiple cell models are obtained by composing many such single cell automata (see *Sec.* 9.3 for a detailed analysis of this model in Tolque).

3.4.2 Wnt Signaling

The Wnt family of proteins is similar to the Delta-Notch family in its pivotal role in signaling in developmental processes. With the major players in this pathway being Wnt, Frizzled, Dishevelled, GSK3b, APC, axin, β-catenin, and TCF, the Wnt family has also been implicated in oncogenesis. Lee et al. [193] developed a mathematical model for the system taking into account the kinetics of protein-protein interactions, protein synthesis / degradation, and phosphorylation / dephosphorylation. Here, Wnt proteins bind cell surface receptors which transduce the signal to β-catenin, which then activates transcription of Wnt target genes by complex formation with TCF. The model also involves the formation of unstable core complexes involved in β-catenin phosphorylation (involving GSK3b, APC and axin) and its subsequent destruction.

The independent variables of the model are: $X_2 \equiv$ Dsha, $X_3 \equiv$ (APC* / axin* / GSK3), $X_4 \equiv$ (APC / axin / GSK3), $X_9 \equiv$ (β-catenin* / APC* / axin* / GSK3), $X_{10} \equiv$ β-catenin*, $X_{11} \equiv$ β-catenin and $X_{12} \equiv$ Axin. The dependent variables of the model are $X_1 \equiv Dsh_i$, $X_5 \equiv$ GSK3, $X_6 \equiv$ (APC / axin), $X_7 \equiv$ APC, $X_8 \equiv$ (β-catenin / APC* / axin* / GSK3), $X_{13} \equiv$ TCF, $X_{14} \equiv$ (β-catenin / TCF) and X_{15} $\equiv$ (β-catenin / APC). The dynamics can be described thus:

$$
\begin{aligned}
\frac{dX_1}{dt} &= -v_1 + v_2 \\
\frac{dX_2}{dt} &= k_1 W(Dsh^0 - X_2) - k_2 X_2 \\
\frac{dX_9}{dt} &= \frac{k_9 X_3 X_{11}}{K_8} - k_{10} X_9 \\
\frac{dX_{10}}{dt} &= k_{10} X_9 - k_{11} X_{10} \\
\frac{dX_4}{dt} &= -(k_3 X_2 + k_4 + k_{-6}) X_4 \\
&\quad + k_5 X_3 + k_6 X_5 \frac{K_{17} X_{12} APC^0}{K_7 (K_{17} + X_{11})} \\
\frac{d(X_6 + X_{12})}{dt} &= v_3 - v_6 + v_{14} - v_{15} \\
\frac{d(X_3 + X_8)}{dt} &= v_4 - v_5 - v_9 + v_{10} \\
\frac{d(X_8 + X_{11} + X_{14} + X_{15})}{dt} &= -v_9 + v_{12} - v_{13} \\
\frac{dX_{11}}{dt}\left(1 + \frac{X_3}{K_8} + \frac{TCF^0 K_{16}}{(K_{16} + X_{11})^2}\right) + \frac{X_{11}}{K_8}\frac{dX_3}{dt} &= v_{12} - \left(\frac{k_9 X_3}{K_8} + k_{13}\right) X_{11}.
\end{aligned}
$$

Lee et al. [193] were able to predict (and subsequently experimentally verify) that axin and APC promote the formation of degradation complexes in different ways. Further, the role of axin degradation in amplifying and sharpening the Wnt signal and the dependence of axin degradation on APC were better clarified. They also performed control theoretic analysis to derive an explicit expression for tumor suppression and oncogenicity. Mishra et al. [215] describe the simulation and analysis of this system within their Simpathica / Valis framework.

3.4.3 Quorum Sensing

Vibrio fischeri is a single cell non-luminescent organism which often “chooses” to exist as a colony of luminescent symbionts in some fish and squid. This decision is based upon "quorum sensing", i.e., the knowledge that there is a high density of similar *V. fischeri* bacteria in its neighborhood [160]. The quorum sensing is achieved by the interaction of the inter-cellular communication mediating autoinducer A_i (which requires the LuxI protein), the luminescent protein luciferase, and the "lux operon" (which contains the genes for LuxI, luciferase and the transcriptional control factor LuxR). The binding of A_i to LuxR forms a complex that promotes transcription of lux genes by binding to the lux box.

Alur et al. [6] proposed a hybrid automaton model of quorum sensing, with three states corresponding to three levels of the luminescence / lux gene transcription: OFF ($A_i < A_i^-$), POS ($A_i^- \leq A_i \leq A_i^+$) and NEG ($A_i > A_i^+$, and hence there is negative growth). The model uses nine players: $x_1 \equiv$ mRNA transcribed from OL, $x_2 \equiv$ mRNA transcribed from OR, $x_3 \equiv$ protein LuxR, $x_4 \equiv$ protein LuxI, $x_5 \equiv$ protein LuxA, $x_6 \equiv$ protein LuxB, $x_7 \equiv$ autoinducer inside the bacterium A_i, $x_8 \equiv$ LuxR:A_i complex Co, and $x_9 \equiv$ autoinducer outside the bacterium $A_{i_{ex}}$. $\dot{s} = f^i(x)$ where $x = [x_1, x_2, \cdots, x_9]^T \in \mathbb{R}^9$, $f_i = [f_1^i, f_2^i, \cdots, f_9^i]$, and $i \in \{OFF, POS, NEG\}$. The description of the dynamics is as follows:

$$
\begin{aligned}
f_1^{OFF} &= \eta_1(\frac{1}{2}c - x_1) \\
f_1^{POS} &= \frac{\eta_1}{4}(3c + \frac{x_8^{\nu_{81}}}{\kappa_{81}^{\nu_{81}} + x_8^{\nu_{81}}} - 4x_1) \\
f_1^{NEG} &= -\eta_1 x_1 \\
f_2^{NEG} &= -\eta_2 x_2 \\
f_2^{POS} = f_2^{NEG} &= \eta_2(\frac{x_8^{\nu_{82}}}{\kappa_{82}^{\nu_{82}} + x_8^{\nu_{82}}} - x_2) \\
f_3^i &= \eta_3(x_1 - x_3) - r_{37,A_i} x_3 x_7 \\
f_4^i &= \eta_4(x_2 - x_4) - r_4 x_4 \\
f_5^i &= \eta_5(x_2 - x_5) \\
f_6^i &= \eta_6(x_2 - x_6) \\
f_7^i &= -\eta_7 x_7 + r_4 x_4 - r_{mem}(x_7 - x_9) - r_{37,R} x_3 x_7 \\
f_8^i &= -\eta_8 x_8 + r_{37,A_i} x_3 x_7 \\
f_9^i &= -\eta_7 x_9 + r_{mem}(x_7 - x_9) + u
\end{aligned}
$$

where, in the last seven equations f_j^i is independent of the mode. $\eta_i = T_0/H_i$ where T_0 is the characteristic time constant of the system and H_i is the half-life of molecule x_i. ν_{ij} is a cooperativity coefficient while κ_{ij} describes the potency of the regulation of the transcription of mRNA j by protein i. r denotes transformation and transfer rates. Alur and colleagues [6] then used Charon to investigate its dynamical properties and were able to verify that an increase in autoinducer concentration lead to bioluminescence. Antoniotti et al. [30] analyzed the dynamics using Simpathica, and suggested a procedure for breaking down each of the three states (of Alur's hybrid automaton) into a series of simpler states, by combining similar and adjacent time-course simulation data points using concepts like bisimulation and collapsing.

3.4.4 Repressilator and Other Artificial Circuits

Elowitz and Leibler [115] introduced the artificial genetic network – the "repressilator" by combining three repressors in a cyclic series. Their path-breaking experiment involved LacI from *E. coli*, which inhibits the transcription of tetR from the tetracycline-resistance transposon Tn10, whose protein product in turn inhibits cI from l-phage, whose protein product CI inhibits lacI expression.

By factoring in the dependence of transcription rate on repressor concentration, the translation rate, and the decay rates of the protein and messenger RNA, a reasonably accurate model of the system was obtained, where the system either converges to a steady state or oscillates. The dynamics of the three repressor-protein concentrations, p_i, and their corresponding mRNA concentrations, m_i was described thus [115]:

$$\begin{aligned}
\frac{dm_i}{dt} &= -m_i + \frac{\alpha}{(1+p_j^n)} + \alpha_0 \\
\frac{dp_i}{dt} &= -\beta(p_i - m_i), (i = lacI, tetR, cI; j = cI, lacI, tetR),
\end{aligned}$$

where the number of protein copies per cell produced from a given promoter type during continuous growth is α_0 in the presence of saturating amounts of repressor, and $\alpha + \alpha_0$ in its absence; β denotes the ratio of the protein decay rate to the mRNA decay rate; and n is a Hill coefficient.

Antoniotti et al. [32] present an S-System formulation of this system, and simulate and analyze it using Simpathica (also see *Sec.* 9.4.2 for an attempt at symbolic analysis of this system by Tolque). The repressilator system can be captured as an S-System as follows:

$$\begin{aligned}
\dot{X}_1 &= \alpha_1 X_3^{g_{13}} X_4^{g_{14}} - \beta_1 X_1^{h_{11}} \\
\dot{X}_2 &= \alpha_2 X_1^{g_{21}} X_5^{g_{25}} - \beta_2 X_2^{h_{22}} \\
\dot{X}_3 &= \alpha_2 X_2^{g_{32}} X_6^{g_{36}} - \beta_3 X_3^{h_{33}}.
\end{aligned}$$

Other Artificial Circuits

The repressilator idea [115] was rigorously extended and all combinations of four genes were created using plasmids capable of transfecting Escherichia coli, and studied by an expanded team including Guet, Elowitz, Hsing and Leibler [137]. Here, lac, λ cI and tet were each made to activate or repress one of the other two genes, while green fluorescence protein (gfp) was used to measure the outcome. IPTG and aTc were used to inactivate lac and tet genes. 125 circuits become possible using the five operons: two LAC-based: PL1, PL2; two λ CI-based: λP_-, λP_+; one TET-based: PT.

If x denotes a gene and X its corresponding protein, the equation for x's transcription is:

$$[\dot{x}] = -[x] + \alpha[\rho + f_x(\Theta, [Y], [u_y])], \, where$$
$$f_x(\Theta, [Y], [u_y]) = \frac{1 + \Theta[Y]^n + [u_y]^k}{1 + [Y]^n + [u_y]^k}.$$

In this equation, the transcription is activated or repressed by a protein Y and Y itself is modulated by a small molecule u_y. Note that, for small values of $[u_y]$, f_x shows a sharp transition from a value of 1 (when $[Y] = 0$) to a value of Θ (when $[Y] = \infty$), as Y increases. However, for large values of $[u_y]$, f_x remains at 1 (when $[u_y] = \infty$), thus inactivating the effect of Y . Similarly, the equation for X's (corresponding protein) translation is:

$$[\dot{X}] = -\beta([X] - [x]).$$

3.4.5 Yeast Cell Cycle

Tyson and Novak [282] formalized the hypothesized molecular mechanisms known to be involved in the cell cycle control into a well-defined model with ordinary differential equations. In their model, the key player is the protein kinase Cdk which requires a cyclin subunit to bind to for activation. Cyclin-bound Cdk initiates the cell cycle into the G1 phase from the quiescent G0 phase, and subsequently into the S and G2 phases. The cell controls Cdk activity primarily using the proteolytic activity of APC which destroys the Cdk-Cyclin binding and also the cyclin molecules. Cdh1/APC, CKI (Cyclin dependent kinase inhibitor) and Cdc20 keep the levels of active Cdk/CycB dimers low, while SK, CycB and Cdk counter the degradation of CycB and inactivation of CycB/Cdk complex. Some of these interactions are: CKI binds to CycB/Cdk to form inactive trimers; SK destroys CKI and inactivates Cdh1; CycB / Cdk inactivates Cdh1 by phosphorylation.

The empirical time-lag between the rise in CycB/Cdk activity and the activation of Cdc20 is accommodated in their model by assuming that a hypothetical intermediary enzyme IEP partakes in the process. The dynamics of the system are then described thus:

$$
\begin{aligned}
[\dot{CycB}] &= k_1 - (k_2' + k_2''[Cdh1])[CycB], \\
[\dot{CycB_T}] &= k_1 - (k_2' + k_2''[Cdh1] + k_2'''[cdc20_A])[CycB_T], \\
[\dot{CKI_T}] &= k_{11} - (k_{12}' + k_{12}''[SK] + k_{12}'''m[CycB])[CKI_T], \\
[\dot{Cdh1}] &= \frac{k_3' + k_3''[Cdc20_A])(1 - [Cdh1])}{J_3 + 1 - [Cdh1]} - \frac{(k_4'[SK] + k_4 m[CycB])[Cdh1]}{J_4 + [Cdh1]}, \\
[\dot{Cdc20_T}] &= k_5' + k_5'' \frac{([CycB]m/J_5)''}{1 + ([CycB]m/J_5)''} - k_6[Cdc20_T], \\
[\dot{Cdc20_A}] &= \frac{k_7[IEP]([Cdc20_T] - [Cdc20_A])}{J_7 + [Cdc20_T] - [Cdc20_A]} - \frac{k_8[Mad][Cdc20_A]}{J_8 + [Cdc20_A]} - k_6[Cdc20_A], \\
[\dot{IEP}] &= k_9 m[CycB](1 - [IEP]) - k_{10}[IEP], \\
[\dot{SK}] &= k_{13}[TF] - k_{14}[SK],
\end{aligned}
$$

where $[Cdc20_A]$ is the concentration of "active" Cdc20, and $[Cdc20_T]$ is the total concentration of both active and inactive forms; $[CycB]$ and $[Cdh1]$ are the average concentrations of cyclin B/Cdk dimers and active Cdh1/APC complexes, respectively; the ks are rate constants, the Js are Michaelis constants, m represents cell "mass"; $[CycB_T] = [CycB] + [Trimer]$; CKI_T is the total CKI.

Antoniotti et al. [28] described a set of tools within their Simpathica system for performing temporal analysis of the trajectories of bio-chemical pathways and to classify them into groups for further characterization. They demonstrated their technique on the Yeast cell cycle example. Barbano et al. [45] use time-frequency analysis of the time-series data of yeast cell cycle to test and validate interesting hypotheses.

3.4.6 Bacterial Chemotaxis

Escherichia coli has evolved an extremely effective strategy for responding to a chemical gradient in its environment, by detecting the concentration of ligands through a number of receptors, and then reacting to the input signal for driving its flagella motors to alter its path of motion[269, 62]. *E. coli* responds in one of two ways: either it "runs" – moves in a straight line by moving its flagella counterclockwise (CCW), or it "tumbles" – randomly changes its heading by moving its flagella clockwise (CW). An *E. coli* cell performs a biased random walk by transiently decreasing its tumbling frequency to move towards a region with greater ligand concentration. A second feature of this control is its sensitivity to concentration gradients and its observed dynamic range: rather than responding to absolute concentrations, the *E. coli* adapts quickly as it compares its environment during the immediate past to what existed a little earlier. Further, it does so over a wide range of input concentrations.

The response is mediated through the molecular concentration of CheY in a phosphorylated form (Y_P), which in turn is determined by the bound ligands at the receptors that appear in several forms (LT variables). The ratio of $y = Y_P/Y_0$ (phosphorylated concentration of CheY to its concentration in the unphosphorylated form) determines a bias with an associated probability that flagella will exert a CW rotation. The more detailed pathway involves other molecules: CheB (either with phosphorylation or without, B_p and B_0), CheZ (Z), bound receptors (LT) and unbound receptors (T), while their continuous evolution is determined by a set of differential algebraic equations derived through kinetic mass action formulation:

$$\begin{aligned}
\frac{dT_2}{dt} &= -Lk_5T_2 + k_{-5}LT_2 - k_8T_2 + B_pk_{-1}T_3 + k_yT_{2p}(Y_0 - Y_p) \\
&\quad + k_bT_{2p}(B_0 - B_p) - V_{max}\frac{T_2}{K_R + T_2} \\
\frac{dY_p}{dt} &= k_yP(Y_0 - Y_p) - k_{-y}ZY_p \\
\frac{dB_p}{dt} &= k_bP(B_0 - B_p) - k_{-b}B_p,
\end{aligned}$$

where $P \equiv T_{2p} + LT_{2p} + T_{3p} + LT_{3p} + T_{4p} + LT_{4p}$ is the total amount of phosphorylated receptor complex. In addition, the total amounts of T, Y, B, and R are conserved.

Casagrande et al. [75] model this system as an Indepdent Dynamics Automaton, where the two possible directions of rotation of the flagella lead to two different states in the hybrid automaton. There, the angular velocity w of the flagella takes discrete values $+1$ for CW and -1 for CCW.

3.4.7 Other Examples

Ras / MAPK Pathway The MAPK enzymatic cascade is activated by EGF through two interconnected pathways: the $PLC_\gamma - PKC$ and the $Ras - Raf - MAPK$ pathways [206]. The mutual activation leads to a feedback loop between PKC and MAPK which results in the multimodal dynamics of the system: depending on the duration and concentration of the EGF stimulation, the system follows different evolution dynamics after the EGF stimulus is withdrawn. The underlying dynamics may be expressed using equations of the form:

$$\dot{X}_i = \Sigma_{1 \leq i,j,k \leq n} a_{ijk} [X_i]^{s_i} [X_j]^{s_j} [X_k]^{s_k} \ 1 \leq i \leq n,$$

whose coefficients can be derived from in vitro data. The simulations of this system were carried out by Bhalla and Iyengar in their seminal publication [63]. Barbano et al. [45] also applied their time-frequency analysis approach to this EGF-modulated RAS pathway as well.

Immune System Modeling In the immune system, B-cells and nonspecific antigen presenting cells (APCs) internalize the antigens that have entered the system, and present an MHC peptide complex that T-cells can recognize, bind to, get stimulated and divide. Stimulated B cells produce plasma cells (which produce the antibodies that remove the antigen) and memory B cells. Rather than using differential equations, in IMMSIM [242], a modified cellular automaton is used to simulate the effects of cell-cell and cell-molecule interactions in the lymphoid system. Based on the number of components participating in the immune response, the simulator applies a specific set of rules to determine the next state. Barbano et al. [45] use time-frequency analysis to study the IMMSIM model and examine its adaptive dynamics in response to primary and secondary infections.

Kam et al. [164] model T-Cell activation using Statecharts [139], an approach which was made more rigorous by Efroni et al. [113]. Large data-sets about cell migration, cell differentiation, histology, electron microscopy, biochemistry and molecular biology were integrated into a two-tier model of thymic maturation using Statecharts. In addition to executing illuminatory simulations and analyzing them, a lucid representation of the generated information is extracted by their tool.

Purine metabolism In purine biosynthesis [289], a linear cascade of reactions converts *5-phosphoribosyl-α-1-pyrophosphate* (PRPP) into *inosine monophosphate* (IMP). IMP is transformed into AMP and GMP. Guanosine, adenosine and their derivatives are recycled into *hypoxanthine* (HX) and *xanthine* (XA). XA is finally oxidized into *uric acid* (UA). Further, *adenine phosphoribosyltransferase* (APRT) and *hypoxanthine-guanine phosphoribosyltransferase* (HGPRT) combine with PRPP to form ribonucleotides. Antoniotti et al. [31] describe an XS-System based analysis of this pathway using the Simpathica tool.

More Instances Lahav et al. [187] used fusion proteins and time-lapse fluorescence microscopy in a novel way to study the p53-Mdm2 feedback loop, and developed a model suggesting a "digital clock" like behavior of the feedback loop in releasing well-timed quanta of p53 in response to cell damage. Monk [219] constructed mathematical models that helped conclude that the oscillatory expression of Hes1, p53, and NF-κB was most likely driven by transcriptional delays, with the time-period being determined by the delay and the protein and mRNA half-lives. In addition to the examples cited previously, we find: the work of Davidson et al. [99] on developmental regulation, cellular signal transduction modeling [190, 42], Wiley et al. [297]'s experiments with the Epidermal Growth Factor, Woolf and Linderman [299]'s study of G-protein Coupled Receptors, general analysis of metabolism [132, 220, 258, 5], and modeling general protein-protein interactions [283].

3.5 Analyzing Biochemical Models

One of the central problems in Systems Biology is: *Given the mathematical representation of the biochemical system and a description of its dynamics, how does one prove or disprove statements about its temporal behavior?* Clearly, the ability to query the model of a biochemical system can hasten the discovery of emergent properties, aid the hypothesizing of interesting inferences and intelligent predictions, help revise the model, and possibly, suggest new wet-lab experiments that could help refine the model. Several modeling tools, some with rigorous analytical capabilities, have been developed, including Systems Biology Workbench [154], Virtual Cell [268], E-Cell [275], Genomic Object Net [226], Simpathica[33] and BioMiner [267] (see *Sec.* 1.1 for references to more tools, organizations and standards; also see *Tables* 3.1 and 3.2). The lay-out of a typical biomodeler is depicted in *Figure* 3.1.

The temporal properties of a network of interacting biochemicals are typically captured by relating two neighboring system-states at time instants t and $t + \delta$, and the biochemical interactions (synthesis, degradation, multimerization, etc.) that occur in that short time interval δ. Nonetheless, a direct model of transitions and flows,

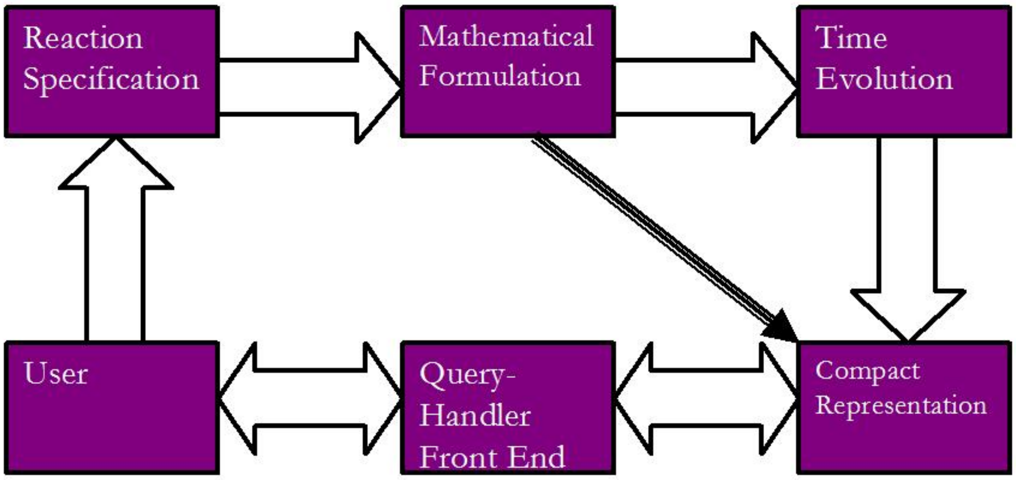

Figure 3.1: Layout of a typical Biomodeler

given through their symbolic description, can be computationally manipulated (either numerically or symbolically) to derive logical conclusions about global temporal properties, that may not have been obvious in the instantaneous description. The exact structure of this approach depends on the complexity of the three underlying frameworks: description of the dynamical system, the expressivity of the temporal logic and the basic operations of the models of computation. We now describe the different approaches to analyzing biochemical models.

Querying the Time-Trace

The simplest type of analysis involves numerically simulating the system for a finite time-interval. In this conventional "numerical" approach, starting with an initial system-state, successive states are chased by an integration scheme (e.g., Runge-Kutta). Conclusions about the behavior of the network are then made by tracing the trajectories over a suitable time-frame and verifying temporal properties (e.g., the Simpathica tool [33]). The numerical simulation-based approach is summarized in *Figure* 3.2.

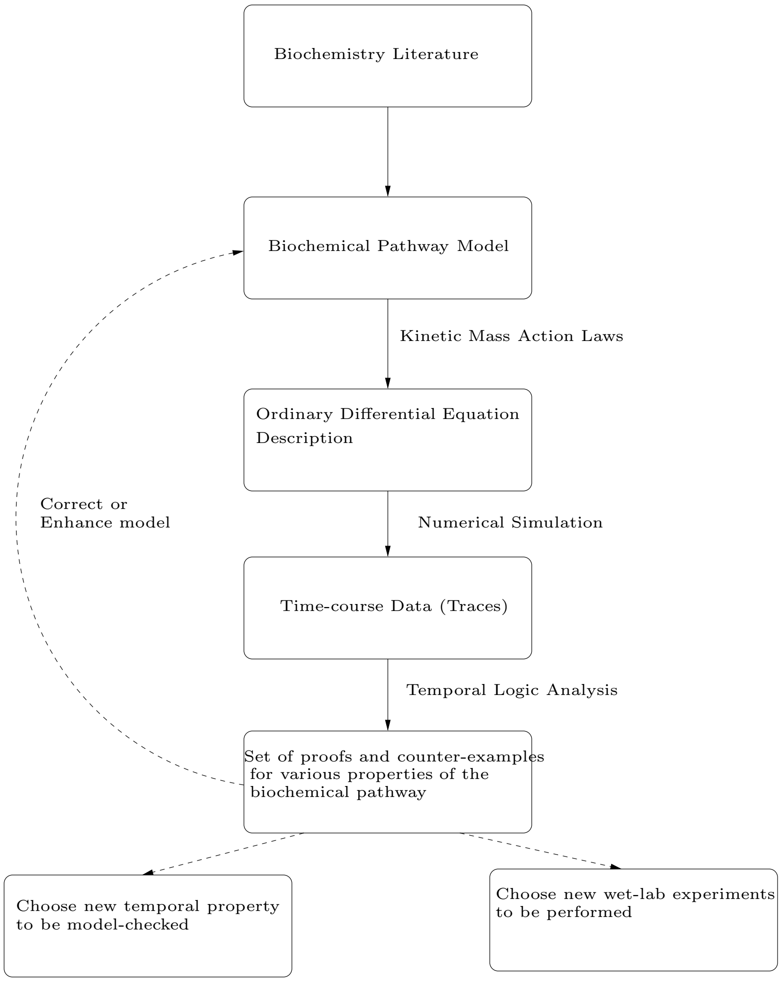

Figure 3.2: Typical steps in analyzing a biochemical model using numerical simulation

Modeling and Simplifying the Time-Trace

Alternatively, one could first remove the redundancy in the time-log of the variables, so queries can be answered faster. Depending on the assumptions that one is allowed to make, one could resort to the correct Machine Learning technique to extract a simpler model with as much mathematical sophistication as possible. It is to be noted that to answer queries about a different starting state or parameter value, a new simulation will most likely need to performed.

Without Calculating the Time-Trace

A more difficult approach is to perform the computations symbolically rather than numerically. This would require the ability to arrive at an easy-to-query algebraic representation from the given mathematical representation, without going through the process of evaluating the time-trace at several initial states and parameter values. This "symbolic" alternative to the numerical procedure would use algebraic methods to characterize the transition of the system with time. In effect, the process of numerically integrating the differential equations and extracting a simpler query-able representation will be replaced by an algebraic procedure that can answer queries about the symbolic state of the system.

3.6 Discussion

In summary, Systems Biology is an intriguing and challenging research area, with a plethora of open problems awaiting rigorous mathematical treatment. Systems Biology attempts to integrate several computational, mathematical and modeling approaches to capture the temporal behavior of biochemical dynamical systems. Once the system and its evolution-rules have been specified through a formal description, formal methods like model checking and theorem proving can be used to check if certain states are ever reached, if some initial condition could cause a certain effect, etc. Hybrid automata represent the broadest class of dynamical systems over which many of these formal temporal reasoning algorithms have been developed. Hybrid automata are likely to emerge as the universal formalism for modeling networks of biochemical interactions, as they present an intuitive way of capturing the KMA-based biochemical dynamics. In addition, the discrete states and transitions may be used to capture mode changes and external events, and also serve as a convenient means of approximation. Further, a hybrid automaton captures a network of biochemical interactions intuitively and accurately, and requires minimal book-keeping of the constructs of the formalism.

Compared to the numerical analysis of biochemical systems modeled as hybrid automata, the symbolic approach can answer deeper questions. This is because the query and its answer can be algebraic expressions describing the different variable values and relationships under which the property of interest is guaranteed to hold. Clearly, more general queries can be answered. Only symbolic algebraic analyses can hope to yield deeper insights into the innate yet non-apparent principles governing biology. On the computational side, simulations do not have to be redone when the parameters or the initial states in the query numerically change, and only the simulations necessary to answer the query are performed. The obvious disadvantage however is that purely symbolic algebraic methods are rarely universally applicable, and typically have very high computational complexity. In this thesis, we describe different ideas and approaches that strive towards making such an analysis of hybrid automata possible.

Chapter 4

Refining the Undecidability Frontier

Having understood the underlying Biochemistry and the role of hybrid automata in modeling such processes, we now strive to find a hybrid automaton subclass suitable for modeling them. Ambitiously, we first peruse decidable classes for which very efficient model checking approaches have been developed. We quickly realize their insufficiency, and try to understand when and how decidable subclasses become undecidable. This leads us to the Hierarchical Piecewise Constant Derivative (HPCD) class that lies in the "open" region between decidability and undecidability. We provide new constructions that dramatically improve our understanding of how "close" HPCDs are to decidable and undecidable subclasses. Though no Systems Biology applications are conceivable, this chapter is very effective in bringing out the power and expressivity of hybrid automata, and is a significant contribution from the theoretical computer science perspective.

4.1 Introduction

Hybrid automaton (HA) is a formalism for capturing a broad range of dynamical systems with continuous and discrete properties. However, the more expressive freedom a formalism is endowed, the more difficult it becomes to analyze the resulting system. Thus, a crucial factor in choosing the appropriate modeling formalism for biochemical systems is the ability to guarantee analyzability of the formalism. In the case of hybrid automata, the question boils down to: "What are the restrictions on the expressions that can appear in the hybrid system description, in order for analysis to be always possible?". The set of all hybrid automata that satisfy certain constraints is referred to as the *class* defined by those constraints. Thus, the question can be rephrased as: "What is the *subclass* of hybrid automata over which *analyzability* can be *guaranteed*?".

To address this problem, we first need to identify interesting biochemical questions about the hybrid automaton model that we wish to eventually answer. Literature helps us conclude that the question "Is a certain final state reachable from a certain initial state?" is a very significant one, as it can be rephrased in several interesting ways. Formally, this question corresponds to the well-studied property of *reachability* – the problem of deciding whether a certain final state is reachable from a given initial state.

Having clarified "analyzability", the second issue is being able to guarantee it. In other words, we would like to claim that we can *always* answer *any* reachability question over *any* hybrid automaton that satisfies certain constraints (i.e., belonging to a certain class). The formal term for this property is "decidability". Thus, our aim is to identify the hybrid automaton subclass for which the reachability problem is decidable.

A problem can be proven to be decidable over a class, if an algorithm (procedure) can be detailed for solving every instance of the problem in that class. Proving that a class is undecidable requires that we identify at least one problem in that class for which no solution can exist. This unwieldy notion of there being a problem which is provably unsolvable was addressed by stalwarts like Gödel, Church and Turing. The result most relevant to our attempt to identify a decidable hybrid automaton subclass for biochemical systems is the "halting problem" of Turing. Turing defined a computational model called the "Turing Machine" (TM), and showed that the problem of deciding whether a given Turing Machine will halt on a given input is undecidable over the class of all Turing Machines [281]. A very important corollary of this stellar result is that any class of automata that can simulate a general Turing Machine is also undecidable.

In practice however, it often becomes simpler to prove that a formalism called the two-counter Minsky Machine (MM) can be simulated. Since the two-counter Minsky Machine has been shown to be able to simulate a TM [211], reachability is undecidable for an MM, and any dynamical system that can simulate an MM as well. Thus, proving that a hybrid automaton subclass is decidable involves providing the actual algorithm for solving the general reachability problem for that subclass, while proving undecidability requires providing an algorithm for simulating a general Minsky Machine in a hybrid automaton, without violating any of the constraints of that particular HA subclass.

Hybrid automata, which can have arbitrary discrete transitions and continuous flows, correspond to a class of immense computational power. HA very easily become undecidable for the reachability query, with only extremely stringent restrictions leading to decidability. Timed automata [12], multirate automata [9], initialized rectangular automata [241, 144], controllable linear systems [273], some families of linear vector fields [186] and O-minimal HA [184] have been shown to be decidable for the reachability query.

The fundamental question continues to be: "What is the simplest class of dynamical systems for which reachability is undecidable?". The conventional answers to this question have involved proving that a certain decidable class becomes undecidable, when given some additional computational power. For instance, 2-dimensional Piecewise Constant Derivative (PCD) systems [201] and Simple Planar Differential Inclusions (SPDIs) [40] are decidable, while 3-dimensional PCDs are undecidable [38]. This chapter focuses on the 2-dim Hierarchical PCD (HPCD) class introduced by Asarin and Schneider [39]. This intermediate class, between decidable 2-dim PCDs and undecidable 3-dim PCDs, is not known to be provably decidable or undecidable. Asarin and Schneider proved that 2-dim HPCDs are *equivalent* to 1-dim Piecewise Affine Maps (PAM). Since the reachability problem for 1-dim PAMs is an open question [177], 2-dim HPCD-reachability is also open. They went a step further, and proved that the HPCD class, when endowed with a little additional computational power, becomes undecidable. Thus, the HPCD [1] class (and equivalently the PAM class) is clearly on the boundary between decidable and undecidable subclasses of HA.

This chapter presents new developments in the analysis of the HPCD class, a sequel to Asarin and Schneider's work [39]. The three questions we seek to answer are: (1) Is there a class, simpler than the HPCD class, which can be shown to be equivalent to PAMs? In other words, is there a simpler "open" class? (2) Are there

[1]Henceforth, "HPCD" refers to 2-dim HPCD, "PAM" to 1-dim PAM, and "decidability" to decidability of reachability, unless explicitly stated otherwise.

alternate extensions of the HPCD class which become undecidable? (3) Can we develop an approximate reachability algorithm, and thus obtain some insight about decidable HPCD subclasses?

We begin this analysis of the proximity to decidability and undecidability in *Section* 4.2, with the definitions of the various subclasses of HA we will encounter in this chapter (more subclasses are discussed in *section* 5.1.1). In *Section* 4.3.1, we present our main result: 2-dim PCDs with translational resets can simulate a PAM. We then construct several very interesting subclasses of HPCDs, which also simulate PAMs. Since PAMs have been shown to be equivalent to HPCDs [39], it proves that surprisingly, these subclasses are just as powerful as the HPCD class itself. This reveals the redundancy in the expressive power of the HPCD, and shows how even closer HPCDs are to decidable systems. In *Section* 4.4, we present some undecidable extensions of HPCDs, different from Asarin and Schneider's constructions. These constructions reveal new dimensions of the fineness of the line separating HPCDs from undecidability. We present a simple algorithm for over-approximating reachability in PAMs in *Section* 4.5.3, and show how decidable subclasses can be identified. We summarize our contributions in *Section* 4.6 and discuss several open research questions. For brevity, most proofs are provided in *Section A* of the *Appendix*.

4.2 Background: Hybrid Automata and Subclasses

We quickly review some terms frequently used to describe different restrictions. Let a, b, c, d stand for numerical constants, and p, q stand for the hybrid system variables. A *rectangular guard* refers to an expression of the form $a < p < b$, while a *non-rectangular* or *comparative guard* is of the form $ap + bq + c < 0$. A *rectangular invariant* is of the form $a < p < b \bigwedge c < q < d$, i.e., the state represents a rectangular region in the $p - q$ plane. State invariants are said to be *non-overlapping* if the regions they represent in their variable-space do not intersect. A *constant reset* refers to $p' = c$, a *translational reset* refers to $p' = p + c$ and an *affine reset* to $p' = ap + b$. An "initialized" automaton is one where all variables, whose flow changes after a discrete state transition, are reset to a constant. An automaton is "timed" if all flow-derivatives are 1.

4.2.1 Timed and Rectangular Automata

Timed Automata are the most restricted subclass of hybrid systems, and also the most well studied. Here, all variables are assumed to be clocks, i.e., all flow derivatives are +1. This apparently over-constrained system has actually found to be extremely useful in analyzing VLSI circuits, where transistors naturally possess only the two Boolean states "on" and "off". The timed constraint has been relaxed by letting variables have dynamics described by a flow inclusion (range). In rectangular hybrid automata, a crucial constraint is imposed on the reset relations: if the dynamics of a variable change as a result of a transition, then the variable has to be reset to a constant value.

Definition 4.2.1 2-Dimensional Timed Automaton [144] *A 2-dim timed automaton is a hybrid automaton in two continuous variables where:*

1. *All flow-derivatives are 1, i.e., both variables are clocks.*

2. *All resets are constant.*

3. *The guards are interval checks and do not involve comparison of the 2 clocks.*

4. *Invariants are constant intervals for each variable.* □

If clock comparisons are allowed for a larger number of clocks, the system becomes undecidable.

Definition 4.2.2 2-Dimensional Initialized Rectangular Automaton [144] *A 2-dim initialized rectangular automaton is a hybrid automaton in two continuous variables where:*

1. *All flow-derivatives are bounded by constant intervals.*

2. *Whenever a variable's flow changes, it is reset to a constant (initialization).*

3. *The guards are interval checks and do not involve comparison of the 2 variables (rectangularity).*

4. *Invariants are constant intervals for each variable.* □

In other words, the real plane is divided into (possibly overlapping) rectangles corresponding to the different states and the two variables evolve within each state with derivatives in a constant range. When the bounds on a variable's flow-derivative changes, it jumps to a different rectangular zone where it restarts from a fixed point (reset to a constant).

4.2.2 PCDs and HPCDs

Among the several formalizations that simplify the reachability problem by curbing the computational power of the HA, we dwell on the PCD construct.

Definition 4.2.3 2-dim PCD [201] *A 2-dimensional Piecewise Constant Derivative system is an HA in two continuous variables, where:*

1. *All flow derivatives are constants.*

2. *The guards are rectangular, i.e.,* $p \in I$, *where* I *is a numerical interval.*

3. *No variable can be reset during transitions, i.e.,* $p' = p \wedge q' = q$.

4. *The discrete states (invariants) correspond to* non-overlapping *rectangles in the real plane with non-empty interiors.* □

The trajectories of a 2-dim PCD are restricted to be broken straight lines, with slopes changing only when a different polygonal region (new discrete state) is entered. The PCD restriction is motivated by the fundamental property of planar systems. It states that the evolution of a (2-dim) point in a plane with fixed slopes (flow) at each point, can *only* trace out a contracting or expanding spiral (if not a simple finite cycle). Maler and Pnueli [201] used the property of planar systems to prove that reachability is decidable for 2-dim PCDs. 3-dim PCDs are the natural extension of 2-dim PCDs with a third dynamic variable (dimension).

Definition 4.2.4 3-Dimensional Piecewise Constant Derivative System [38] *A 3-dim PCD is an HA in three continuous variables with non-overlapping state-invariants of the form* $x \in I_x \wedge y \in T_y \wedge z \in I_z$, *where* x, y *and* z *are the 3 dimensions and* I_x, I_y *and* I_z *are numerical intervals. Further, the flows are constant and no resets are allowed.* □

Asarin, Maler and Pnueli [38] proved that 3-dim PCDs are undecidable.

Subsequently, Asarin and Schneider set out to discover an "open" class in between 2-dim and 3-dim PCDs. They proceeded by studying HA that could simulate a known open problem - the 1-dim PAM. To understand their equivalence result, we first introduce PAMs, where computation is modeled as iterative function evaluation.

Definition 4.2.5 Piecewise Affine Map [177] *A PAM [177] is of the form $f(x) = a_i x + b_i$, $x \in I_i$, $i = 1, 2, \cdots, n$, where all a_i, b_i and the ends of the non-overlapping intervals I_i are rational. f is closed, i.e., $\forall x, i\ (x \in I_i) \Rightarrow (\exists j,\ f(x) \in I_j)$. Further, the intervals are in ascending order.* $\square$

In other words, there are n non-overlapping partitions of the real line (which may not cover it entirely). The current value of the variable x decides which interval I_i it falls in, and hence its next value $f(x)$ is uniquely defined. The reachability problem is also defined in the natural way: "Is the point x_f reachable from the point x_0 by repeated application of the piece-wise affine maps?". Note that unlike HA, there is no non-determinism or choice – the starting point defines a unique trajectory.

Recall that a class A simulates a class B if every computational trajectory of B has a unique counterpart in A. Two classes are equivalent if they simulate each other: thus if the reachability problem is (un)decidable for one class, so it is for the other. Asarin and Schneider characterized the PCD extensions necessary to simulate a PAM, keeping in mind that the resulting HA subclass in turn needed to be expressible as a PAM. The result was the HPCD class which augmented a PCD, by allowing comparative guards and affine resets in overlapping regions of the plane.

Definition 4.2.6 2-Dimensional Hierarchical Piecewise Constant Derivative System [39] *A 2-dim HPCD [39] is an HA in 2 continuous variables where:*

1. *All flow-derivatives are constants.*

2. *The guards are of the form $(ax + by + c = 0 \wedge x \in I \wedge y \in J)$ where I and J are intervals and a, b, c and the extremities of I and J are rational-valued.*

3. *The reset functions are affine functions: $x' = ax + b$.*

4. *The state invariant, which could overlap with other state invariants, is the negation of the union of the guards.* □

The term hierarchical was used originally, to indicate that an HPCD could also be thought of as a PCD with overlapping state invariants, where each state was actually a PCD.

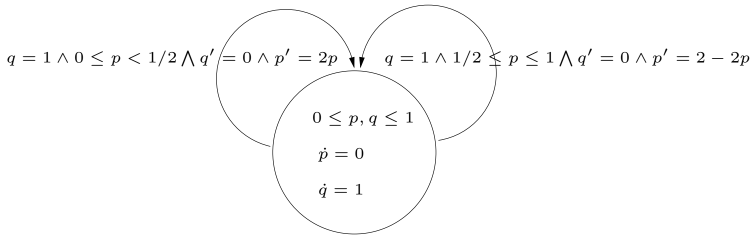

Figure 4.1: One-State Tent Map HPCD

Example 4.2.1 *Consider the PAM describing the Tent Map [277]:*

$$f(x) = 2x + 0, \; x \in [0, 1/2)(\equiv I_1)$$
$$f(x) = -2x + 2, \; x \in [1/2, 1](\equiv I_2)$$

The HPCD simulating this PAM is shown in Figure *4.1.* □

Theorem 4.2.1 ***Summary of HPCD Results*** *[39] The main results concerning HPCD decidability are:*

1. *The HPCD class is equivalent to the PAM class.*

2. *The restricted class $HPCD_{iso}$, with translational instead of affine resets, is also equivalent to the PAM class.*

3. *The extended classes $HPCD_{1c}$ (with an additional counter), $HPCD_{\infty}$ (with infinite partitions) and $HPCD_x$ (origin-dependent rates) are undecidable, as they can simulate a TM.* □

4.3 Open HPCD Subclasses

The HPCD class, unlike the PCD class, does not require that the different discrete states correspond to non-overlapping rectangles of $\mathbb{R} \times \mathbb{R}$. Further, it extends the PCD class by allowing comparative guards and affine resets. Asarin and Schneider's results thus have two implications: (1) a PAM can capture an HPCD; (2) an HPCD can capture a PAM. Their first result is clearly the more significant one, also involving a non-trivial construction. It demonstrates that a 2-dim HPCD, which seems dramatically more complex than a 1-dim PAM, actually has no additional computational power. Their second result seems simple in comparison, as HPCDs seem to have more expressivity than necessary, to capture a PAM. This becomes evident in their proof, where a PAM can be trivially captured by a PCD with just 1 state, with all computations done only using affine resets along self-loops (see for example, the Tent Map HPCD in *Figure* 4.1). In particular,

$$\begin{aligned} f(x) &= a_i x + b_i,\ x \in I_i \ \textit{is captured as}: \\ \textit{flow}: \dot{p} &= 0,\ \textit{guard}: p \in I_i,\ \textit{reset}: p' = a_i p + b_i \\ \textit{flow}: \dot{q} &= 1,\ \textit{guard}: q = 1,\ \textit{reset}: q' = 0. \end{aligned}$$

Note that comparative guards and overlapping state invariants are not used at all. Further, the second variable q is used as a dummy stop-watch, only serving to make the system non-zeno. Schneider has proved [259] that these affine resets can be made translational ($HPCD_{iso}$). However, that construction uses all the other enhancements. To summarize, PCDs with just *affine* resets can simulate a PAM. However, multiple states with overlapping invariants and comparative guards seem necessary, when only *translational* resets are allowed. The question that we now ask is: "What is the minimal set of enhancements required by 2-dim PCDs to simulate 1-dim PAMs?".

4.3.1 PCD with Translational Resets

In this section, we first prove our main result: a 1-dim PAM can be simulated using a 2-dim PCD, with just translational resets of the form $x' = x + c_i$. Comparative guards and affine resets can be done away with by making the two PCD variables (p and q) take turns simulating the PAM variable (x), while non-overlapping state invariants become sufficient because the PCD variables are guaranteed to lie in a bounded region. We then show how 1-dim PAMs can be simulated by other simple subclasses of HPCDs, each revealing proximity to a different decidable HA subclass. The following lemma simplify the proof:

Lemma 4.3.1 *A 1-dim PAM is bounded.* $\square$

Lemma 4.3.2 *Every 1-dim PAM is equivalent to a 1-dim "positive" PAM where all intervals are positive.* $\square$

Now we are ready to prove our main result:

Theorem 4.3.1 *A 1-dim PAM can be simulated by a 2-dim PCD with translational resets.*

Proof 4.3.1 *Consider an equivalent 1-dim positive PAM* $f(x) = a_i x + b_i$, $x \in I_i (\equiv [l_i, r_i))$, $i = 1, 2, \cdots, n$. *Let* L *be a number such that* $L > r_n \wedge \forall i,\ L > b_i$. *Corresponding to the* i*-th function of the PAM, we will have two states* P_i *and* Q_i. *In* P_i, p *flows from* $p_0 = b_i$ *to* $x_{n+1} (\equiv b_i + a_i x_n)$ *at the rate* $\dot{p} = a_i$. q *drops from* $q_0 = x_n$ *to* 0 *at the rate* $\dot{q} = -1$. *The guard* $q = 0$ *thus ensures that the system spends* $t = q_0$ *time in this state. This allows the affine term* $a_i x_n$ *to be computed, without using comparative guards or affine resets. In the "Q" states, the roles of* p *and* q *are* *reversed, i.e., q uses p's value to grow to the next iterate, while p just drops to 0, effectively keeping track of time.*

From P_i, there are transitions to each possible state Q_j. p retains the value it just computed, while q is reset to the constant portion (b_j) of the next iterate of x. In Q_j, q will accumulate the rest of its target value ($a_j \times x$) by flowing for time x (stored in p) at the rate a_j. Similarly, from Q_i, there are transitions to each possible state P_j, while there are no transitions within P-states or within Q-states.

The above expressions are adjusted, now assuming that each state is associated with a different large constant "base". In a state, all numbers are represented with respect to this base. Thus, x becomes $L_{S_i} + x$ in state S_i, where L_{S_i} is the base. Even if x increases or decreases to its maximum / minimum possible value, p and q will not cross over to an adjoining state. This is because the different base constants are themselves very far apart. This base-adjustment creates the translational resets, when the current iterate needs to be remembered and passed on to a new state. It is to be noted that just constant resets suffice if the state invariants are allowed to overlap.

We now construct the PCD with translational resets and $2n$ states:

- *Corresponding to the i-th function of the PAM, we have two states P_i and Q_i associated with the constants $L_{P_i} = 4iL - 3L$ and $L_{Q_i} = 4iL - L$.*

- *In P_i, p grows at rate $\dot{p} = a_i$ from $L_{P_i} + p_0 (= b_i)$ to $a_i q_0 (= x_n) + b_i + L_{P_i}$, while q drops from $q_0 + L_{P_i}$ to L_{P_i} at the rate $\dot{q} = -1$. q_0 denotes the unscaled previous iterate x_n, using which x_{n+1} is being computed by spending exactly $t = q_0$ time in this state.*

- *Q_i behaves exactly as above with p and q swapped, i.e., this corresponds to the case where q grows to the next iterate, while p just drops to L_{Q_i}.*

- *In P_i and Q_i, the values of p and q are both bounded by $\{(L_{P_i/Q_i} - L, L_{P_i/Q_i} + L)\}$, which is equal to $\{(4iL - 4L, 4iL - 2L)\}$ in P_i and $\{(4iL - 2L, 4iL)\}$ in Q_i. Clearly, none of rectangular regions can overlap.*

- *From P_i, there are transitions to each possible state Q_j with guard $q = L_{P_i} \wedge p \in I_j$, i.e., "$p$ has reached the next iterate of x"* and *"p is in the interval corresponding to the j-th PAM function". The reset (note: constant or translational) is $p' = p - L_{P_i} + L_{Q_j} \wedge q' = L_{Q_j} + b_j$, i.e., "$p$, which holds the current value of x, is translated to the range of the destination state (to prevent overlap)"* and *"q is reset to the constant portion (b_j) of the next iterate of x". The portion proportional to x_n ($a_j \times x_n$) will be gained by flowing for time x_n (stored in p) with slope a_j.*

- *Similarly, from Q_i, there are transitions to each possible state P_j. There are no transitions within P-states or within Q-states.*

This PCD with translational resets simulates the PAM, as p and q take turns simulating x. It can be seen that x_f is reachable from x_0: (i) if $(p = x_f + L_{P_i}, q = L_{P_i})$ and $(p = x_f + L_{Q_j}, q = L_{Q_j} + b_j)$ are reachable; or (ii) if $(p = L_{Q_i}, q = x_f + L_{Q_i})$ and $(p = L_{P_j}, q = x_f + L_{P_j})$ are reachable. This needs to hold for some i and j, such that $x_f \in I_j$ and one of the pre-images of x_f lies in I_i. The "and" terms are necessary to eliminate intermediate points during continuous evolution from satisfying the query. The "or" term is necessary because p reaches only even iterates and q reaches only the odd iterates of x_0. The starting state is $(p = x_0 + L_{Q_k}, q = L_{Q_k} + b_k)$ (or $(p = L_{P_k} + b_k, q = L_{Q_k} + x_0)$), where $x_0 \in I_k$. □

Example 4.3.1 *We will now construct a PCD with translational resets that simulates the Tent Map, in two variables p and q and $2 \times 2 = 4$ states. Setting $L = 3(>$*

$max(r_n, b_i) = 2)$, *we get* $L_{P_1} = 3, L_{Q_1} = 9, L_{P_2} = 15, L_{Q_2} = 21$. *Thus:*

P_1: *flows* $\dot{p} = a_1 = 2$ *and* $\dot{q} = -1$, *with transitions:*

$\to Q_1$: *guard* $q = 3 \wedge p \in [3 + 0, 3 + 1/2)$, *reset* $p' = p - 3 + 9 = p + 6 \wedge q' = 9 + 0 = 9$

$\to Q_2$: *guard* $q = 3 \wedge p \in [3+1/2, 3+1]$, *reset* $p' = p - 3 + 21 = p + 18 \wedge q' = 21 + 2 = 23$

P_2: *flows* $\dot{p} = a_2 = -2$ *and* $\dot{q} = -1$, *with transitions:*

$\to Q_1$: *guard* $q = 15 \wedge p \in [15+0, 15+1/2)$, *reset* $p' = p - 15 + 9 = p - 6 \wedge q' = 9 + 0 = 9$

$\to Q_2$: *guard* $q = 15 \wedge p \in [15 + 1/2, 15 + 1]$, *reset* $p' = p - 15 + 21 = p + 6 \wedge q' = 21 + 2 = 23$

Q_1: *flows* $\dot{q} = a_1 = 2$ *and* $\dot{p} = -1$, *with transitions:*

$\to P_1$: *guard* $p = 9 \wedge q \in [9 + 0, 9 + 1/2)$, *reset* $q' = q - 9 + 3 = q - 6 \wedge p' = 3 + 0 = 3$

$\to P_2$: *guard* $p = 9 \wedge q \in [9+1/2, 9+1]$, *reset* $q' = q - 9 + 15 = q + 6 \wedge p' = 15 + 2 = 17$

Q_2: *flows* $\dot{q} = a_2 = -2$ *and* $\dot{p} = -1$, *with transitions:*

$\to P_1$: *guard* $p = 21 \wedge q \in [21+0, 21+1/2)$, *reset* $q' = q - 21 + 3 = q - 18 \wedge p' = 3 + 0 = 3$

$\to P_2$: *guard* $p = 21 \wedge q \in [21 + 1/2, 21 + 1]$, *reset* $q' = q - 21 + 15 = q - 6 \wedge p' = 15 + 2 = 17$

The result is presented in Figure *4.2.* □

4.3.2 Other Open Subclasses

Various other intermediates – subclasses of HPCDs, simulate a 1-dim PAM. We now present some of the interesting cases, which extend known decidable systems.

Theorem 4.3.2 *A 1-dim PAM can be simulated by:*

1. *an HPCD with comparative guards, 3 different flows* $+1, -1, 0$ *and no resets;*
2. *an initialized PCD, with comparative guards;*

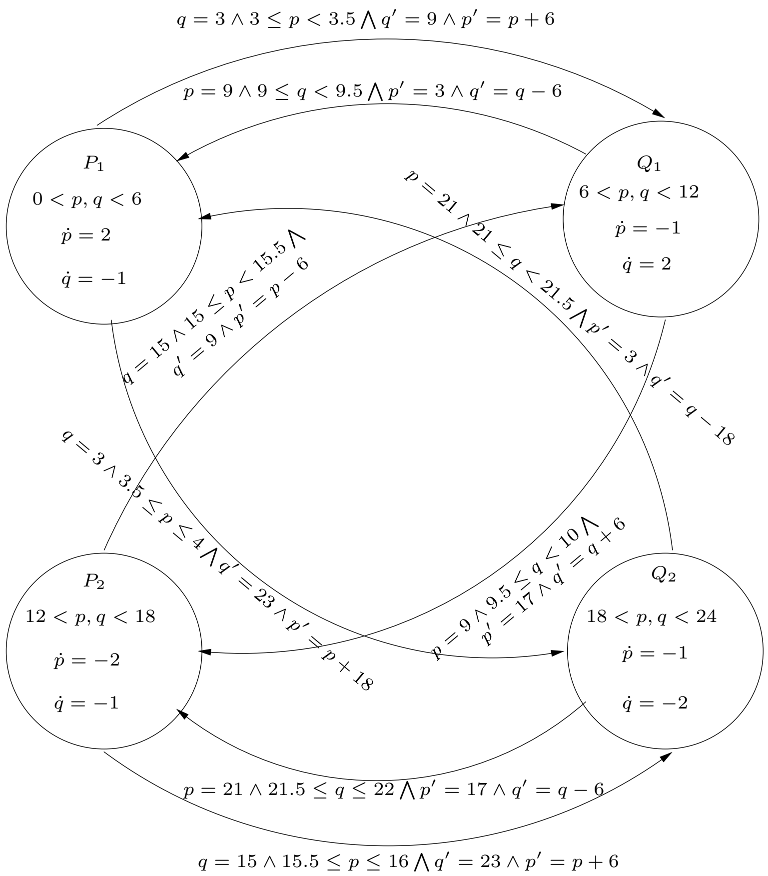

Figure 4.2: PCD with Translational Resets simulating the Tent Map

3. *an HPCD with rectangular guards, i.e.,* $p = 0 \wedge q \in I_i$, *when simple constant resets of the form* $q' = a_j \wedge p' = p$ *are allowed;*

4. *a PCD with just clocks, when translational resets and comparative guards are allowed; and*

5. *an HPCD with just clocks, when simple constant resets* $(p', q') = (0, q)$ *or* $(p, 0)$ *and comparative guards are allowed.*

Proof 4.3.2 *All the proofs are based on the techniques demonstrated in the proof of* Theorem *4.3.1. So, for brevity, we only give a flavor of results* (1) *and* (4).

For result (1)*, we will construct an HPCD with* $4n$ *states of the form* $P_j^{\pm}$ *and* $Q_j^{\pm}$ *that simulates this PAM. We will now have* p *evolving from* x_{n-1} *to* x_{n+1}*, while* q *remains stationary at* x_n*. The affine guard condition* $p = a_i q + b_i$ *makes the HA jump to the next state at the correct time. Since* x_{n+1} *could be greater or less than* x_{n-1}*, the flow will need to be* $+1$ *or* -1 *respectively. Hence, each* P *(and* Q*) state now corresponds to two states:* P^+ *and* P^-*. In the state* P_j^+*,* q *flows from* $q_0(\in$ *some* $I_i)$ *to* $q' = a_j p_0 + b_j$*, with flow is* $\dot{q} = +1$*. In* P_j^-*,* $q' > q_0$ *and* $\dot{q} = -1$*.* $\dot{p} = 0$ *to ensure that* q *flows to the correct amount. The transitions are of the form* $P_j^+ \rightarrow Q_k^{\pm}$*, with the guard being* $q = a_j p + b_j \wedge q \in I_k \wedge p < (b_k + a_k b_j)/(1 - a_j a_k)$*. The last term will be* $p \geq$ *if we are jumping to* Q_k^-*. The* $Q_j^{\pm}$ *states are defined exactly as above, with* p *and* q *interchanged. Clearly, the above HPCD without resets simulates the given PAM. In particular, the reachability query "*Is x_f reachable from x_0*" is true* iff $(p = x_f, q = x_{f-1})$ or $(p = x_{f-1}, q = x_f)$ is reachable from $(p = x_0, q = x_1)$*, where* x_{f-1} *is some pre-image of* x_f *and* x_1 *is the successor of* x_0*.*

For result (4)*, an HPCD with* $2n$ *states of the form* P_j *and* Q_j *can simulate the equivalent positive PAM. In state* P_j*,* p *flows from* p_0 *to* $p_0 + a_j p_0 + b_j$ *with* $\dot{p} = +1$*, while* q *flows from* 0 *to* $a_j p_0 + b_j$ *with* $\dot{q} = +1$*. The discrete transitions will be of the form* $P_j \rightarrow Q_k$ *with guard* $a_j p - (1 + a_j) q + b_j = 0 \wedge q \in I_k$ *and reset* $p' = 0 \wedge q' = q$*.*
□

HPCDs as HPCDs without Resets

We continue to focus on resets in an attempt to bring HPCDs closer to PCDs. Here we try to understand the simplest HPCD subclass that can simulate a PAM without using resets at all. Unfortunately, the best construction still seems to need overlapping invariants in addition to comparative guards.

Theorem 4.3.3 *A 1-dim PAM can be simulated by a 2-dim HPCD with comparative guards, 3 different flows* $+1, -1, 0$ *and no resets.* $\square$

Extensions of Initialized Rectangular Automata

In a similar vein, we prove here that decidable initialized rectangular automata [241, 144] can simulate a PAM, when extended with comparative guards or when uninitialized.

Theorem 4.3.4 *A 1-dim PAM can be simulated by an initialized PCD, with comparative guards.* $\square$

Theorem 4.3.5 *A 1-dim PAM can be simulated by an HPCD with rectangular guards, i.e.,* $p = 0 \wedge q \in I_i$*, when simple constant resets of the form* $q' = a_j \wedge p' = p$ *are allowed.* $\square$

Timed Automata HPCDs

Since the literature on timed automata [12, 9] is the largest, it makes sense to characterize their minimal extension that can simulate a PAM. Our construction below shows that we need comparative guards and resets.

Theorem 4.3.6 *A 1-dim bounded PAM can be simulated by a PCD with just clocks, when translational resets and comparative guards are allowed.* $\square$

Theorem 4.3.7 *A 1-dim PAM can be simulated by an HPCD with just clocks, when simple constant resets* $(p', q') = (0, q)$ *or* $(p, 0)$ *and comparative guards are allowed.* $\square$

1-dim HPCD with Zeno Executions

While a 2-dim $HPCD_{zeno}$ is undecidable, a 1-dim $HPCD_{zeno}$ can trivially capture a 1-dim PAM almost by definition. The guards will be of the form $x \in I_i$ and the resets of the form $x' = a_i x + b_i$ where the 1-dim PAM is of the form $f(x) = a_i x + b_i\ ,\ x \in I_i\ ,\ i = 1, 2, \cdots, n$. The flows are not used at all.

Theorem 4.3.8 *A 1-dim PAM can be simulated by a 1-state 1-dim HPCD with rectangular guards, affine resets and zeno-executions, without using the flows.* □

Origin-Dependent HPCDs without Resets or Comparative Guards

We find that we can do without resets or comparative guards, if we use origin-dependent rates and overlapping state-invariants.

Theorem 4.3.9 *A 1-dim PAM can be simulated by a 2-dim HPCD without comparative guards or affine resets using origin dependent flows.* □

4.4 Undecidable HPCD Extensions

True to its "open" nature, the HPCD class does not present any direct mechanism to simulate a TM / MM. Asarin and Schneider [39] have shown that the HPCD class becomes undecidable when extended with one additional counter ($HPCD_{1c}$), with infinite partition ($HPCD_{\infty}$) and with origin-dependent rates ($HPCD_x$).

Using the flow to simulate the counters of an MM is problematic, because we have no way of measuring one time unit. Both the variables are busy keeping track of the two independent counters, so one cannot be used as a temporary variable to increment / decrement the other. The other option is to store at least one of the counters in both the variables (or both counters in both variables). Finding such a mapping corresponds almost directly to the original PAM problem: Can we encode 2 unbounded positive integers (counters) and 1 bounded integer (the next instruction to be executed) in one real valued number, and still allow incrementing, decrementing and checking for equality to zero?

In this section, we present a new set of extensions of HPCDs that manage to be undecidable. We proceed by simulating the MM with the least possible additional work. Recall that an MM uses two (positive) integer counters m and n. Incrementing and decrementing a counter, and branching based on equality to zero are the operations that need to be supported.

Review of Minsky Machines

For the sake of completeness, we present the definitions of the Minsky Machine (MM) which we use in our undecidability proofs.

Definition 4.4.1 ***2-Counter Minsky Machine***[211] *A two counter Minsky Machine is a model of computation over two (positive) integer counters* m *and* n*, which support the following operations:*

- *Increment / Decrement a counter* $(m++, m--, n++, n--)$
- *Branching based on equality to zero (if* $m = 0/n = 0$ *go to line* l_i *else go to line* l_j*)* $\square$

Theorem 4.4.1 ***Undecidability Of MM*** *[211] A two-counter Minsky Machine is undecidable as it can simulate a Turing Machine.*

4.4.1 HPCDs with Zeno Executions

Any program over a 2-counter MM can be almost trivially captured as an HPCD, using just the discrete transitions without using the flows. Recall that a "zeno" system is one where one cannot bound the number of discrete transitions that can occur in a finite time window, i.e., potentially, all the computations could be done in the resets, in zero time. Thus:

Theorem 4.4.2 *Reachability over HPCDs with zeno-paths ($HPCD_{zeno}$) is undecidable.* □

Each statement of the program will correspond to one discrete state, with the transitions leading to the next state (line of code of an MM program) to be executed. Unfortunately, this "theorem" does *not* translate into "PAMs are also undecidable". This is because, the construction for simulating a HPCD using a PAM[39] is not applicable to states with no continuous flow (i.e., the "non-empty interior" criterion is invalid).

4.4.2 HPCDs with Integer-Checks

Alternatively, we can capture the value of both the counters m and n using just one continuous variable x as:

$$x = p_1^m p_2^n,$$

where p_1 and p_2 are two prime numbers. Clearly, given the integer product x, there is exactly one way of factoring it and hence m and n can be extracted. The second variable y is now free to be used as a temporary variable for computations and to make the system non-zeno. Incrementing and decrementing the counter correspond respectively, to multiplying and dividing by the appropriate prime factor. The problem of simulating a 2-counter MM over a HPCD now reduces to the problem of checking if $m > 0$ given the numerical value of $x = p_1^m p_2^n$, and being able to recover the original value of x at the end of the procedure. One approach is to divide x by the prime number corresponding to the counter we wish to check for zero, and then check if the result of the division is an integer. *The problem of simulating a 2-counter MM over an HPCD thus reduces to the problem of checking whether a given number is an integer using the 2-dim HPCD infrastructure, and being able to recover the original number at the end of the procedure.* Surprisingly, there is no known way of doing this.

Theorem 4.4.3 *Reachability over the following HPCD-extensions are undecidable:*

1. *$HPCD_{fn-int}$, where the guard can include a function $integer(x)$ that returns* true *if the parameter x is an integer.*

2. *$HPCD_{zeno-int}$, where the integer-check function is now simulated by a zeno execution of repeatedly subtracting* 1 *and checking if the number equals* 0. $\square$

Unlike the HPCD$_{zeno}$ class, the entire MM simulation is not a zeno-execution in HPCD$_{zeno-int}$. This system does spend 1 unit of time in each discrete state, except when this zeno integer-check "oracle" is consulted.

4.5 Understanding PAMs

Having refined the decidable and undecidable frontiers of the HPCD class, we explore one last avenue – treating HPCDs as PAMs, and subjecting them to a similar extend-restrain analysis.

4.5.1 PAM's Proximity to Undecidability

Just like we enhanced a 2-dim HPCD to make it undecidable, we present the flavor of a similar effort for PAMs.

Theorem 4.5.1 *1-dim PAMs that can check if a given number x can be expressed as p^{-i} (the class "PAM_{pow}"), where p is a given prime number and i is an unknown positive integer, can simulate an MM.* $\square$

We stop with this contrived extension, and move on to restricted subclasses.

4.5.2 PAM's Proximity to Decidability

The simplest PAM is one where every interval maps exactly on to another interval. Thus the mapping unwinds to a cyclical application of functions, possibly preceded by a linear sequence.

Definition 4.5.1 ***1-dim oPAM*** *A 1-dimensional Onto PAM (oPAM) is a 1-dim PAM where, for every interval I_i in the PAM definition, there is an interval I_j also in the definition such that $\{a_i x + b_i | x \in I_i\} = \{x | x \in I_j\}$.* $\square$

Next we prove a crucial lemma:

Lemma 4.5.1 *In a 1-dim oPAM with k intervals, every point has at most $2k$ unique successors.*

Proof 4.5.1 *If interval I_i maps on to I_j, the end points (l_i, r_i) have to map on to (l_j, r_j) or to (r_j, l_j). No other mapping is possible because of our restriction, that the affine post-image of I_i has to exactly and completely overlap with I_j. Hence, there are only two possible equations linking x_j with x_i:*

1. *Direct ($l_i \to l_j, r_i \to r_j$): $x_j = l_j + \frac{x_i - l_i}{r_i - l_i}(r_j - l_j)$*

2. Flipped ($l_i \to r_j, r_i \to l_j$): $x_j = l_j + \frac{r_i - x_i}{r_i - l_i}(r_j - l_j)$

In other words, if we define $d = \frac{x_0 - l_{x_0}}{r_{x_0} - l_{x_0}}$, only the points that are $l_j + d(r_j - l_j)$ or $l_j + (1-d)(r_j - l_j)$ are ever reachable. Thus, every interval has only two possible reachable points from a given x_0. Since there are k intervals, after $2k$ iterations all possible successors would have been explored, and there will be a cycle of period $\leq 2k$ in the path. $\square$

Using this observation about exactly onto affine maps over linear intervals, we can prove that:

Theorem 4.5.2 *Reachability is decidable for 1-dim oPAMs.* $\square$

Remark 4.5.1 *We can quickly conclude that x_f is unreachable if $\frac{x_f - x_{l_f}}{x_{r_f} - x_{l_f}}$ is not equal to $\frac{x_0 - x_{l_0}}{x_{r_0} - x_{l_0}}$ or $\frac{x_{r_0} - x_0}{x_{r_f} - x_{l_f}}$, where $[x_{l_f}, x_{r_f}]$ and $[x_{l_0}, x_{r_0}]$ are the interval partitions containing x_f and x_0 respectively.*

Example 4.5.1 *$f(x) = 2x + 1/3,\ x \in [0, 1/3)(\equiv I_1)$ and $f(x) = 1/2 - x/2,\ x \in [1/3, 1](\equiv I_2)$ is a oPAM as $f([0, 1/3]) = [1/3, 1]$ and $f([1/3, 1]) = [0, 1/3]$. Thus, all points reachable from $x_0 = 1/4$ are given by $x_1 = 2/4 + 1/3 = 5/6, x_2 = 1/2 - 5/12 = 1/12, x_3 = 2/12 + 1/3 = 1/2, x_4 = 1/2 - 1/4 = 1/4 = x_0$ as expected.* $\square$

4.5.3 An Approximate Reachability Algorithm

Reachability is easily semidecidable for PAMs: we just keep iterating x_0, $f(x_0)$, $f(f(x_0))$, $\cdots$ until x_f is reached. If x_f is not reachable, this algorithm will never converge. We now present a simple algorithm for over-approximating the reachable points (see *Alg.* 1 below). The idea is to repeatedly partition the intervals I_i of the PAM, until all the successors (post-images) of points in one interval map on to exactly one complete interval, i.e., domain and range are fully covered (an extension of this idea is presented in *Sec.* 7.2.1).

Algorithm 1 Over-Approximation of PAM Reachability

1. Let the initial set of partitions P be the set of PAM intervals $\{I_i\}$

2. Pick an interval P_i in P and calculate its post-image P_i'. Let P_i' span the intervals $P_l, P_{l+1}, \cdots, P_{r-1}, P_r$.

3. P_i' induces $r-l+1$ partitions of P_i: $P_{i_1} \cdots P_{i_{r-l+1}}$ such that P_{i_j} maps on to P_{l+j-1}. It could also partition P_l and P_r in case it maps on to a sub-interval rather than covering the whole of P_l or P_r. In all, the total number of partitions can increase by 0 to $n+1$.

4. Update P so it now holds the newly induced partitions as well.

5. Repeat steps $2-4$ until every interval P_i maps on to exactly one interval P_j already in P

By treating each interval as a node and connecting P_i and P_j if the post image of P_i is P_j, we get a graph representation of the PAM. Thus, x_f is reachable from x_0, if there is a path from P_{x_0} to P_{x_f} in this graph (where $x_i \in P_{x_i}$). □

Clearly, *Algorithm* 1 is not guaranteed to converge. However, we can terminate after a reasonable number of steps and still use the resultant graph to approximately decide reachability. Also note that the graph needs to be constructed only once no matter how many different reachability queries we need to answer. A rewarding observation is that a 1-dim oPAM is obtained, if the above partitioning algorithm converges; this concurs with the fact that oPAMs are decidable.

4.6 Discussion

In this chapter, we refined the decidability frontier by exploiting the expressive redundancy of the HPCD class definition. We introduced the "taking-turns" idea where the two PCD variables alternately compute PAM iterations. A similar idea was used by Berard and Duford to prove that the emptiness query is undecidable for timed automata with four clocks and additive clock constraints [61]. We also showed how we could exploit the finite range of the PAM to construct non-overlapping state invariants. These ideas helped show that a 1-dim PAM can be simulated by a 2-dim PCD with translational resets. Further, resets can be disposed, if we allow overlapping invariants and comparative guards. We also demonstrated how decidable classes, like timed and initialized rectangular automata, can be extended into open problems. On the undecidability front, we showed that zeno HPCD executions can naturally capture MMs. More interestingly, the ability to check if a number is an integer was seen to be the computational ability separating an HPCD from universal Turing computability. A simple algorithm for over-approximating reachability was presented. It revealed that the problem is decidable, for those PAMs that converge during this iteration (oPAMs). The current understanding of this undecidability frontier of HA is summarized in *Figure* 4.3 (results not marked with a "*" are contributions of this chapter).

There are many related questions that need to be explored. Using the reductions of an HPCD to a PCD or initialized rectangular automaton with extensions, can we identify more interesting decidable subclasses and approximate reachability algorithms? One suggestion we offer is to construct the "PCD-graph" of a 1-dim PAM, and then show how planarity can correspond to decidability. Using the construction in *Theorem* 4.3.1, we can build a graph with a set of nodes capturing each state.

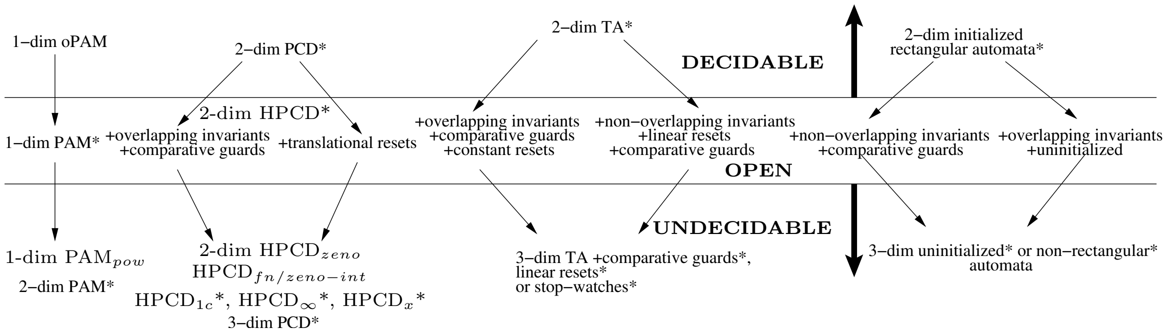

Figure 4.3: Decidable, Open and Undecidable subclasses of HA

Different nodes can correspond to the different intervals, with an edge corresponding to each reset. We can then check whether it is possible to rearrange the given PCD-graph into a planar graph, by using the standard graph drawing literature. Thus, we can prove that:

Theorem 4.6.1 *A 1-dim PAM is decidable if its PCD-Graph is planar.* □

Other important questions inviting research are: What is the least constrained 2-dim HPCD that is decidable? Can we sharper the decidability frontier currently defined by *initialized rectangular HPCDs* and *SPDIs*? In the process of showing that every known undecidable HA can simulate a 1-dim PAM, can we identify any direct undecidability construction? Can we equate a 1-dim PAM to one-stopwatch automata, which are apparently "open"? Another perspective to be explored is the influence of discrete chaotic dynamical systems (like the Tent Map) on the decidability of the PAM class. Alternatively, we could understand this problem in other models of computation. For example, Blum et al. [66] proposed the more general "real" Turing machine that has exact rational operations and comparison of real numbers built-in as atomic operations. Similarly, Saito and Kaneko [251] suggest a characterization of undecidability in terms of the inaccessibility to the ideal decision procedure and its invariance against fractal code transformations.

It is hoped that the ideas presented in this chapter will aid the eventual "deciding" of the 1-dim PAM / 2-dim HPCD reachability problem. From the Systems Biology perspective, it is clear that decidable hybrid automaton subclasses cannot hope to capture interesting biochemistry because of the low dimensionality requirement for decidability. Though many of the constructions for generating undecidable queries may seem artificial and unnatural, it nevertheless goes to prove that the solution is not generalizable. As our main goal was to provide general powerful methods rather than restricted efficient approaches, we continue our quest in the semi-decidable realm of hybrid automata in the next chapter.

Chapter 5

Semi-Algebraic Hybrid Systems

Decidable PCD-like hybrid automaton subclasses are not apt for Systems Biology applications, given the ease with which decidable subclasses become undecidable even in very low dimensional cases. We continue our inspection of hybrid automaton subclasses, this time focusing on subclasses for which at least a semi-decidable algorithm is known to exist. Once again, we find that most well-studied subclasses like timed and linear hybrid automata cannot approximate general biochemical systems reasonably. We find *O-Minimal* and *Polynomial* hybrid systems very attractive. O-minimal systems allow more complex continuous dynamics but do not allow interesting discrete behavior, while polynomial systems seem more than suitable. In this chapter, we extend this subclass to define the new subclass Semi-Algebraic Hybrid Automata. We discuss quantifier elimination and related algorithms in computational real algebraic geometry, which motivate the definition of this class. We present its syntax and semantics, and demonstrate the solution of the reachability problem. The semi-decidable algorithm becomes decidable if the physical hybrid system is non-zeno, and if the property we are interested can be verified by inspecting the system evolution up to a bounded time. Since both these assumptions are reasonable for biochemical networks, we have effectively identified a decidable bounded symbolic [1] reachability algorithm for biochemical networks modeled as semi-algebraic hybrid automata. We also show that semi-algebraic hybrid automata are undecidable even in the Blum-Shub-Smale model of real computation. This tangential result is significant because semi-algebraic sets appear naturally in the real Turing machine formalism. We conclude by comparing our approach with other approaches that use quantifier elimination for verification.

5.1 Background

5.1.1 Hybrid Automaton Subclasses

We first briefly go over the well-studied hybrid system subclasses in increasing order of system complexity:

- *Timed Automaton* - a discrete transition system where the only continuous variable allowed is the clock has been proved to be decidable for the reachability query [12].
- *Multirate Automaton* - a discrete transition system where there can be many continuous variables with a constant flow was also proved decidable by Alur et al. [9].
- *Rectangular Automaton* is a discrete transition system where the flows are allowed to vary within a range. Puri and Varaiya [241] and Henzinger et al. [144] proved that rectangular automata were amenable to reachability analysis.

[1]The ability to handle symbolic / algebraic parameters in our approach is not to be confused with traditional *symbolic* model-checking, where we refer to the use of binary decision diagrams (BDDs) as opposed to explicit enumeration of states as being *symbolic*.

- *Linear Systems* The reachability problem for sub-classes of linear hybrid systems was proved by Lafferriere et al. [185]. Tabuada and Pappas proved that model checking LTL over controllable linear systems is decidable [273, 274]. Tiwari et al. [279, 125] present techniques for over-approximating reachability sets for linear systems. The work of Alur et al. [10] also assume *linear predicates* and *linear dynamics* for the hybrid system. The abstraction is then built during the reachability analysis procedure using a numerical algorithm that checks polyhedral inclusions and memberships [35, 71]. The "linearity" assumption yields an efficient implementation of the reachability analysis procedure, while grossly approximating the reachable set computed.

- *Time-Invariant Systems* Tiwari et al. [278] have proposed a fully symbolic approach to the construction of the abstracted discrete system from a concrete hybrid system and have recently extended their work to non-linear systems [280]. However, they assume that the flow equations, guards and state invariants are all independent of time. These restrictions are fundamental to this approach and allow symbolic computations with the integrals of the system.

- *O-Minimal Systems* are another class of algebraic hybrid systems which have been shown to be amenable to reachability analysis [185]. While the other cases of hybrid automata modeling restrict continuous dynamics to derive a bisimulation, here the discrete dynamics are restricted. Indeed, as per the restricted *jump* condition the new continuous state cannot depend on the old state. Also, the system is assumed to be time-invariant so that the trajectories cannot self-intersect. These restrictions effectively decouple the discrete and continuous dynamics allowing bisimulations to be performed on each state separately. O-minimal theories which describe finite unions of points and open intervals, generalize the notion of semi-algebraic sets. They have nice topological properties, but not complete quantifier elimination. Instead they have a weaker property known as *model completeness*.

- *Polynomial Systems* [121, 122] extend linear hybrid automata by allowing polynomial (instead of only linear) activities and polynomial (instead of linear) predicates for state invariants, transition guards, and transition effects. Every state and transition is described by just one polynomial predicate, formalized through the first-order logic over the real-closed field, denoted $FOL(\mathbb{R}, +, .)$. Formally, a (polynomial) hybrid automaton of dimensionality $d(d \in \mathbb{N})$ is a five-tuple: $(\Sigma, x, (act_\sigma)_{\sigma\in\Sigma}, (trans_{\sigma\rightarrow\sigma'})_{\sigma,\sigma'\in\Sigma}, (initial_\sigma)_{\sigma\in\Sigma})$, where Σ is a finite set, representing the discrete states, and $x = (x_1, \cdots, x_d)$ is a vector of length d of variable names, the continuous variables of the hybrid system. $(act_\sigma)_{\sigma\in\Sigma}$ is a Σ-indexed family of formulae from $FOL(\mathbb{R}, +, .)$ representing the continuous activities and corresponding state constraints, and $(trans_{\sigma\rightarrow\sigma'})_{\sigma,\sigma'\in\Sigma}$ is a doubly Σ-indexed family of formulae from $FOL(\mathbb{R}, +, .)$ representing the discrete transitions and their guarding conditions. Finally, the initial states of the automaton are given by the Σ-indexed family $(initial_\sigma)_{\sigma\in\Sigma}$ of formulae from $FOL(\mathbb{R}, +, .)$. The free variables of the $initial_\sigma$ predicates are from x, while the free variables of the act_σ and the $trans_{\sigma\in\sigma'}$ predicates are from $\bar{x}$ and x, as these predicates relate pre-states, which are denoted by the $\bar{x}$, to post-states, which are denoted by undecorated variables x. The decidability of $FOL(\mathbb{R}, +, .)$ [276] is used to prove semi-decidability of reachability, recurrence and path-boundedness, and undecidability of several dynamic properties and robustness.

Since biochemical system dynamics are described by ordinary differential equations often involving quadratic or higher degree terms, timed, multirate, rectangular and linear systems get immediately eliminated from consideration as they offer only linear approximation based schemes for handling complex dynamics. Clearly, *O-minimal systems* afford the maximum freedom in terms of continuous dynamics as they allow exponential terms as well. However, all their transitions involve constant resets, which is an unnatural requirement for biochemical networks. *Polynomial systems* on the other hand definitely offer the ideal mix of discrete and continuous dynamics, restricting all expressions to be polynomials. In this chapter, we extend this class to create the new class "semi-algebraic hybrid systems", where all expressions are restricted to be semi-algebraic sets. Further, we handle more complex flow expressions and show how symbolic integration can be used to handle ordinary differential equations as well. We first survey the fundamentals of real algebraic geometry relevant to our discussion in the following section, and then define the syntax and semantics more formally.

5.1.2 Computational Real Algebraic Geometry

The symbolic manipulation of the variables of the hybrid system requires mathematical infrastructure that deals with variables (algebra) as opposed to numbers (arithmetic). Polynomials naturally appear in the Systems Biology domain in the kinetic mass action equations described earlier. They are equally prevalent in most engineering domains as well. The algorithmic manipulation of the Boolean combination of polynomial equations and inequalities is the goal of computational real algebraic geometry.

Semi-Algebraic Sets

The conventional algebraic sets correspond to equations of polynomials (see *Sec.* 8.4.1 for an introduction to Gröbner bases, used for simplifying algebraic sets). Their generalization is the semi-algebraic concept.

Definition 5.1.1 Semi-Algebraic Set[214] *Every quantifier-free Boolean formula composed of polynomial equations and inequalities defines a semi-algebraic set (i.e., unquantified first-order formulæ over the reals -* $(\mathbb{R}, +, \times, =, <)$*). Formally, a set* S *is semi-algebraic if:*

$$S = \{\langle \xi_1, \cdots, \xi_n \rangle \in \mathcal{R}^n | \psi(\xi_1, \cdots, \xi_n) = True\}, \text{ or}$$

$$S = \bigcup_{i=1}^{I} \bigcap_{j=1}^{J_i} \{\langle \xi_1, \cdots, \xi_n \rangle \in \mathcal{R}^n | sign(f_{i,j}(\xi_1, \cdots, \xi_n)) = s_{i,j}\}$$

where $\psi(\xi_1, \cdots, \xi_n)$ *is a quantifier-free formula involving* n *algebraic variables,* $f_{i,j}s$ *are multivariate polynomials over* R *and the* $s_{i,j}s$ *are corresponding sets of signs in* $\{-1, 0, +1\}$. □

Quantifier Elimination

Quantifier elimination is one of the most powerful algorithms over semi-algebraic sets. Their generality effectively translates to its all-encompassing trait, allowing diverse and deep questions to be encoded as quantifier elimination problems. Tarski[276] proved that a semi-algebraic set results from the existential ($\exists x$) or universal ($\forall x$) quantification over any variable ($x \in X$) of a given semi-algebraic set $S(X)$, by providing a computable procedure for performing this quantifier elimination. In other words, the decision problem for the first order theory of reals was shown to be decidable, though the algorithm itself was almost impractical.

Cylindrical Algebraic Decomposition Let $\mathcal{F}$ be a set of real multivariate polynomials in n variables. Collins [90] discovered the cylindrical algebraic decomposition (CAD) technique, which partitions the state space $\mathbb{R}^n$ into "cells" within which the signs of all the polynomials in $\mathcal{F}$ are invariant. Effectively, a set of points guaranteed to be characteristic of all possible sign behaviors of the given set of polynomials can be obtained. Thus, existential quantification translates to the semi-algebraic formula evaluating to true, at least at one point in the set, while universal quantification is equivalent to the formula being true at all points in the set. Collins' doctoral student Hong developed the first quantifier elimination software Qepcad [151], which has subsequently been extended and maintained by a larger team.

For the convenience of the interested reader, we now summarize the mathematical details of this technique from [214]. (This section may also be skipped[2], and the reader may choose to proceed to the *Beyond CAD* section.)

Definition 5.1.2 ***Sign Assignment and Class*** *[214]* *Any point p equal to $\langle \eta_1, \cdots, \eta_n \rangle \in \mathbb{R}^n$ has a sign assignment with respect to $\mathcal{F}$ as follows:*

$$sgn_{\mathcal{F}}(p) = \left\langle sgn(f(\eta_1, \cdots, \eta_n)) | f \in \mathcal{F} \right\rangle.$$

A sign assignment induces an equivalence relation: given two points $p, q \in \mathbb{R}^n$, $p \sim_{\mathcal{F}} q$, if and only if $sgn_{\mathcal{F}}(p) = sgn_{\mathcal{F}}(q)$. The sign class *of $\mathcal{F}$ is an equivalence class in the partition of $\mathbb{R}^n$ defined by the equivalence relation $\sim_{\mathcal{F}}$.* □

[2]The mathematical details provided here are not essential for the appreciation of this thesis, as we just use quantifier elimination as a black-box tool in our methodology. However, one of the lines of future research is developing efficient approximate quantifier elimination strategies that exploit the inherent structure of the problems we are likely to encounter in the algebraic model checking of biochemical systems.

Definition 5.1.3 ***Semi-Algebraic Decomposition*** *[214]* *A finite collection of disjoint connected semi-algebraic subsets C_i such that each C_i is contained in some semi-algebraic sign class of $\mathcal{F}$. That is, the sign of each $f \in \mathcal{F}$ is* invariant *in each C_i. The collection of connected components of the sign-invariant sets for $\mathcal{F}$ forms a semi-algebraic decomposition for $\mathcal{F}$.* □

Definition 5.1.4 ***Cell Decomposition*** *[214]* *is a semi-algebraic decomposition for $\mathcal{F}$ into finitely many disjoint semi-algebraic subsets C_i called* cells*, such that each cell C_i is homeomorphic to $\mathbb{R}^{\delta(i)}, 0 \leq \delta(i) \leq n$. $\delta(i)$ is called the* dimension of the cell *C_i, and C_i is called a $\delta(i)$-cell.* □

Definition 5.1.5 ***Cylindrical Algebraic Decomposition*** *[214]* *is a recursively defined cell decomposition of $\mathbb{R}^n$ for $\mathcal{F}$. Every $(n-1)$-dimensional CAD cell C' has the property that the distinct real roots of $\mathcal{F}$ over C' vary continuously as a function of the points of C'.*

Let $\phi(\mathcal{F})$ be a set of at most $O((md)^2)$ polynomials of degree no more than d^2 in $n-1$ variables. By considering principal subresultant coefficients*, it is so defined as to ensure that the polynomials of $\mathcal{F}$ do not intersect or "fold" in a cylinder over an $(n-1)$-dimensional cell. Further, the following quantities need to remain invariant over an $(n-1)$-dimensional cell: (1) the total number of complex roots of each polynomial of $\mathcal{F}$; (2) the number of distinct complex roots of each polynomial of $\mathcal{F}$; and (3) the total number of common complex roots of every distinct pair of polynomials of $\mathcal{F}$.*

An $\mathcal{F}$-sign-invariant cylindrical algebraic decomposition of $\mathbb{R}^n$ is:

- Base Case*: $n = 1$. A univariate cellular decomposition of R^1.*
- Inductive Case*: $n > 1$. Let $\mathcal{K}'$ be a $\phi(\mathcal{F})$-sign-invariant CAD of R^{n-1}. For each cell $c' \in \mathcal{K}'$, define an* auxiliary polynomial $g_{C'}(x_1, \cdots, x_{n-1}, x_n)$ *as the product*

of those polynomials of $\mathcal{F}$ that do not vanish over the $(n-1)$-dimensional cell C'. The real roots of the auxiliary polynomial g'_C over C' give rise to a finite number (perhaps zero) of semi-algebraic continuous functions, which partition the cylinder $C' \times (\mathbb{R} \cup \pm\infty)$ into finitely many $\mathcal{F}$-sign-invariant "slices". The auxiliary polynomials are of degree no larger than md.

Assume that the polynomial $g'_C(p', x_n)$ has l distinct real roots for each $p' \in C'$: $r_1(p'), r_2(p'), \cdots, r_l(p')$, each r_i being a continuous function of p'. The following sectors and sections are cylindrical over C':

$$C_0^* = \{\langle p', x_n\rangle | p' \in C' \wedge x_n \in [-\infty, r_1(p'))\}$$

$$C_1 = \{\langle p', x_n\rangle | p' \in C' \wedge x_n \in [r_1(p'), r_1(p')]\}$$

$$C_1^* = \{\langle p', x_n\rangle | p' \in C' \wedge x_n \in (r_1(p'), r_2(p')]\}$$

$$\vdots$$

$$C_l^* = \{\langle p', x_n\rangle | p' \in C' \wedge x_n \in (r_l(p'), +\infty]\}$$

The n-dimensional CAD is the union of all the sections and sectors computed over the cells of the $(n-1)$-dimensional CAD. The algorithm is polynomial in $m = |\mathcal{F}|$ and $d = deg(\mathcal{F})$, and doubly-exponential in n, as the number of polynomials produced at the lowest dimension is $(md)^{2^{O(n)}}$, each of degree no larger than $d^{2^{O(n)}}$. The number of cells produced is also doubly-exponential. □

Beyond CAD The real limitation of quantifier elimination is its doubly-exponential complexity lower bound [98], and is manifest in Collins' CAD algorithm which has a double-exponential dependence on the number of variables [90]. Alternative CAD-based methods have been proposed by Grigoriev [136], Renegar [245] and Heintz et al. [141] that are doubly exponential in the number of *quantifier alternations* rather than the number of *variables*. Weispfenning's work on cubic quantified variables [294, 295] has been implemented on the computer logic system *Reduce* as "Redlog" [109] and "Risa/Asir 1" by Sturm [272]. It has been shown to solve problems that CAD-based methods could not, as its complexity is independent of the number of free variables. New quantifier elimination approaches such as those of Basu, Pollack and Roy [50, 51, 49] are yet to be implemented. Another source of speed-up is paralellization of the underlying algorithms, which is also being actively investigated [254, 107, 207].

5.2 Semi-Algebraic Hybrid Automata

The notion of *hybrid automata* was first introduced as a model and specification language for systems with both continuous and discrete dynamics, i.e., for systems consisting of a discrete program within a continuously changing environment. A useful restriction is through the notion of *semi-algebraic hybrid automata* whose defining conditions are built out of polynomials over the reals, and reflect the algebraic nature of the differential algebraic equations appearing in kinetic mass-action models of regulatory, metabolic and signal transduction processes.

Definition 5.2.1 Semi-Algebraic Hybrid Automata *A k-dimensional* hybrid automaton *is a 7-tuple, $H = (Z, V, E, Init, Inv, Flow, Jump)$, consisting of the following components:*

- *$Z = \{Z_1, \dots, Z_k\}$ and $Z' = \{Z'_1, \dots, Z'_k\}$ are two finite sets of variables ranging over the reals $\mathbb{R}$*
- *(V, E) is a directed graph of discrete states and transitions*

- *Each discrete state $v \in V$ is labeled by "Init"(initial), "Inv"(invariant) and "Flow" labels of the form $Init_v[Z]$, $Inv_v[Z]$, and $Flow_v[Z, Z', t, h]$*

- *Each edge $e \in E$ is labeled by a "Jump" condition of the form $Jump_e[Z, Z'] \equiv Guard_e(Z) \wedge Reset_e(Z, Z')$*

- *Init, Inv, Flow, and Jump are semi-algebraic.* $\square$

We say that H is semi-algebraic *if the constraints in Init, Inv, Flow, and Jump are unquantified first-order formulæ over the reals (i.e., over $(\mathbb{R}, +, \times, =, <)$).* $\square$

Definition 5.2.2 Semantics of Hybrid Automata *Let $H = (Z, V, E, Init, Inv, Flow, Jump)$ be a hybrid automaton of dimension k.*

- *A* location *ℓ of H is a pair $\langle v, R\rangle$, where $v \in V$ is a discrete state and $R \in \mathbb{R}^k$ is an assignment of values to the variables of Z. A location $\langle v, R\rangle$ is said to be* admissible, *if $Inv_v(R)$ is satisfied.*

- *The* continuous reachability transition relation *$\xrightarrow[\mathcal{C}]{h}$ forces the discrete state invariant to hold at every location except the end-location, along the evolution curve determined by the flow equations during the $h(> 0)$ time units from the current time t_0:*

$$\langle v, R\rangle \xrightarrow[\mathcal{C}]{h} \langle v, S\rangle \quad \textit{iff}$$
$$\Big(Flow_v(R, S, t_0, h) \ \wedge\ \forall Z', h' \in [0, h)\ Flow_v(R, Z', t_0, h') \Rightarrow Inv_v(Z')\Big),$$

where $Flow_v(Z, Z', T, h)$ is the flow label of v.

- *The* discrete reachability transition relation *$\xrightarrow[\mathcal{D}]{0}$ ensures that both parts of the* zero-time *jump – the guard condition which needs to be satisfied just before the* *transition is taken, and the reset condition which determines the values after the transition, are satisfied. By definition, discrete state transitions take* zero *time. Thus:*

$$\langle v, R\rangle \xrightarrow[\mathcal{D}]{0} \langle u, S\rangle \qquad \textit{iff} \qquad \langle v, u\rangle \in E \ \wedge \ Jump_{v,u}(R, S).$$

- *The* transition relation $\mathcal{T}$ *of* H *connects the possible values of the system variables before and after* one step *- a discrete step for a time* $h = 0$ *or a continuous evolution for any time period* $h > 0$*:*

$$\mathcal{T}(\ell \xrightarrow{h} \ell') = \ \{h = 0 \wedge \ell \xrightarrow[\mathcal{D}]{0} \ell'\} \ \vee \ \{h > 0 \wedge \ell \xrightarrow[\mathcal{C}]{h} \ell'\}.$$

- *A* trace *of* H *is a sequence* ℓ_0,ℓ_1, ..., ℓ_n, ... *of admissible locations such that*

$$\forall i \geq 0,\ \exists h_i \geq 0,\ \mathcal{T}(\ell_i \xrightarrow{h_i} \ell_{i+1}). \quad \Box$$

The trace of a hybrid automaton is visualized in *Figure* 5.1

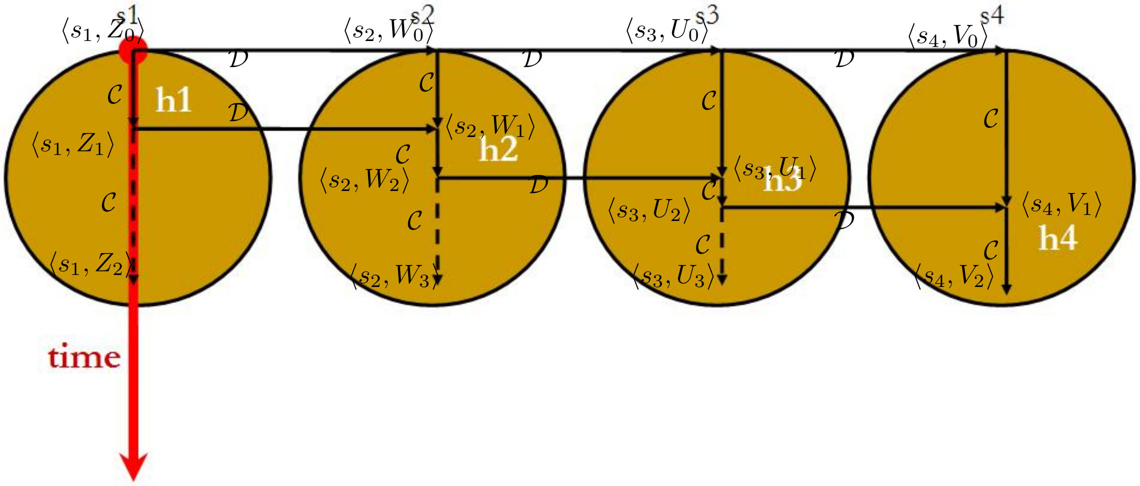

Figure 5.1: Trace of a Hybrid Automaton

Remark 5.2.1 *Few remarks about this definition of* trace *are in order: It admits two continuous transitions to occur consecutively, which is necessary for compositionality of traces. Further, two consecutive continuous transitions of time-steps* h_1 *and* h_2 *are* not *necessarily equivalent to one continuous transition of time-step* $h_1 + h_2$ *in the case of non-linear approximation errors in* $\xrightarrow[C]{h}$.

$Flow_v(R, S, t, h)$ *is a relation between the continuous state* R *at time* t *and the continuous state* S *after* h *time units in the discrete state* v. *It is "well-defined" in the sense that* $\forall R, S, t, h \; Flow_v(R, S, t, h) \Rightarrow \{\forall h' \in [0, h) \; \exists S' \; Flow_v(R, S', t, h')\}$.

When a semi-algebraic *relation* $Flow_v(R, S, t, h)$ *is used between the continuous states* R *at time* t *and* S *at time* $t + h$ *in a discrete state* v*, it may have been "derived" in two ways: (1)* Solution Is A Polynomial*: The equation describing the continuous evolution of the variables in a discrete state is a polynomial, say* $Y(t)$*, and* $Flow_v(Z, Z', t, h) \equiv \{ \; Z = Y(t) \;\; \wedge \;\; Z' = Y(t + h) \; \}$. *Or, (2)* Differential Equation Is A Polynomial*: Differential equations describing the continuous evolution are* approximated *in* $Flow_v$ *using one of the symbolic integration schemes (e.g., the* Taylor *series in [239] or based on a direct integration scheme such as the linear* Euler *or the higher degree* Runge-Kutta*). The error is controlled by an upper bound (say* Δ*) on the time spent in one continuous step, as we aim for over- or under-approximating the flow equations. In particular, if at each step the derivatives of order* $j + 1$ *of the involved flows are bounded on the set of points satisfying the invariant conditions, the Lagrange Remainder Theorem can be exploited to estimate error (see [189]).*

Example 5.2.1 Euler Forward Method *The first order Taylor polynomial corresponds to the Euler forward method which approximates the local flow curve as a straight line with slope equal to the first derivative. Thus, if* R *represents the vector of variables of the hybrid system at time* t *and* $\dot{R}$ *the vector of first derivatives expressed* *as a polynomial in R and t, the approximate value S of $R(t+h)$ can be obtained with an $O(h^2)$ error as $Flow_v(R, S, t, h) \equiv \{S = R + h.\dot{R}\}$.*

Note 5.2.1 *We say we are discretizing time only if we are going to observe the system every Δ time units (and not any intermediate points) and use those samples alone to make temporal inferences. More often than not, discretization also leads to several intermediate discrete transitions becoming impossible. Thus, enforcing an upper-bound is* not *equivalent to discretizing time as* all *possible intermediate points and discrete transitions are considered.*

Example 5.2.2 *Consider a biochemical system where initially (state Q_1) the biochemicals A, B, C and D are quiescent, i.e., their time derivatives are zero. In response to an external event that raises their concentrations instantaneously (by amounts e_A, e_B, e_C and e_D respectively), the system moves to state Q_2 if sufficient A exists ($A \geq l_A$) in the system. In Q_2, the conversion of A to its active trimer form B occurs as per $3A \rightarrow B$. When enough B accumulates ($B \geq l_B$), the biochemical reaction $B + C \rightarrow 2D$ is triggered as the system transitions to state Q_3, if enough C exists ($C \geq l_C$). Alternatively, the external addition of A, B, C and D could have directly caused a transition to state Q_3 from Q_1. A semi-algebraic hybrid automaton that captures the dynamics of these 4 biochemicals is presented in* Figure *5.2. For clarity, the discrete state transitions in the reverse direction are not shown. The* flow *expressions have been derived by applying the Euler forward scheme to the ODEs that result from the kinetic mass action laws.*

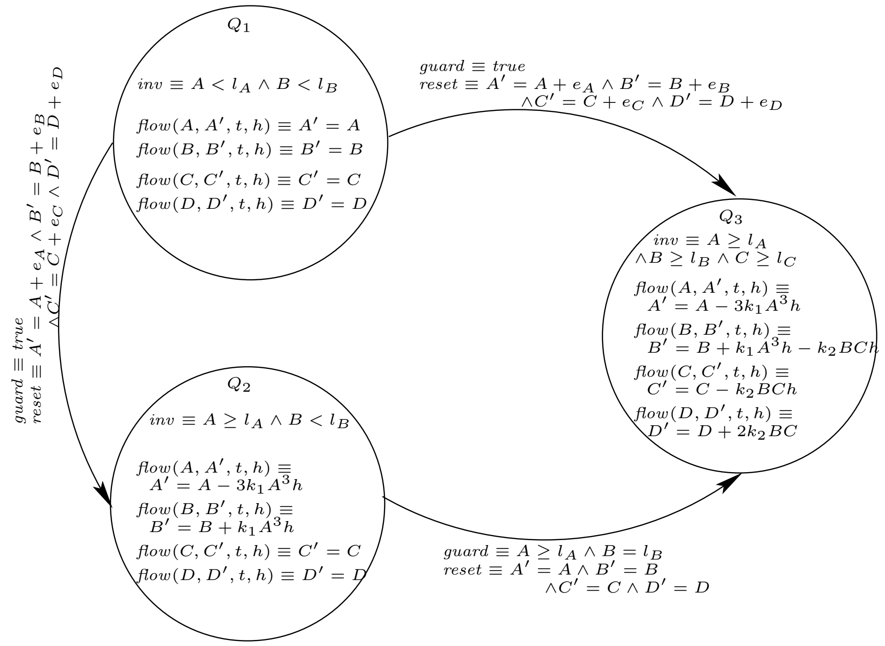

Figure 5.2: A 3-State Semi-Algebraic Hybrid Automaton

5.3 Reachability

Reachability is the problem of deciding if the hybrid automaton can reach the symbolic “final” state $\mathbb{B}$ from the “start” state $\mathbb{S}$. In the bounded version of the problem, the question is whether $\mathbb{B}$ can be reached within a given time bound `end`. In this section, we explore solutions to the bounded-reachability problem through symbolic computation methods, applied to the descriptions of the traces of the hybrid automaton. Semi-algebraic decision procedures provide a succinct description of algebraic constraints over the initial values and parameters for which proper behavior of the system can be expected. In addition, by keeping track of conservation principles (e.g., of mass and energy) in terms of constraint or invariant manifolds on which the system must evolve, we avoid many of the obvious pitfalls of numerical approaches.

Note also that the “Algorithmic Algebraic Model Checking” approach proposed here naturally generalizes many of the basic ideas inherent to BDD-based symbolic model checking or even the more recent SAT-based approaches. As the AAMC procedure involves proceeding along the traces using a time step δ, the answer to the reachability query is relative to this limited time interval. However, by existentially quantifying over every possible intermediate time-point ($\exists \delta',\ 0 \leq \delta' \leq \delta$), the problem of inadvertent discretization is overcome. As with numerical-simulation based ODE analysis, when the solutions of the differential equations cannot be computed, approximate is resorted to; hence the error accumulated also depends on δ.

Algorithmic Algebraic Bounded Reachability Approach

The first-order formula

$$\mathbb{F}[v, \mathbb{S}](Z_0, Z) \ \equiv \{\mathbb{S}(Z_0) \ \wedge \ \exists \delta', \left((0 \leq \delta' \leq \delta) \wedge \langle v, Z_0 \rangle \xrightarrow[\mathcal{C}]{\delta'} \langle v, Z \rangle \right)$$

characterizes the points reached within time δ in the discrete state v from the initial state $\mathbb{S}$, under the approximation implied by the use of $Flow_v(Z, Z', T, h)$, the flow equations of v. Thus, the formula

$$\exists Z_0(\mathbb{F}[v, \mathbb{S}](Z_0, Z)) \ \wedge \ \mathbb{B}(Z)$$

is satisfiable if and only if the set $\mathbb{B}$ can be reached from $\mathbb{S}$ without leaving discrete state v within the time step δ. In this case, the points of $\mathbb{S}$ which reach $\mathbb{B}$ are characterized by $\exists Z(\mathbb{F}[v, \mathbb{S}](Z_0, Z) \wedge \mathbb{B}(Z))$. If the preceding formula is not satisfiable, we have to consider all possible alternative situations: that is, either we continue to evolve within the discrete state v or we discretely jump to another discrete state, $u \in V$. We define the formula $\mathbb{S}_\delta^{vu}$

$$\mathbb{S}_\delta^{vu}(Z) \ \equiv \ \begin{cases} \exists Z_0(\langle v, Z_0 \rangle \xrightarrow[\mathcal{C}]{\delta} \langle v, Z \rangle), & \text{if } u = v; \\ \exists Z_0, Z_1(\langle v, Z_0 \rangle \xrightarrow[\mathcal{C}]{\delta} \langle v, Z_1 \rangle \wedge \langle v, Z_1 \rangle \xrightarrow[\mathcal{D}]{0} \langle u, Z \rangle), & \text{otherwise.} \end{cases}$$

representing the states reached within time δ in the discrete state u. In this way, in the worst case we generate $|E|$ satisfiable formulæ on which we have to iterate the method, treating them as we treated $\mathbb{S}(Z)$ in the first step. In practice, many of these formulæ would be unsatisfiable, and hence at each iteration, the number of formulæ we have to consider will remain considerably low. We may also use an optimized traversal over the graph to reduce the number of generated formulæ.

Let end be the total amount of time during which we examine the hybrid system's evolution in terms of at most $m = \lceil \mathsf{end}/\delta \rceil$ time steps: the number $m \in \mathbb{N}$ is such that $(m-1)\delta < \mathsf{end} \leq m\delta$. Since at each iteration the jumps can occur before δ instants of time have passed, just iterating the method for m steps does not ensure that we have indeed covered the entire time interval $[0, \mathsf{end}]$. In particular, if there are Zeno paths starting from $\mathbb{S}$, i.e., paths in which the time does not pass since only the jumps are used, our method will fail to converge in a finite number of steps. For these reasons, at each step, we must check the minimum elapsed time before a jump can be taken[3]. Let $\mathbb{M}(Z) \equiv \mathbb{S}^{v,u\dots,w}(Z)$ be one of the formulæ obtained after some number of iterations. Suppose now that we intend to jump from this discrete state w to the next discrete state z. We will then need to check whether the minimum amount of time has passed before the jump can be taken. Consider the formula:

$$\mathbb{T}(w,z,\mathbb{M})(T) \equiv \exists Z_0, Z_1, Z\Big(\mathbb{M}(Z_0) \ \wedge\ \langle v, Z_0\rangle \xrightarrow[\mathcal{C}]{T} \langle v, Z_1\rangle \ \wedge\ \langle v, Z_1\rangle \xrightarrow[\mathcal{D}]{0} \langle w, Z\rangle\Big).$$

The minimum amount of time can now be computed as a solution of the formula

$$\mathit{Min}(w,z,\mathbb{M})(T) \ \equiv\ \mathbb{T}(w,z,\mathbb{M})(T) \ \wedge\ \forall T'\Big(T' < T \to \neg\mathbb{T}(w,z,\mathbb{M})(T')\Big).$$

To avoid Zeno paths, we could eliminate the paths in which the minimum is 0. Along

[3]Approaches based on time discretization and bounding the number of discrete transitions are discussed in *Sec.* 7.5

each generated path we have to iterate until the sum of the minimum amounts reaches end. If all the paths accumulate a total amount of time greater than end and $\mathbb{B}$ is never reached we can be sure that $\mathbb{B}$ cannot be reached from $\mathbb{S}$ in the time interval $[0, \mathsf{end}]$. If $\mathbb{B}$ is reached, i.e., one of the formulæ involving $\mathbb{B}$ is satisfiable before m iterations, then we can be sure that $\mathbb{B}$ is reachable from $\mathbb{S}$ in the time interval $[0, \mathsf{end}]$. If $\mathbb{B}$ is reached after the first m iterations, then $\mathbb{B}$ is reachable from $\mathbb{S}$ but we are not sure about the elapsed time, since we keep together flows of different length. It is possible that some paths do not accumulate a total time greater than end, e.g., the sequence of the minimum times converges rapidly to 0. In this case our method could not converge. Notice that even in this general case, we can extend the method to rational flows.

In order to provide a time-complexity, assume the special situation where no path accrues more than M discrete jumps (i.e., our method has converged). When we terminate, we are left with deciding the satisfiability of a quantified semi-algebraic formula with $O(M)$ alternations and involving $n = k \cdot [\lceil \mathsf{end}/\delta \rceil + O(M)] + N(\mathbb{S}) + N(\mathbb{B})$ variables in degree $d = \max[j + \deg(\mathit{Init},\ \mathit{Inv},\ \mathit{Jump}), \deg(\mathbb{S}), \deg(\mathbb{B})]$, where N and deg denote the number of variables and total degree, respectively as before. Assume that the coefficients of the polynomials can be stored with at most L bits. Then the total time complexity (bit-complexity) [212, 214, 298] of the decision procedure is $(L \log L \log \log L) d^{2^{O(n)}}$, i.e., double-exponential in the number of variables.

5.4 General Undecidability of Reachability

5.4.1 Real Turing Machines

The major problem with current approaches that analyze continuous dynamical systems is that the classical theory of computation and complexity analysis centered around the "binary" Turing machine is not sufficient to fully characterize problems involving real-valued mathematics. Blum et al. [66] proposed the more general "real" Turing machine that has exact rational operations and comparison of real numbers built-in as atomic operations. This allows the algebraic complexity analysis of algorithms on reals.

Definition 5.4.1 Finite-Dimensional Machine Over $\mathcal{R}$: [66]. *A finite dimensional machine M over $\mathcal{R}$ consists of a finite directed connected graph with four types of nodes:* input, computation, branch *and* output*. The unique input node has no incoming edges and only one outgoing edge. All other nodes have possibly several incoming edges. Computation nodes have only one outgoing edge, branch nodes exactly two,* Yes *and* No*, and output nodes none. In addition the machine has three spaces:* input space $\mathcal{I}_M$, state space $\mathcal{S}_M$ *and* output space $\mathcal{O}_M$ *of the form $\mathcal{R}^n, \mathcal{R}^m, \mathcal{R}^l$, respectively, where n, m and l are positive integers. Associated with each node of the graph are maps of these spaces and* next node *assignments.*

1. *Associated with the* input node *is a linear map $I : \mathcal{I}_M \to \mathcal{S}_M$ and a unique next node β_1.*

2. *Each* computation node η *has an associated* computation map*, a polynomial (or rational) map $g_\eta : \mathcal{S}_M \to \mathcal{S}_M$ given by m polynomials (or rational functions) $g_j : \mathcal{R}^m \to \mathcal{R}, j = 1, \cdots, m$, and a unique next node β_η. If $\mathcal{R}$ is a field, g_η can*

be a rational map. If g is a rational map associated with a computation node (in the case $\mathcal{R}$ is a field), we assume each g_j is given by a fixed *pair of polynomials* (p_j, q_j)*, where* $g_j(x) = (p_j(x))/(q_j(x))$.

3. *Each* branch node η *has an associated* branching function*, a nonzero polynomial function* $h_\eta : \mathcal{S}_M \to \mathcal{R}$*. The next node along the* Yes *outgoing edge,* β_η^+*, is associated with the condition* $h_\eta \geq 0$ *and the next node along the* No *outgoing edge,* β_η^-*, with* $h_\eta(z) < 0$.

4. *Each* output node η *has an associated linear map* $\mathcal{O}_\eta : \mathcal{S}_M \to \mathcal{O}_M$ *and no next node.* □

Theorem 5.4.1 Path Decomposition Theorem: [66]. *For any machine* $\mathcal{M}$ *over* $\mathcal{R}$ *the following properties hold.*

1. *For any* $T > 0$*, the* time-T halting set *of* $\mathcal{M}$*:* Ω_T $(= \bigcup_{\gamma \in \Gamma_T} \nu_\gamma)$ *is a finite disjoint union of basic semi-algebraic sets (respectively, basic quasi-algebraic sets, in the unordered case), where* Γ_T *is the* set of time-T halting paths *and* ν_γ *is the* initial path set.

2. *The* halting set *of* $\mathcal{M}$*:* Ω_M $(= \bigcup_{\gamma \in \Gamma_{M'}} \nu_\gamma)$ *is a countable disjoint union of basic semi-algebraic (respectively, basic quasi-algebraic) sets, where* $\Gamma_{M'}$ *is the* set of minimal halting paths.

3. *For* $\gamma \in \Gamma_M$ *(the* set of halting paths *of* $\mathcal{M}$*), the* input-output map Φ_M *restricted to* ν_γ – $\Phi_{M|\nu_\gamma}$ *is a polynomial map, or a rational map if* $\mathcal{R}$ *is a field.* □

Definition 5.4.2 The Mandelbrot Set [202], $\mathcal{M}$ *is the subset of the set of complex numbers* $\mathcal{C}$ *that remains bounded when subject to the following iterative procedure:*

$f_0(C) = C$, $f_{n+1}(C) = f_n(C)^2 + C$. *Formally, the complement* $\mathcal{M}'$ *of the Mandelbrot set is defined as*

$$\mathcal{M}' = \{C \in \mathcal{C} | f_n(C) \to \infty \text{ as } n \to \infty\}. \quad \square$$

It is to be noted that $f_i(C) \geq 2$ implies that eventually $f_n(C) \to \infty$.

In what follows, when we refer to the Mandelbrot set we mean the 2-dimensional set of real numbers corresponding to the Mandelbrot set, i.e., the set of pairs of the form $\langle C_r, C_i \rangle$ such that $C = C_r + iC_i$ is in the Mandelbrot set.

Remark 5.4.1 *Though defined using complex numbers, the Mandelbrot set corresponds to a 2-dimensional set of real numbers ($\subset \mathbb{R} \times \mathbb{R}$) as per the straightforward mapping* $f_n(c) \leftrightarrow (x_n, y_n)$, *where* $f_n(c) = x_n + \imath . y_n$ *and* $(x_{n+1}, y_{n+1}) = (x_n^2 - y_n^2 + x_n,\ 2x_n y_n + y_n)$.

Theorem 5.4.2 Undecidability Of The Mandelbrot Set: [66]. *The Mandelbrot set cannot be expressed as the countable union of semi-algebraic sets over* $\mathbb{R}$, *and hence not decidable over* $\mathbb{R}$. $\square$

5.4.2 General Undecidability Of Reachability

The undecidability result we will prove is based on the model of finite-dimensional machines over a field $\mathcal{R}$, which in our case will be $\mathbb{R}$, and on the undecidability of the Mandelbrot set over these machines. (We only introduce these "real" Turing Machines here, and refer the interested reader to [66].)

Reachability has been one of the major properties of hybrid systems which has been under investigation. System-state (or equivalently, "location") reachability is undecidable for hybrid automata with just two clocks [144], as the Turing machine halting-problem can be encoded as a reachability query. It becomes pertinent to ask if this undecidability result holds for the more powerful "real" computing machines of Blum et al.[66], where semi-algebraic sets appear naturally in the computability definition (see *Path Decomposition Theorem* [66]). We now prove that reachability for semi-algebraic hybrid systems is indeed undecidable even in this more general sense of computation. In the following construction, we present a semi-algebraic hybrid system and encode the Mandelbrot set as a reachability query. Since Blum and Smale have proved that the *Mandelbrot set is undecidable* [66], this proves that reachability over semi-algebraic hybrid systems is also undecidable, *even* under the "real" Turing Machine model.

We first provide the specifications of a hybrid system that we claim encodes the procedure for deciding if a complex number C belongs to the Mandelbrot set or not. We then prove this claim, also showing the corresponding reachability query.

Definition 5.4.3 The Mandelbrot Hybrid Automaton *Let* $C = \langle C_r, C_i \rangle$ *be a pair of real numbers. The* Mandelbrot Hybrid Automaton M_C *consists of*

- *One discrete state* s_0 *with invariant* False *and two continuous variables* Z_1 *and* Z_2.
- *Flow*$_1$: $\{\ Z_1' = Z_1 \ \wedge \ Z_2' = Z_2 \ \}$ *(no continuous evolution).*
- *One Discrete State Transition:* $1 \rightarrow 1$ *with Jump*$_1$: $(Z_1' = Z_1^2 - Z_2^2 + C_r) \wedge (Z_2' = 2Z_1Z_2 + C_i)$. □

Notice that in M_C the only possible trace is the infinite zeno path of self-loops. The Mandelbrot hybrid automaton is depicted in *Figure* 5.3

Theorem 5.4.3 General Undecidability Of Reachability *For semi-algebraic hybrid systems, reachability is undecidable even in Blum et al.'s "real" Turing machine formalism.*

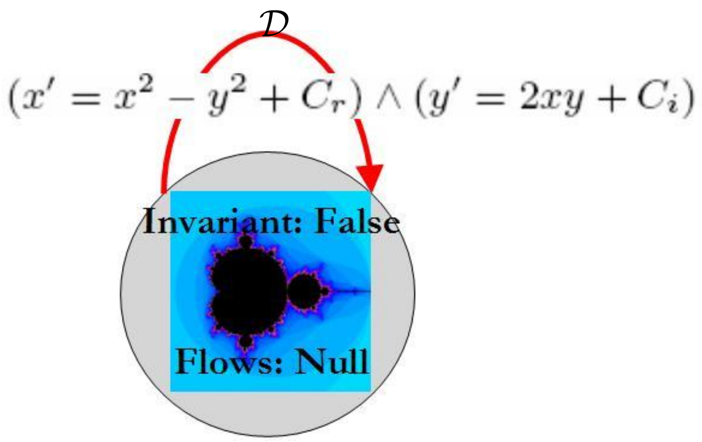

Figure 5.3: Mandelbrot Hybrid Automaton

Proof 5.4.1 *Let $S(t) = (Z_1(t), Z_2(t))$ be the point reached after t discrete transitions from the initial location $\langle s_0, (0,0)\rangle$ of the Mandelbrot hybrid automaton M_C defined above. After one more discrete transition (self-loop), we get*

$$\begin{aligned} S(t+1) &= S'(t) = \{Z_1(t)^2 - Z_2(t)^2 + C_r\} + \imath.\{2Z_1(t)Z_2(t) + C_i\} \\ &= \{Z_1(t) + \imath.Z_2(t)\}^2 + \{C_r + \imath.C_i\} \end{aligned}$$

In other words, if we consider the pairs of real numbers as complex numbers, we have $S'(t) = S^2(t) + C$ which is the defining equation of the Mandelbrot Set*. Clearly, there exists an evolution where $|S(t)| \geq 2$ if and only if $C = C_r + iC_i$ does not belong to the Mandelbrot set, i.e., the decidability of the reachability query $(Z_1^2 + Z_2^2 \geq 4)$ would imply the decidability of the Mandelbrot set, thus resulting in a contradiction.* □

5.5 Discussion

In the AAMC approach for the symbolic reachability problem: (1) The only approximation introduced is in the *Flow* expressions, which might have involved symbolic integration; (2) Only used existential quantified formulæare used; (3) The degree of the *Flow* polynomials and the degrees of the expressions in the hybrid automaton description influence the complexity of the first-order formulæ created and the number of steps needed to get sufficient precision. The Lagrange Remainder Theorem can be used to both under and over approximate the set of reachable points within the time interval $[0, \mathsf{end}]$. To place the results described here in the context of a large existing and continually growing literature, we mention a few related results (also see *Sec.* 6.4).

In [17] symbolic computation over $(\mathbb{R}, +, <, =)$ is used to compute preconditions on automata with linear flow conditions. Avoiding multiplication ensures good performance, but the class of automata on which the result can be applied is quite restricted, and of limited descriptive power.

In the d/dt tool (see [37]), a method involving several successive time steps is applied. Since the flow conditions (differential equations) are linear, the exact solution after a time step dt is used to compute the set of points that can be reached in that time. In another similar tool CheckMate (see [82]), a more sophisticated method involving time steps is introduced for the case of regions defined by polyhedra and solvable flow differential equations.

In a much closer related result of [278], predicate abstraction was introduced to map a hybrid automaton into a discrete one. The states of the discrete automaton represent sets of values which are indistinguishable with respect to a fixed set of predicates over the reals. Symbolic computation is used to determine the edges of the discrete automaton. In [125], the method was applied on piecewise linear hybrid automata to study the Delta-Notch signaling process.

In [10], predicate abstraction is combined with symbolic computations over the reals and with the use of time steps. The symbolic computation is used to determine the transitions between the abstract states, but the differential equations are kept linear so that the exact solutions are used in the symbolic computation. In particular, abstract states are forced to evolve at a given time step and symbolic computation is used to draw transitions by determining if intersections between (abstract) states are non empty. The main differences with respect to the AAMC methods are as follows: (1) We do not use predicate abstraction; (2) We can apply our method in the case of non-linear differential equations as well, through the use of Taylor polynomials or other symbolic integration schemes.

In a related computational analysis, Ghosh et al. [125] proposed a piecewise linear approximation to the continuous time model to generate a hybrid automaton. On this automaton, they conducted a symbolic reachability analysis using SAL - a heuristic symbolic decision procedure, to characterize the reachable region by numerical constraints, further sharpening the observations of Collier et al. [89]. We analyzed this system using the theory described in the next chapter. The preliminary results can be found in the chapter describing our tool Tolque.

SACoRe and IDA

During the development of this thesis, Casagrande et al. [74, 75] have developed alternative constructions of decidable subclasses that exploit Tarski's decidability result. In [74], they introduced the *Semi-Algebraic Constant Reset Hybrid Automata* (SACoRe), which extended O-minimal automata over the reals, in the case of flows obtained from non-autonomous systems of differential inclusions. SACoRe automata were shown to admit decision procedures for reachability and model checking for a limited fragment of CTL, by combining Tarski's decidability result over the reals and Michael's selection theorem. However, this formalism was found to be quite restrictive to the biochemical domain, as the constant reset requirement has no physical counterpart. This is because, when a biochemical system changes its "discrete" state, it is very unnatural for the concentrations to be reset to constant values. In fact, the most common result of a state change is no change, because of the continuous nature of chemical concentrations and other biochemical variables. In other words, identity resets are necessary to capture this fundamental aspect of most biological state transitions.

In [75], they introduced a new class of hybrid automata – *Independent Dynamics Hybrid Automata* (IDA), whose characterizing conditions are based upon a decidable first-order theory over the reals (e.g., $(\mathbb{R}, 0, 1, +, *, =, <)$). In particular, a hybrid automaton of dimension k can be defined only using formulæ over k dimensional vectors of reals. The dynamics are solutions of autonomous systems of differential equations. The reset conditions can be either constants as in the case of O-minimal hybrid automata [184] or the identity function. They distinguish *independent variables*, whose resets are the identity function, from *dependent variables* whose resets are constant functions. The flows and the reset functions of the dependent variables can depend on the independent ones, but not vice-versa.

Their motivation for defining this new subclass had two sources: (1) Extend O-minimal automata to make them suitable for Systems Biology applications; (2) Restrain semi-algebraic hybrid automata and make them more amenable to analysis. They exploit the decidability of the first-order theory over which an IDA is defined, both to bound the time interval they need to consider to solve a reachability problem, and to prove the decidability of reachability. The bounds on the time interval do not always exist on IDA, but they prove that they are always defined on an interesting subclass of IDA called ∞IDA. As a consequence, reachability is always decidable on

∞IDA. It is important to observe that they do not explicitly compute these time bounds, but check their existence, again, by solving a satisfiability problem.

Conclusions

The Algorithmic Algebraic Model Checking approach outlined here provides a general framework, but still lacks the needed degree of applicability, especially in the context of biological questions. We enumerate these issues: (1) Can one deal with unbounded time interval? (2) Can one deal with different and adaptively chosen time steps? This is particularly important if one is dealing with slow reactions as well as reactions that are relatively fast. (3) Can one conclude about the limiting situations when the time step sizes approach zero in the limit? (4) Is there a purely differential algebraic approach (e.g., Ritt algebra) for studying reachability? In the other directions, one can ask similar questions about how to extend these constructs for reachability to cases involving various modal operators (e.g., next). Beyond these questions, the other remaining problems are of algorithmic nature dealing with approximability, complexity and probabilistic computations. These are of enormous interest, if our approach is to be applicable for large biological systems that can be modeled modularly and hierarchically. Furthermore, to study decidability, we introduced the use of Blum et al.'s "real" Turing machine (or equivalently, finite-dimensional machine over a field) formalism [66] – a more apt approach to analyzing problems involving real computations. We found that reachability is undecidable even in this more powerful computational model. What are the extensions of this result? Many of these questions are addressed in the next chapter, where we show how this method can be extended to model check TCTL [7, 143]. Approximations are the topic of the chapter that follow the next one.

Chapter 6

Decidability of TCTL Model Checking

The real motivation for extending Fränzle's polynomial hybrid systems into semi-algebraic hybrid systems was to go beyond bounded reachability and to document the full power of quantifier elimination. This leads us to this chapter where we show that we can perform dense-time TCTL model-checking with parameters over semi-algebraic hybrid automata. Further, we can expand the repertoire of possible queries by allowing semi-algebraic relations connecting the current and the next state in the temporal logic queries. Once again, semi-decidability reduces to decidability in time-bounded non-zeno systems.

6.1 Introduction

The subject *Algorithmic Algebraic Model Checking* examines connections between systems biology, dynamical systems, modal logic and computability, and how they can be useful in the biological context. Towards this aim, we began by addressing the symbolic bounded reachability problem for a new class of hybrid models arising in systems biology – *semi-algebraic hybrid systems*, introduced in the previous chapter (based on the first and second papers in the "AAMC" series [239, 222]). There, we aimed to characterize the widest range of automata that admit sound albeit expensive mathematical techniques, as opposed to focusing on a very narrow class of systems that often prematurely sacrifice generalizability for the sake of efficiency. It was shown that the bounded reachability problem can be solved using real algebraic techniques like Taylor series approximation and quantifier elimination. It was expected that, building upon this algebraic bounded reachability algorithm [239] and other recent techniques (e.g. some of Fränzle's ideas [122]), we can address the algebraic model-checking problem over the dense time logic $TCTL$ [7]. The current chapter deals with this subject.

We build upon and integrate many existing ideas: we use Henzinger et al. [143]'s characterization of the *Until* operator as a fixpoint expression involving the *one-step until* operator. Exploiting the power of a symbolic approach, we retain all parameters as variables thus obtaining an algebraic expression representing the possible solutions to the temporal logic query. The ability to perform an entirely symbolic analysis of semi-algebraic hybrid systems (with polynomials of arbitrary degree) over a full temporal logic, limited only by computational power, distinguishes our approach from the other methods in literature. The appropriate frameworks for this setting consist of the following: semi-algebraic hybrid automata which allow polynomial expressions, TCTL logic to capture the continuous changes, and the "real" Turing machine model (discussed in *Sec.* 5.4.1) that computes a semi-algebraic operation in one unit step.

6.2 Background: TCTL

We now report the basic definitions[1] of the temporal logics TCTL and $T\mu$-Calculus which we use to study properties of our semi-algebraic hybrid automata.

Definition 6.2.1 TCTL[7] *It has the following syntactic structure:*

$$\phi ::= p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \exists\mathcal{U}\phi_2 \mid \phi_1 \forall\mathcal{U}\phi_2 \mid z.\phi.$$

Its associated semantics are described below:

- **z.**: *The freeze quantification "z." binds the associated variable z to the current time. Thus the formula $z.\phi(z)$ holds at time t iff $\phi(t)$ does.*
- $\phi_1\forall\mathcal{U}\phi_2$ *and* $\phi_1\ \exists\mathcal{U}\ \boldsymbol{\phi_2}$*: universal (on all paths) and existential (on at least one path) "until" operators. For $\phi_1\ \mathcal{U}\ \phi_2$ to be true on a path, ϕ_2 is required to be true somewhere along the path, and ϕ_1 is required to be true all along the path up to (but not necessarily at) that location.* $\square$

Remark 6.2.1 *The basic notations are often extended by the following syntactic abbreviations [7].*

1. $p\ \exists\mathcal{U}_{\leq max}\ q \ \equiv\ p\ \exists\mathcal{U}\ (q \wedge z.(z \leq max))$ *and* $p\ \forall\mathcal{U}_{\leq max}\ q \ \equiv\ p\ \forall\mathcal{U}\ (q \wedge z.(z \leq max))$*: "subscripted"* Until *operators (max is the time-bound).*
2. $\forall\mathcal{F} \ \equiv\ true\ \forall\mathcal{U}\ p$ *and* $\exists\mathcal{F} \ \equiv\ true\ \exists\mathcal{U}\ p$*: "eventuality" operators.*
3. $\forall\mathcal{G} \ \equiv\ \neg\exists\mathcal{F}\neg p$ *and* $\exists\mathcal{G} \ \equiv\ \neg\forall\mathcal{F}\neg p$*: "invariance" operators.*
4. $\forall\mathcal{X}$ *and* $\exists\mathcal{X}$*: universal and existential "next time" operators are typically* omitted *since they are not well-defined in dense-time systems.*

[1]The symbols used in this thesis are defined in *Table* .2 in *Sec. C* of the *Appendix*

Example 6.2.1 ***TCTL Queries***

- $p\ \forall\mathcal{U}\ q$ *asks whether on* every path *leading off the state where the modal formula is being considered, p is true everywhere until the state where q is true.*

- $\exists\mathcal{F}\ q$ *asks whether on* some path *leading off the state where the modal formula is being considered, there exists a state where q is true.*

- $\exists\mathcal{G}\ q$ *asks whether q will be true forever on* at least one path *leading off the state where the modal formula is being considered.*

Definition 6.2.2 Single-Step Until Operator, $\triangleright$, [143]. *The formula $p \triangleright q$ holds if $p \vee q$ is true all along "one step" of the hybrid system and q is true at the end of the transition.* $\Box$

Remark 6.2.2 *While in discrete time model-checking, the $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{U}$ operators are interpreted as fixpoints of the "next" state operator $\mathcal{X}$, the continuous-mode fixpoints rely on the "single-step until" operator $\triangleright$, which is the continuous-time next-state operator in this extended sense.*

Definition 6.2.3 $T\mu$-Calculus Syntax: [143]. *While μ-calculus works for discrete time systems, we have $T\mu$-calculus for capturing the continuous time properties of hybrid systems: $\phi ::= X \mid p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \triangleright \phi_2 \mid z.\phi \mid \mu X.\phi$, where μ is the* least-fixpoint *operator. Thus,*

- *The* greatest-fixpoint ν *can be expressed as* $\neg\mu X.(\neg\phi[X := \neg X])$.

- Existential Until: *operator can be expressed as the least fixpoint of a $T\mu$-calculus formula as* $p\ \exists\mathcal{U}\ q \;=\; \mu X.(q \vee (p \triangleright X))$.

- Universal Until: $p\forall\mathcal{U}q \;=\; \neg(\neg q\ \exists\mathcal{U}\ (\neg p \wedge \neg q))$ $\Box$

Notice that the translation of the universal until is valid only when q is "finitely variable" over all premodels [143].

Note 6.2.1 *Henzinger et al. [143] proved that the "possibility" operator* $\exists\mathcal{U}$ *is definable as a fixpoint over all premodels, while the "inevitability" operator* $\forall\mathcal{U}$ *is not definable as a fixpoint over all real-time systems. However in divergence-safe systems, testing for a property up to a certain finite fixed amount of time is sufficient to infer about its overall properties. Thus:*

$$s_1 \forall\mathcal{U} s_2 \;=\; s_2 \vee (s_1 \forall\mathcal{U}_{\leq c}(s_1 \forall\mathcal{U} s_2))$$

where,

$$s_1 \forall\mathcal{U}_{\leq c} s_2 \;=\; z.(s_1 \forall\mathcal{U}(s_2 \wedge (z \leq c))).$$

Combining this with the fact that $s_1 \forall\mathcal{U} s_2 \;=\; \neg((\neg s_2)\ \exists\mathcal{U}\ (\neg s_1 \wedge \neg s_2))$, *they derive the fixpoint expression for* $\forall\mathcal{U}$ *as:*

$$s_1 \forall\mathcal{U} s_2 \;=\; \mu X.(s_2 \vee \neg z.((\neg X)\exists U(\neg(s_1 \vee X) \vee z > c))), \text{ where } c > 0.$$

Definition 6.2.4 Least Fixpoint Computation *The standard algorithm for the evaluation of the least fixpoint* $\mu X.\phi$ *is given by*

1. $\psi :=$ false
2. repeat
   (a) $\phi := \psi$
   (b) $\psi := \phi[X := \phi]$
   (c) until $[\phi] = [\psi]$
3. return ϕ

6.3 Symbolic Algebraic Model Checking

In the last chapter, we proved the decidability of bounded model-checking and the undecidability of reachability in Blum et al.'s "real" Turing machine formalism. Our main results for semi-algebraic hybrid systems in this chapter may be summarized thus: (1) The "existential" segment of TCTL (including reachability) and the negation of the "universal" segment are *semi-decidable*. Further, all subscripted operators become *decidable* in the absence of zeno-paths; (2) Finally, a *quantifier elimination* tool (e.g. Qepcad [151],Redlog [109]) may be used to perform the fixpoint iterations of a TCTL query. The technical details are presented below.

To demonstrate our approach, we picked *Timed Computation Tree Logic (TCTL)* as it is a popular member of the class of dense-time temporal logics (see *Sec.* 2.2 for a list). It is to be noted that we neither restrict the temporal formulæ to be linear nor insist that the hybrid system be a timed automaton. While the non-symbolic approach to model-checking TCTL-specifications of real-time programs relies on the explicit construction of the region graph [8], we take the symbolic route.

The symbolic route to model-checking TCTL-specifications of hybrid systems is via the fixpoint expression for the *until* operator, which uses the standard *single-step until* operator $\triangleright$ [143] (also, see [239, 122]). The exact expression for the $\triangleright$ operator for semi-algebraic hybrid systems proves the basis of our approach: $\triangleright$ *corresponds to a semi-algebraic expression and is hence decidable.*

Since the $p \triangleright q$ operator is defined as $p \vee q$ holding all along one step of the hybrid system and q being true at the end of the one-step evolution, it may be written out for the the semi-algebraic hybrid system in terms of the continuous and discrete transition relations $\xrightarrow[\mathcal{C}]{t}$ and $\xrightarrow[\mathcal{D}]{0}$.

Definition 6.3.1 $\triangleright$ for Semi-Algebraic Hybrid Systems. *The expression* $p \triangleright q$ *is* True *at the current continuous state* R *if* q *is true now,* or

- *For one of the possible current discrete states* v*, there exists at least one discrete state* u *to which a transition can be taken such that* q *holds at the end,* or
- *For one of the possible current discrete states* v*, there exists a continuous transition (of at most* Δ *time units when we need to upper-bound the flow approximation error) all along which* $p \vee q$ *holds, with* q *being true at the end*[2]*.*

$$
\begin{aligned}
p \triangleright q \;=\; & q(R) \bigvee\nolimits_{\forall v} \\
& \Big(\{\exists S \bigvee\nolimits_{\forall u} \langle v, R\rangle \xrightarrow[\mathcal{D}]{0} \langle u, S\rangle \wedge q(S)\} \bigvee \\
& \{\exists S, h\ (0 < h \le \Delta) \wedge \langle v, R\rangle \xrightarrow[\mathcal{C}]{h} \langle v, S\rangle \wedge q(S) \wedge \\
& \forall S', h'\ ((0 \le h' < h) \wedge \langle v, R\rangle \xrightarrow[\mathcal{C}]{h'} \langle u, S'\rangle) \Rightarrow (p(S') \vee q(S'))\} \Big) \quad \square
\end{aligned}
$$

The one-step until operator of a hybrid automaton is visualized in *Figure* 6.1

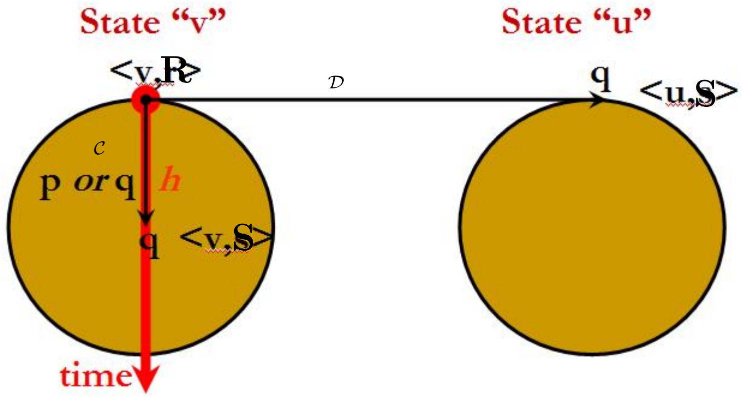

Figure 6.1: One-Step Until Operator

[2]The last term in the formula, $p(S') \vee q(S')$, can be replaced with just $p(S')$ for evaluating $\exists\mathcal{U}$ over semi-algebraic hybrid systems.

Remark 6.3.1 *The upperbound* Δ *on* h *should be* omitted *if there is no error in the* $Flow_v$ *expression. Also, since the discrete jump is instantaneous,* $p(R)$ *does not appear in the discrete-jump expression (second line).*

Theorem 6.3.1 *The* one-step-until *operator* $\triangleright$ *is decidable for semi-algebraic hybrid systems if* p *and* q *are also semi-algebraic.*

Proof 6.3.1 Semi-algebraic sets are closed under Boolean operations and quantifier elimination. Since *Jump*, *Inv* and *Flow* are semi-algebraic, so are the expressions $\xrightarrow[\mathcal{C}]{t}$ and $\xrightarrow[\mathcal{D}]{0}$. Thus $p \triangleright q$ is semi-algebraic since p and q are also semi-algebraic. Since quantifier elimination over semi-algebraic sets is decidable [276], $p \triangleright q$ is decidable. $\square$

Unlike Bounded Model Checking (BMC)[64], it is not possible to calculate the maximum bound on the number of future/past iterations that the formula should be checked in order to guarantee that the property holds. Such a *completeness threshold*[3] cannot be obtained for semi-algebraic hybrid systems as it would effectively imply decidability again. However, the well-established trick of incorporating the time variable into the temporal logic query can handle this, i.e., we ask if a property holds until we are no longer in the time-frame of interest, as opposed to asking if the property holds forever. In the bounded version of the model-checking task, we are only interested in the system evolution over a bounded *time horizon* or a bounded *number of steps.* Many practical problems are of this nature as is also evident from the burgeoning BMC field.

[3]For every finite state system M, a property p, and a given translation scheme, there exists a number CT , such that the absence of errors up to cycle CT proves that $M| = p$. CT is called the Completeness Threshold of M with respect to p and the translation scheme.

Corollary 6.3.1 *For semi-algebraic hybrid systems:*

1. $\exists\mathcal{U}$, $\exists\mathcal{F}$, $\exists\mathcal{G}$ *and their subscripted versions* $\exists\mathcal{U}_{\leq z}$, $\exists\mathcal{F}_{\leq z}$ *and* $\exists\mathcal{G}_{\leq z}$ *are* semi-decidable.

2. *The* negations *of* $\forall\mathcal{U}$, $\forall\mathcal{F}$, $\forall\mathcal{G}$ *and their subscripted versions* $\forall\mathcal{U}_{\leq z}$, $\forall\mathcal{F}_{\leq z}$ *and* $\forall\mathcal{G}_{\leq z}$ *are* semi-decidable.

3. *All subscripted operators become* decidable *in the absence of* zeno *paths.*

Proof 6.3.2 *The conclusions can be drawn as follows:*

- *The* $\exists\mathcal{U}$ *operator can be evaluated by iterating (indefinitely) over the decidable "one-step-until" operator* $\triangleright$ *as per the fixpoint characterization* $p\ \exists\mathcal{U}\ q \equiv \mu X.(q \vee (p \triangleright X))$. *Hence it is semi-decidable, i.e., the computation procedure is guaranteed to converge if the query is* True.

- *Since* $p\forall\mathcal{U}q \equiv \neg(\neg q\ \exists\mathcal{U}\ (\neg p \wedge \neg q))$, *it can be guaranteed to converge only when it is* False. *Thus the negation of* $\forall\mathcal{U}$ *is semi-decidable by our procedure.*

- *Since* $\exists\mathcal{F}p \equiv true\ \exists\mathcal{U}\ p$, *reachability is semi-decidable.*

- $\forall\mathcal{F}p \equiv true\ \forall\mathcal{U}\ p$ *and is not semi-decidable since* $\forall\mathcal{U}$ *is not.*

- *Since* $\exists\mathcal{G}p \equiv \neg\forall\mathcal{F}\neg p$, *we can guarantee that it will converge if it is* True *since* $\forall\mathcal{F}$ *is guaranteed to converge if it is* False. *Thus it is semi-decidable.*

- *Since* $\forall\mathcal{G}p \equiv \neg\exists\mathcal{F}\neg p$, *it is guaranteed to converge only when it is* False.

- *A new variable* time *is introduced, with* initial *value* 0, flow 1 *in all discrete states and identity resets. This allows the interpretation of* freeze *(z.X) and* subscripted until *($U_{\leq a}$) operators.*

- *Subscripts, being upper-bounds on the time-frame of interest, effectively bound the number of continuous steps while the absence of zeno-paths bounds the number of discrete jumps. In non-zeno systems, every path of a specified time-length can be explored fully. Hence all subscripted operators are decidable.* □

Remark 6.3.2 *Purely symbolic reachability cannot be convergent as many sets (including the Mandelbrot set) cannot be expressed as the finite union of semi-algebraic sets [66]. Similarly, the solution of many coupled, non-linear differential equations and simple discrete difference equations are inexpressible even using exponential and trigonometric terms [246], let alone as a finite union of polynomial inequalities. However, the conventional semi-decidability notion only applies to cases where the query can be answered as* True *or* False. *It was under this default assumption (also used by Fränzle while discussing "polynomial" hybrid systems [122]) that the above results were derived.*

Queries About Nature Of Change During A Time-Step

The repertoire of possible queries has naturally expanded, since polynomial terms can appear in the Boolean constructs part of the query. Temporal logic queries typically talk about the state of the system at a certain time in a certain path. However, with our representation we can allow the queries that connect the state of the system before (Z_-) and after (Z) a time-step as well, i.e., the nature of the change (and not just the changed state) can be queried. This does not affect the computational complexity substantially, as no new variable is introduced: Z and Z' were always part of the transition relation, and correspond to Z_- and Z in the temporal logic query. Sample queries include:

1. Is there a possible system evolution where y monotonously increases? $EG(y >$

y_-).

2. When y overshoots a threshold value y^{thr}, is it brought back under control within a time step? $AG(y_- > y^{thr}) \Rightarrow (y \leq y^{thr})$.

3. Is a spike possible in the evolution of y? $EF(y > 100y_-)$.

6.4 Discussion

Literature Review

Simulating and analyzing hybrid systems has become a field with tremendous import in various regimes of science and technology. However, no overwhelmingly superior mathematical approach has emerged, and several tools for addressing some aspects of restricted hybrid systems have been developed (see *Sec.* 9.2 for a survey of computational tools). In this section, we first review some of the restricted hybrid systems that are amenable to reachability analysis. We then peruse some techniques for approximate analysis. Having surveyed the two avenues of compromise, we discuss the developments in algebraic model checking and finally help put our semi-algebraic hybrid systems in perspective.

Simplifying The Problem

The most natural compromise to make is to see if less complex systems are amenable to analysis. Timed automata [12] , multirate automata [9] ,initialized rectangular automata [241, 144] , controllable linear systems [273] , some families of linear vector fields [186] and O-minimal hybrid automata [184] have been shown to be decidable for the reachability query. The problem easily becomes undecidable, for example in timed automata with linear expressions as guards or stop-watches, in multi-rate automata with clock-comparisons or without resets and in rectangular automata without the initialization requirement.

While semi-algebraic hybrid systems have been suggested in one form on another before [162, 21, 122, 186], the full potential of this formalization is only beginning to be appreciated [239]. Beyond timed, multirate and initialized rectangular automata [9, 241], the linearity of continuous dynamics is another extensively studied restriction [10, 37]. Controllable linear systems [273], some families of linear vector fields [186] and O-minimal hybrid automata [184] have also been shown to be decidable for the reachability query. In the case of O-minimal hybrid automata, the decidability is guaranteed by the decidability of the underlying theory and by the fact that the resets are constant. In semi-algebraic hybrid automata, we do not have any restriction on the resets. However, O-minimal systems admit more complex functions (beyond polynomials) in the flows, invariants and guards. As per the restricted *jump* condition the new continuous state cannot depend on the old state. Also, the system is assumed to be time-invariant so that the trajectories cannot self-intersect. These restrictions effectively decouple the discrete and continuous dynamics allowing bisimulations to be performed on each state separately. O-minimal theories which describe finite unions of points and open intervals, generalize the notion of semi-algebraic sets. They have nice topological properties, but not complete quantifier elimination. Instead they have a weaker property known as *model completeness.*

Simplifying The Analysis

In addition to making simplifying assumptions about the dynamics of the hybrid system, one could investigate incomplete approaches that work in many but not all cases, and approaches that operate not on the hybrid system but on its "simplified" abstraction. One such approach is over- or under-approximation, where the reachable region is assumed to have a (mathematically) convenient geometric shape such as a polyhedron, a level set or an ellipsoid [71, 82, 37, 180, 181, 183]. Some of these geometric entities are closed under some but not all operations like union and intersection.

Bisimulation on the other hand is an intelligent partitioning of the concrete system-state space of the hybrid system into fewer abstract discrete-states such that the properties of interest continue to hold in the simpler smaller model [138]. These "quotient" systems are typically obtained by refining an initial partition until it becomes compatible with the system dynamics and the property to be preserved. Thus it becomes sufficient to perform the model checking over this conservative abstraction of the continuous system. Predicate abstraction has also been frequently used to map a hybrid automaton into a discrete one [278, 10], with a good application-example being the analysis of the Delta-Notch signaling process [125].

Algebraic Model Checking

The translation of problems from the Boolean to the algebraic domain has been studied indirectly in control theory for a long time [25] and more thoroughly in the context of temporal logic for about a decade. While the algebraic model checking of *discrete systems* [43, 249] has also been explored, it is the more complex case of *hybrid dynamics* replete with open problems that is of more practical interest. Though the *reachability* problem is undecidable for general hybrid systems [144], algorithmic modeling and verification tools using quantifier elimination, SAT-solvers and Boolean methods for systems exhibiting simple continuous dynamics have been successfully developed in the last couple of years.

On the algebraic side, Jirstrand [162] demonstrated the use of Qepcad for problems of stationarizable sets, range of controllable output, following a curve and reachability in the context of non-linear control system design. Anai [21] and Fränzle [122] independently suggested the use of quantifier elimination for the verification of polynomial (semi-algebraic) hybrid systems. Anai and Weispfenning subsequently expounded the use of quantifier elimination for the reachability analysis of continuous systems with parametric inhomogeneous linear differential equations [22]. Fränzle went on to prove that progress, safety, state recurrence and reachability are semi-decidable using quantifier elimination of semi-algebraic formulæ[122] and develop proof engines for bounded model checking [123]. Lafferriere et al. [186] have described a quantifier-elimination-centric method for symbolic reachability computation of linear vector fields.

Ratschan and She [243] have recently suggested a new constraint propagation based abstraction refinement for the safety verification of hybrid systems. Carbonell and Tiwari [247] and Sankaranarayanan et al. [252] have devised schemes for generating invariants for hybrid systems. Other recent developments include Becker et al.'s integration of bounded model checking and inductive verification [57] and Lanotte and Schettini's new categorization - monotonic hybrid systems [188]. Lanotte and Tini [189] have recently proved that the semi-algebraic hybrid automaton obtained by approximating each formula in any (non-polynomial) hybrid system definition with its Taylor polynomial (of some degree k) is an over-approximation. From the perspective of *parametric analysis*, some of the hybrid system model-checkers can perform purely symbolic model checking to a small extent [16, 26, 27] and naturally, do not guarantee termination. In [292], a double exponential bound on the time complexity of parametric analysis of upper bound parametric timed automata is proved.

In Perspective

As the literature survey revealed, assumptions about the continuous and/or discrete dynamics have been central to the development of symbolic algebraic model checking techniques for hybrid systems. The "semi-algebraic" assumption is a constraint on both the continuous and discrete dynamics, essentially limiting the types of functions that can appear in the flow, guard and jump expressions.

While the approaches of Alur et al. [10] and Tiwari and Khanna [278] also have strong similarities to our approach, Lafferriere et al.[184]'s *O-minimal* system is the only model that allows more complex continuous dynamics.

However, the semi-algebraic scheme does not need time-invariance, linearity or any other assumption as it is ensured that we always remain in the semi-algebraic domain. This is because semi-algebraic sets are closed under union (*or*), intersection (*and*), complement (*not*), projection, and most importantly, quantifier elimination. Indeed, the geometric objects such as ellipsoids and grids are subclasses of semi-algebraic hybrid systems, where their specific shape simplifies numerical computation (i.e., less computationally complex than quantifier elimination over the reals) but disallows generalization and fully algebraic manipulation. The symbolic integration that we use effectively avoids explicit numerical integration, restricting our computation only to those aspects of the system relevant to the query.

Conclusion

In this chapter (based on [222]), the compromise we explored was bounded verification and model-checking, i.e., assuming there is a finite upper-bound on time beyond which we either do not care about the system or assume it works. Another course of compromise is *space abstraction*, where we depart from the continuous infinite space discretizing both space and time. Approximation with rectangular grids, ellipsoids and convex polyhedra is the key intelligence behind this approach. Alternatively, the polynomial inequalities and equations in the hybrid system specification and in the formula to be model-checked could be used to first identify connected components of the space relevant to the query. Rectangular grid abstraction could then be applied to each of these components which can be handled separately. The analysis of reachability results for perturbed and robust systems [122, 34, 182] is also being pursued. Techniques for converting non-polynomial specifications into polynomials [248, 196], such as the recent Taylor series method of Lanotte and Tini [189] could potentially increase applicability.

To summarize, the "semi-algebraic" method outlined here enables sophisticated symbolic algebraic model checking of a large class of hybrid automata, well beyond the capabilities of current applications of symbolic methods in this area. Based on the results of this chapter, we can focus on complexity improvement through meaningful approximations. The next chapter of the thesis (based on the third paper in the AAMC series [221]) focuses on approximate methods like bisimulation-partitioning, space discretization (using grids and polyhedra) and time discretization.

Chapter 7

Approximate Methods

We have established semi-algebraic hybrid automata as a broad subclass amenable to rigorous analysis. We now address the gargantuan computational complexity of the procedure. Since algebraists are working on improving the quantifier elimination procedure, we focus on the application side: How can we create smaller simpler problems by being willing to get an approximate answer? In this section, we extend four popular approaches, namely bisimulation partitioning, grid-based approximation, polyhedral approximation and time-discretization, to the semi-algebraic domain. We identify well-behaved subclasses and understand their biochemical relevance, if any. We provide new optimizations and improvements to the existing techniques, and discuss when and where these techniques might be applicable.

7.1 Introduction

A *semi-algebraic hybrid automaton* [239, 222] is a hybrid automaton, whose expressions corresponding to the initial values, state invariants, continuous flows, and the guards and resets of the discrete transitions are all semi-algebraic, i.e., Boolean combinations of polynomial equations and inequalities. They are often used to approximate more general systems, whose flow equations are not polynomial, since truncated Taylor series, polynomial splines and other symbolic integration schemes provide good semi-algebraic local approximations for flows, etc. A *location* of a semi-algebraic hybrid automaton H is a pair $\langle v, X \rangle$, where $v \in V$ is a state and $X \in \mathbb{R}^k$ is an assignment of values to the k system variables. The *transition relation* $\langle u, X \rangle \xrightarrow[T]{h} \langle v, X' \rangle$ of H connects all possible values of the system variables before and after *one step*; namely, it is either a discrete step $\langle u, X \rangle \xrightarrow[\mathcal{D}]{0} \langle v, X' \rangle$ for a time $h = 0$ or a continuous evolution $\langle u, X \rangle \xrightarrow[\mathcal{C}]{h} \langle v(= u), X' \rangle$ for a time period $h > 0$.

Earlier, in *Chapter* 5, we introduced this class and demonstrated the use of real algebraic methods for solving the bounded reachability problem. In *Chapter* 6, we examined the *single-step until* operator $p \triangleright q$ of the dense-time logic *Timed Computation Tree Logic* (TCTL) [7] , which is defined as $p \vee q$ holding all along "one step" of the hybrid system and q being true at the end of the transition. *Since quantifier elimination over semi-algebraic sets is decidable [276],* $p \triangleright q$ *was shown [222] to be decidable for semi-algebraic hybrid systems if* p *and* q *were also semi-algebraic.* It was further proved [222] that the "existential" segment of TCTL (including reachability) and the negation of the "universal" segment are semi-decidable over semi-algebraic hybrid automata. Further, all subscripted TCTL operators become decidable in the absence of zeno-paths.

Effectively, the symbolic algebraic model checking problem was reduced to a series of quantifier elimination problems which could be solved by a software tool such as Qepcad [151]. The only source of error (if any) arose in approximating non-polynomial systems. However, the computational complexity (double exponential) of the cylindrical algebraic decomposition severely limited the applicability of the method. In this chapter, we discuss the applicability of the different approximation approaches involving space-discretization to semi-algebraic hybrid automata. Approximate methods have been very successful in timed automata and linear hybrid systems, yielding efficient decidable algorithms in many cases [82, 127, 71, 37]. However, these methods rely on computational techniques that exploit the low dimensionality and other restrictions, of the dynamics of these subclasses of hybrid systems. In other words, the techniques are seldom applicable to more complex systems. Our first goal in this chapter is to show that many existing ideas can be made applicable to semi-algebraic hybrid systems, by using quantifier elimination in place of the original efficient-but-restrictive computational method. The second goal is to develop these ideas to obtain new optimizations and techniques. Further, we seek to identify well-behaved subclasses that are more general than timed or linear automata.

By suitably relaxing accuracy requirements, we aim to model-check the vast semi-algebraic class, without being severely computationally hindered. Quantifier elimination will still remain our engine of computation, though it will be used differently; namely, it will be invoked to solve many simple problems instead of a few complex problems. In this chapter, we will present the modified versions of existing techniques and understand their behavior over the semi-algebraic class. Clearly, different techniques will prove to be effective in different scenarios. In this chapter, we do not delve into this aspect, but instead focus on generalizing and optimizing existing techniques. In this sense, it is the first effort to catalogue the algorithms for approximate verification (reachability) for the general semi-algebraic class.

In this chapter, we develop new approximation methods applicable to the semi-algebraic class, based on the existing literature for much simpler subclasses of hybrid automata. For brevity, all proofs are provided in *Section B* of the *Appendix*.

7.2 Bisimulation Partitioning

The bisimulation idea is to convert the given hybrid automaton into a simpler one, which only preserves the properties of interest to us (in the query). The conventional bisimulation partitioning algorithm [127] involves splitting the discrete states based on the out-going discrete transitions. The source state of a transition is split so that, each new state (its partitions) has the minimal number of out-going transitions (ideally, each new partition will have only one possible successor discrete state). The rationale is that one expects only some of the guards (of the different out-going transitions) to be satisfiable, from different parts of the continuous space representing the discrete state (its state invariant).

For instance, Chutinan and Krogh [82] demonstrated how an approximate simulation, where only query-relevant states are partitioned, can be used to verify temporal properties of polyhedral-invariant hybrid automata. A more recent example is Ghosh and Tomlin's iterative refinement algorithm for computing symbolic discrete abstractions of the Delta-Notch hybrid automaton [127]. Their system, implemented using MATLAB and Qepcad, divides the state into several subsets with at most one exit transition each. Depending on the sign of the boundary polynomial inside the partition and the sign of its Lie derivative, the direction of the flow from the interior to the boundary is determined. The partitioning of the destination states is a new concept introduced in *Alg.* 4.5.3, in the analysis of 1-dimensional Piecewise Affine Maps (PAMs) [223].

7.2.1 Extended Bisimulation Partitioning

We first prove that these partitions are *computable* for semi-algebraic hybrid systems by expressing the task as a quantifier elimination problem.

Theorem 7.2.1 *For semi-algebraic hybrid automata, the standard bisimulation partitions are computable.* □

Having proved that the existing idea becomes applicable via quantifier elimination, we now suggest an improvement of the technique. This approach is founded on the observation that only a portion of the destination state may ever be accessed, after a specific discrete transition. Thus, by *splitting the destination state* as well, based on what fraction of it is accessible from the source state, we can refine the partitions. This simple extension was not necessary in linear systems, as the destination state space was typically entirely reachable from the reset region (after a discrete transition). Since semi-algebraic sets have innate complexity, it is very unlikely that continuous evolution from different reset regions will all envelope the entire state invariant. Clearly, since we chop off the region of the state invariant that is not reachable, the state invariants represent smaller sets. Hence, the extended-bisimulation-partitioning is likely to be a sharper one than the standard approach. The second advantage to this extended algorithm is that well-behaved subclasses can be characterized (see *Convergent Deterministic Automata* in *Sec.* 7.2.2). The complete series of computations are enumerated in *Algorithm* 2.

It is to be recalled that convergence of the partitioning does not imply decidability of reachability for general hybrid automata. As in the standard bisimulation case, the *over-approximated* set of points reachable from $\langle s_0, X_0 \rangle$ in the original hybrid system is given by the *union of the invariants* of all the states along all the trajectories starting at the state d_0 of the partitioned system corresponding to the partition of s_0 containing X_0. Even in the non-convergent case, this procedure yields an estimate of the reachable set if we roll-out H for a reasonable number of steps. Similarly, we can check if a specific X_f is reachable from a specific X_0. We iteratively partition until the

Algorithm 2 Extended Partitioning For Semi-Algebraic Automata

1. Pick a state s (source) with a discrete transition to state d (destination);

2. Split s into two states s_d and $s_{\bar{d}}$ thus: $Inv_{s_d}(X) \equiv \exists h, X' \ \langle s, X\rangle \xrightarrow[C]{h} \langle s, X'\rangle \wedge Guard_{s,d}(X')$ and $Inv_{s_{\bar{d}}}(X) \equiv Inv_s(X) \wedge \neg Inv_{s_d}(X)$;

3. Split d is into d_s and $d_{\bar{s}}$ thus: $Inv_{d_s}(X) \equiv \exists X'', X' \ Inv_s(X'') \wedge \langle s, X''\rangle \xrightarrow[D]{0} \langle d, X'\rangle \wedge \{\exists h \ \langle d, X'\rangle \xrightarrow[C]{h} \langle d, X\rangle\}$ and $Inv_{d_{\bar{s}}}(X) \equiv Inv_d(X) \wedge \neg Inv_{d_s}(X)$;

4. The states $s_d, s_{\bar{d}}, d_s, d_{\bar{s}}$ replace s and d. The transition from s to d is replaced by the one from s_d to d_s. All other transitions to (or from) s (d) are each replaced by two transitions to (or from) s_d and $s_{\bar{d}}$ (d_s and $d_{\bar{s}}$);

5. Repeat steps $(i) - (iv)$ until no transition from any state s to any state d can be found which splits s or d. $\square$

partition containing X_f is not in any trajectory starting from the partition containing X_0. We can then conclude guaranteed unreachability, or approximate reachability otherwise (counterexample-guided abstraction refinement, CEGAR [86]).

Example 7.2.1 *The approach is summarized in* Figure *7.1. In standard bisimulation, region X would have yielded 2 new partitions X_1 and X_2 corresponding to the regions which satisfy the guard for the transitions to Z and Y respectively. In the extended partitioning, $Z_i \cup Z_1^C$ and $Y_1 \cup Y_1^C$ are two* additional *states that are created because of the two transitions. Here, Z_1^C is the region reachable via continuous evolution from Z_1, which is itself the region reachable after a discrete reset from X_1; similarly for Y_1 and Y_1^C.*

Having generalized and extended an existing technique, we now characterize the broadest subclasses of hybrid systems where this new technique is well-behaved.

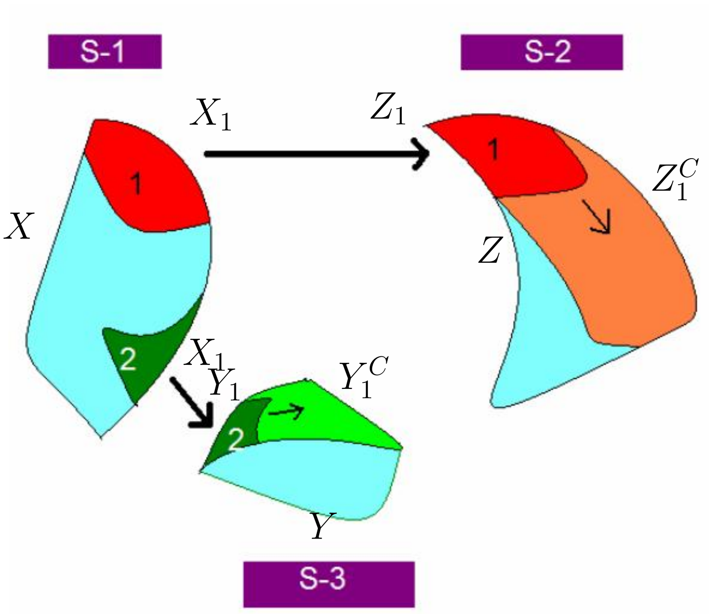

Figure 7.1: Standard Bisimulation Partitioning and its Extended Version

7.2.2 Convergent Deterministic Automata

In the most restrictive case of 1-dimensional PAMs, we proved[221] that affine 1-to-1 maps ($f(x) = a_i x + b_i,\ x \in I_i$) over 1-dimensional intervals guarantee finite cycle of successors (see *Lemma* 4.5.1). This section is an elaboration of that observation over semi-algebraic hybrid automata.

In *deterministic* hybrid automata, a discrete transition is taken the moment its guard is satisfied (with no two guards ever holding simultaneously). Hence there is a unique future trajectory for every initial system state. In this case, the source state s can be split such that being in partition s_i will guarantee that the guard corresponding to the transition to state d_i *will* be reached (and hence taken). If the *extended partitioning* procedure converges for a deterministic hybrid automaton, the original automaton will now correspond to a set of disconnected trajectories. Each of these will be a cycle of discrete states, with each state possibly preceded by a linear path of unique discrete states (all other topologies get excluded because there is no "future-branching" in deterministic automata). The extended partitioning *unlike the standard bisimulation partitioning*, produces exactly onto maps between successive states when convergent. We now show how many of their mathematical properties can be fruitfully exploited to address the reachability problem, for broad subclasses of convergent deterministic semi-algebraic hybrid automata.

In this section, we assimilate these ideas to prove decidability of linear maps in higher dimensions and bounded-error decidability of monotonic maps for hybrid automata that converge for our bisimulation partitioning procedure. Chaotic behavior because of discrete transitions is ruled out when we assume that the partitioning converges, while chaotic continuous dynamics are excluded because of the monotonicity requirement. In this section, we prove that for convergent systems:

1. If the flows and resets are monotonic, then the procedure can converge to an arbitrary degree of accuracy.

2. Further, if the resets and flows are linear, then a point has only a finite number of unique successors before it begins to cycle, unless the sets have infinite axes of symmetry (e.g., circle and sphere). Hence we can accurately conclude (un)reachability.

1-to-1 Linear Maps

Linearity is in general a friendly domain that admits many decidable algorithms and good bisimulations. In *linear convergent deterministic semi-algebraic automata*, all flows and reset maps are linear. Thus, infinite cycles are ruled out, since there are only a finite number of exactly onto maps possible (except when the sets have infinite axes of symmetry as does a circle). The types of linear maps and their geometrical correlations are elaborated below to explain this observation:

1. **Linear Shift** If the resets are of the form $x'_i = x_i + c_i$, S_1 and S_2 have to be identical but for a linear shift ($c_i > 0$ implies shifting right, $c_i < 0$ implies shifting left). Hence each point in S_1 can have only 1 possible post-image in S_2. Further, that point *has* to map back on to the source. Hence, if we have n such states forming a cycle, the length of the longest cycle of successors will also be n.

2. **Affine Shift** If the resets are of the form $x'_i = a_i x_i + b_i$, S_1 and S_2 have to be identical but for a linear shift and stretch. ($|a_i| > 1$ implies stretching, $a_i < 0$ implies flipping). Each coordinate x_i of a point in S_1 can have one such post-image in S_2 and a second "inverted" one if there is an axis of linear symmetry in the i-th dimension. Thus the total number of possible post-images is 2^{s_l} where s_l is the number of dimensions with an axis of linear symmetry. There are 2^{s_l} post-images possible in S_1 for each of these points. However, these *have* to overlap. If not, we can create a new map from S_1 to S_2 by following the sequence of back-and-forth maps $f^{new} = f_i(g_j(f_k(x_0)))$. In other words, the maximum length of the cycle is $2 \times 2^{s_l} = 2^{s_l+1}$.

   Example 7.2.2 *In a mapping between rectangles, each point has 4 possible post-images and 8 possible unique successors. Between cubes, each point has 8 possible post-images and 16 possible unique successors. The effect of repeated alternate application of two linear functions in one dimension that exactly span each others' domain is demonstrated in* Figure *7.2. The only two maps possible in each direction are $f_1(y), f_2(y)$ and $g_1(x), g_2(x)$: some subcycle* *of $x_0 \rightarrow y_1 \rightarrow x_1 \rightarrow y_0 \rightarrow x_0$ is inevitable.*

3. **Coupled Affine Shift** When the resets are of the form $x'_i = \Sigma a_j x_j$, rotations become possible in addition to shifting, stretching and flipping.

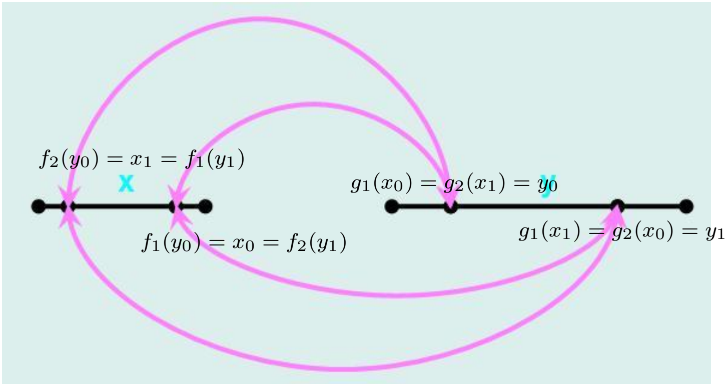

Figure 7.2: Typical result of cycling over two linear functions

These observations may be formalized thus:

Theorem 7.2.2 *There are only a finite number of 1-to-1 linear maps $f(X) = \Sigma AX + B,\ A_i, B_i \in \mathbb{R}^d$ possible, between two d-dimensional sets with finite axes of symmetry.* □

Corollary 7.2.1 *Given a cycle $S_1, \cdots, S_n, S_1$ of n d-dimensional sets with s_l axes of linear symmetry and s_r axes of rotational symmetry each, where each set maps exactly onto its successor ($S_{i+1} = f(S_i)$), the number of unique successors of any point is at most $ns_r2^{s_l}$.* □

Thus, we get:

Theorem 7.2.3 *Reachability over a deterministic semi-algebraic hybrid system with linear resets $Reset_{u,v}(X, X') \equiv (X' = \Sigma AX + B),\ A_i, B_i \in \mathbb{R}^d$ and linear flows*

$Flow_u(X, X') \equiv (X' = \Sigma AX + B),\ A_i, B_i \in \mathbb{R}^d$ is decidable, if the extended partitioning algorithm converges into states with finite axes of symmetry. $\square$

1-to-1 Monotonic Maps

The more general notion of monotonicity has recently been identified as a useful restriction in characterizing hybrid systems [188]. A function is said to be monotonic (with respect to its arguments), if it is always increasing or always decreasing or constant in the specified interval. For *monotonic convergent deterministic semi-algebraic automata*, monotonic flow and reset maps guarantee that the system has to eventually converge to a fixed point or a limit cycle (chaotic behavior can be ruled out). We now show that, unlike linearity which ensures decidability of reachability, monotonicity only guarantees that approximate reachability can be decided upto any specified accuracy:

Theorem 7.2.4 *If there exist 1-to-1 monotonic maps between two sets, all points converge to one of a finite number of fixed points or limit cycles.* $\square$

Thus, we get:

Theorem 7.2.5 *Reachability over a deterministic semi-algebraic hybrid system with resets and flows monotonic (with respect to all system variables), that converges upon extended partitioning, is decidable up to an arbitrary degree of accuracy.* $\square$

Example 7.2.3 *The effect of repeated alternate application of two monotonic functions $f(x)$ and $g(x)$ in one dimension that exactly span each others' domain is demonstrated in* Figure *7.3.*

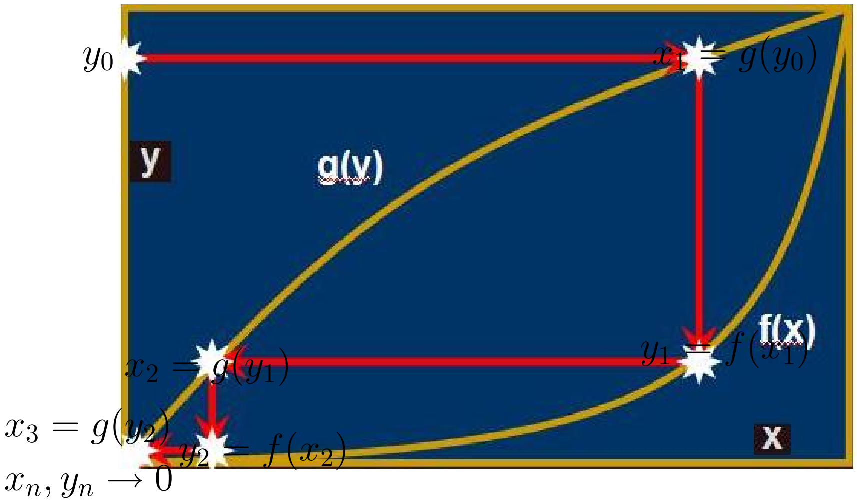

Figure 7.3: Typical result of cycling over two monotonic functions

7.3 Approximating as a Polytope

The bisimulation approach produced a new hybrid system more amenable to approximate temporal analysis. A completely different approach, very popular for the reachability problem, is to approximate from the first step. This involves assuming a mathematically convenient geometrical shape for the initial set—the simplest being a polytope (bounded polyhedron), which can be written as a Boolean combination of linear inequalities [82]. At each iteration, we compute the successor polyhedron by expanding it using the (one-step) transition relation of the hybrid system. Also, we need to ensure that the successor is also a polyhedron. At each iteration, the mathematics involves keeping track of the movement of the vertices and computing their new convex hull, or keeping track of the faces and moving them based on their maximum outward growth along the normal.

Clearly, a polyhedron can serve as a complexity restricting approximation of a semi-algebraic set as well. However, the conventional computational techniques are not applicable for two reasons. First, the *convex hull* of the successors of vertices of a polyhedron cannot be guaranteed to over-approximate the successor of the polyhedron. This is because, unlike linear systems, the flows cannot be assumed to be convexity preserving in semi-algebraic systems. Secondly, the *face-lifting* approach is not applicable in its basic from. This difficulty arises because, there is no straightforward way of calculating the maximum outward component of the flow along the normal to each face, of a polyhedron evolving with arbitrary polynomial dynamics.

In this section, we develop two new approaches that circumvent this problem.

7.3.1 Hyper-Rectangular Approximation

Instead of directly computing the approximated successor, we calculate the accurate complex successor (of the polyhedron), and *then* approximate it with a new polyhedron. Though the accurate successor computation slows us down, it is still better than the entirely exact computation. This is because the quantified semi-algebraic expression for the successor is relatively simple (polyhedron). We first describe a very coarse over-approximation which merely keeps track of the extremities along each dimension. This simple over-approximation can be obtained by calculating the maximum and minimum value along each dimension and bounding by one hyper rectangle. The complete series of computations are enumerated in *Algorithm* 3, where we denote the value of the i-th dimension of X by X_i.

Example 7.3.1 *The approach is summarized in* Figure *7.4. The dark blue region* $R(x, y)$ *is over-approximated by the light blue rectangle marked* 2 *defined using its extremities as* $(x^R_{min} \leq x \leq x^R_{max}) \wedge (y^R_{min} \leq y \leq y^R_{max})$*. Its accurate successor is region* $S(x, y)$*. Its extremities define its over-approximation* $(x^S_{min} \leq x \leq x^S_{max}) \wedge (y^S_{min} \leq y \leq y^S_{max})$*.*

Algorithm 3 Over-Approximating as One Hyper-Rectangle

1. Initialize the current over-approximation of the reachable set $\mathcal{R}$ with the starting hyper-rectangle $\bigwedge_i (i_{min} \leq X_i \leq i_{max})$;

2. Calculate the exact successor of $\mathcal{R}$ thus:
$\mathcal{R}_E(\langle s, X\rangle) \equiv\ \exists s', X', h\ R(\langle s, X'\rangle) \wedge \langle s', X'\rangle \xrightarrow[T]{h} \langle s, X\rangle$;

3. Calculate the maximum value of each dimension X_i in $\mathcal{R}_E$ thus:
$\{\exists s, X\ (X_i = i'_{max}) \wedge \mathcal{R}_E(\langle s, X\rangle)\} \bigwedge \{\forall s, X\ \mathcal{R}_E(\langle s, X\rangle) \Rightarrow X_i \leq i'_{max}\}$;

4. Calculate the minimum value of each dimension X_i in $\mathcal{R}_E$ thus:
$\{\exists s, X\ (X_i = i'_{min}) \wedge \mathcal{R}_E(\langle s, X\rangle)\} \bigwedge \{\forall s, X\ \mathcal{R}_E(\langle s, X\rangle) \Rightarrow X_i \geq i'_{min}\}$;

5. For each dimension, $i'_{min} \equiv min(i_{min}, i'_{min})$, $i'_{max} \equiv max(i_{max}, i'_{max})$;

6. If $j'_{max} \neq j_{max}$ or $j'_{min} \neq j_{min}$ for some dimension X_j, repeat the steps $(ii) - (v)$ with $\mathcal{R} \equiv\ \bigwedge_i (i'_{min} \leq X_i \leq i'_{max})$; else, the procedure has converged. $\square$

While the utility of such a gross over-approximation is questionable, it is nevertheless a technique one can resort to when the complexity of the problem is very high.

7.3.2 Hyper-Polygonal Approximation

If we want to approximate with a general polyhedron (more than just a hyper-rectangle), we have to resort to the convex-hull or face-lifting approaches. As arbitrary face-lifting is not known to be amenable to computational analysis, we suggest a convex-hull-based approach. Since the new positions of the vertices cannot capture the new convex-hull, we move them by the maximum possible increments and decrements in one step of the hybrid system. In other words, we compute the maximum

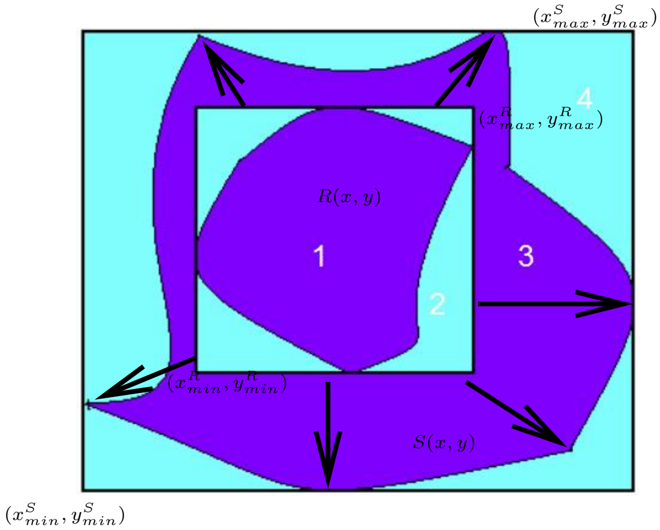

Figure 7.4: Over-Approximating as 1 Hyper-Rectangle

(and minimum) displacement (along each dimension) of any point in the polyhedron; and then assume that all the vertices could have moved by these amounts. The convex hull of the vertices, moved by these maximal amounts, is clearly guaranteed to be an over-approximation of the original polyhedron. The algorithm that results from this approach is detailed in *Algorithm* 4.

Example 7.3.2 *The approach is summarized in* Figure *7.5. Region* 1 *is over approximated using its maximal convex hull* $CH(0)$. *Its accurate successor* 3 *defines the maximum possible increase and decrease possible in each dimension, defining the red rectangle. By superimposing this rectangle at the 6 vertices of* $CH(0)$, *we get 24 new points. Their convex hull* $CH(1)$, *now with 9 vertices, is guaranteed to be an over approximation.*

Algorithm 4 Over-Approximating as One Hyper-Polygon

1. Initialize the current over-approximation of the reachable set $\mathcal{R}$ with the starting hyper-polygon, composed of the initial set of n vertices $v_1, \cdots, v_n$;

2. Calculate the exact successor of $\mathcal{R}$ thus:
$\mathcal{R}_E(\langle s, X\rangle) \equiv \exists s', X', h\ \mathcal{R}(\langle s', X'\rangle) \wedge \langle s', x'\rangle \xrightarrow[T]{h} \langle s, X\rangle$;

3. Calculate the maximum increment δ_{inc} in each dimension X_i thus:
$\{\exists s, X, s', X'\ \mathcal{R}(\langle s, X\rangle) \wedge \mathcal{R}_E(\langle s', X'\rangle) \wedge (X'_i - X_i = \delta_{inc})\} \bigwedge$
$\{\forall s, X, s', X'\ (\mathcal{R}(\langle s, X\rangle) \wedge \mathcal{R}_E(\langle s', X'\rangle)) \Rightarrow (X'_i - X_i \leq \delta_{inc})\}$;

4. Calculate the maximum decrement δ_{dec} in each dimension X_i thus:
$\{\exists s, X, s', X'\ \mathcal{R}(\langle s, X\rangle) \wedge \mathcal{R}_E(\langle s', X'\rangle) \wedge (X_i - X'_i = \delta_{dec})\} \bigwedge$
$\{\forall s, X, s', X'\ (\mathcal{R}(\langle s, X\rangle) \wedge \mathcal{R}_E(\langle s', X'\rangle)) \Rightarrow (X_i - X'_i \leq \delta_{dec})\}$;

5. Each vertex contributes 2^d new points, with each dimension being increased or decreased by the maximum amounts. R is assigned the convex hull of these $n2^d$ points;

6. Iterate $(ii) - (vii)$ until $\delta_{inc} = \delta_{dec} = 0$. $\square$

7.4 Rectangular Grid Abstraction

Instead of using one large polytope, the *grid abstraction* approach relies on keeping track of a number of small simple hyper-rectangles. Rectangular grids admit canonical representations and the number of faces grows linearly with the dimension, as opposed to convex polyhedra which become intractable in higher dimensions [71]. Two common simplifying strategies are restricting the vertices to be integers ("griddy") and the edges to be axis-parallel ("isothetic") [37].

Clearly, a griddy polyhedron that can be decomposed into few large rectangles can

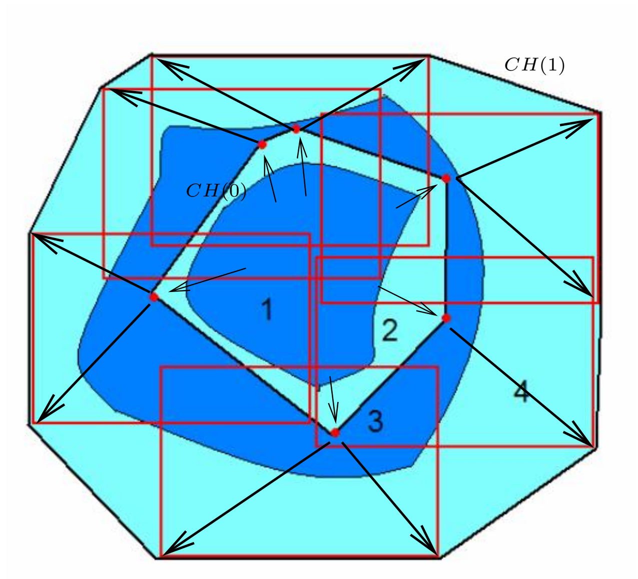

Figure 7.5: Over-Approximating as 1 Hyper-Polygon

be represented as a succinct isothetic polyhedron. Further, since edges can be arbitrary real numbers, we can push only as much as needed and not always by multiples of 1. Thus face lifting can cause a refinement of the grid. Both methods are not very space efficient. As pointed out by Bournez et al. [71], the representation of intermediate polyhedra (non-convex in general), identifying their faces, decomposing them into convex subsets, and performing face lifting as well as other set-theoretic operations is a complex computational problem. They discuss several canonical representation schemes for non-convex orthogonal polyhedra in any dimension. For piecewise-linear dynamical systems, the d/dt tool of Asarin et al. [37] integrates the above ideas by discretizing time and restricting the sets to be griddy polyhedra. The errors accumulated in the process of moving from the actual set to its bloated convex hull and from there to the griddy polyhedron, do not propagate to the next step as the original accurate set is also maintained. The calculated forward step is intersected with the guard to obtain the next state, similar to KRONOS [100, 303] (for timed automata) and HyTech [16, 142, 146] (for hybrid automata with constant derivatives).

Over vs Under Approximation The idea is to abstract the space into rectangular regions such that all points in a grid unit behave identically. This behavior can correspond to the best-case or worst-case and thus corresponds to an under or over approximation. A grid unit $[x_l, x_r]$ can reach the grid unit $[y_l, y_r]$ iff

- **Over-Approximation**:

$$\exists x \; x \in [x_l, x_r] \wedge \{\exists h, x' \; x \rightarrow_C^h x' \wedge x' \in [y_l, y_r]\}$$

- **Under-Approximation**:

$$\forall x \; x \in [x_l, x_r] \Rightarrow \{\exists h, x' \; x \rightarrow_C^h x' \wedge x' \in [y_l, y_r]\}$$

7.4.1 Union of "Griddy" Hyper-Cubes

We first show that the extension to semi-algebraic hybrid automata of the standard procedure is possible, because quantifier elimination can be used to compute the transitions between hyper-rectangles. One can partition the entire space into N^d hyper-rectangles, where N is the number of the $\mathcal{A}$-sized partitions [1] of each of the d dimensions. We use $B(X)$ to denote the k-dimensional grid unit $\bigwedge_i (B_i \leq X_i < (B_i + \mathcal{A}))$ of size $\mathcal{A}^d$. States will be connected to some of their $3^d - 1$ immediate neighbors, which differ by $+\mathcal{A}$,$-\mathcal{A}$ or 0 units in each dimension (with the identity-case alone excluded), and to some farther ones resulting from discrete resets. *Algorithm* 5 summarizes the series of computations necessary to calculate the reachable region starting from a specific grid unit.

[1] $\mathcal{A}$ should be fixed in relation to the error in the $\xrightarrow[C]{h}$.

Algorithm 5 Reachability Over Numerical Grids

1. Given one hyper-rectangle $F(X)$ corresponding to the source;

2. Initialize "frontier" set $\mathcal{F}$ with $\{F(X)\}$, and "reachable set" $\mathcal{R}$ with *null*;

3. For each new hyper-rectangle $P(X) \in \mathcal{F}$

   (a) Compile the set of neighbors:
   $\mathcal{N} \equiv \{Q(X)|(|Q_i - P_i| = \mathcal{A} \vee 0) \wedge \bigvee_i (|Q_i - P_i| \neq 0)\}$;

   (b) For each neighbor $Q(X)$ of $P(X)$ not already in the reachable set, test if it is reachable, i.e., $\exists X,\ P(X) \wedge \bigvee_{\forall v} \{\exists Y\ \bigvee_{\forall u} \langle v, X\rangle \xrightarrow[D]{0} \langle u, Y\rangle \wedge Q(Y)\}\ \bigvee \{\exists Y, h\ (0 < h \leq \mathcal{A}) \wedge \langle v, X\rangle \xrightarrow[C]{h} \langle v, Y\rangle \wedge Q(Y)$;

   (c) All candidate non-adjoint cells $Q(X)$ that can be reached by discrete state transitions can be tested thus:
   $\exists X,\ P(X) \wedge \bigvee_{\forall v} \{\exists Y\ \bigvee_{\forall u} \langle v, X\rangle \xrightarrow[D]{0} \langle u, Y\rangle \wedge Q(Y)\}$.

   (d) Add all reachable cells to both the reachable set $\mathcal{R}$ and the frontier set $\mathcal{F}$ and remove $P(X)$ from $\mathcal{F}$;

4. Iterate until there are no more new-hyper-rectangles. $\square$

Example 7.4.1 *The approach is summarized in* Figure *7.6. Region* 1 *is over approximated by* $RG(0)$*, the union of all the cells of the rectangular grid with at least one point in* 1*. Region* 3 *is the accurate successor of* $RG(0)$*, which is over-approximated as* $RG(1)$*.*

Algorithmic Enhancements Instead of estimating every transition possibility separately, one could use symbolic queries to get a general expression in a for all states of the form $[a, a+1)$ that allow a transition to $[a+1, a+2)$. Simplification of the

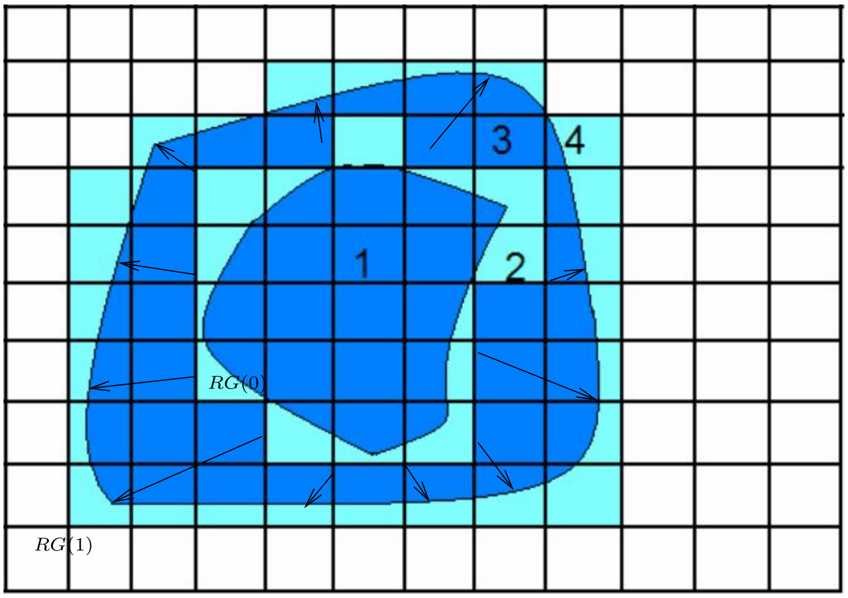

Figure 7.6: Over-Approximating using a Rectangular Grid

set of hyper-rectangles can lead to fewer bigger hyper-rectangles. While a quantifier elimination tool can do this, one needs to ensure the result continues to be a union of hyper-rectangles. We could over- (respectively, under-) approximate further by including extra (respectively, omitting some) hyper-rectangles to allow for simpler representation using fewer hyper-rectangles. The sides will now be of the form $n\mathcal{A},\ n = 1, 2, 3, \cdots$. $[p_l, p_r]$ is reachable $[p_l - a, p_r], [p_l - a, p_r - a]$ and $[p_l, p_r + a]$ are reachable, where a is a change in only 1 dimension. We can have similar rules for the other 3 quadrants of a square.

7.4.2 Union of "Isothetic" Hyper-Rectangles

Having shown that the standard procedure is applicable, we now develop a new approach for computing a sharper over-approximation (successor set of small hyper-rectangles) of the given hyper-rectangle. The idea is to compute the exact successor of a hyper-rectangle, and then over-approximate the region outside the initial hyper-rectangle (the "spill") by hyper-rectangles. In the previous case, we considered each of the $3^d - 1$ non-overlapping neighboring zones, and tested the transition to each. To simplify the expressions further, we suggest considering fewer overlapping neighbors; in particular, the zones with exactly one of the d dimensions increased or decreased i.e., $2d$ in all. To summarize, the standard method (previous case) accumulates the hyper-rectangles reachable from the given hyper-rectangle by testing transition to each of the $3^d - 1$ non-overlapping neighbors. The size of each neighbor is fixed ("griddy") forcing the approximation error to be at least that big. In the new technique, the hyper-rectangles continue to be axis-parallel ("isothetic"), but their vertices are not fixed. As a result, the approximation is guaranteed to be much better than the "griddy" case. The additional trick of considering fewer overlapping rectangles cannot be applied to the standard method, as the approximation will become too coarse.

Algorithm 6 presents the details of the method. We estimate the spill in each neighboring zone by calculating the extremities in that zone, along the lines of the scheme for over-approximating the entire set as a single-hyper-rectangle. We use $B(X)$ to denote the k-dimensional grid unit $\bigwedge_i (B_i^l \leq X_i < B_i^r)$ (side of the hyper-rectangles are no longer fixed at $\mathcal{A}$). Further, $B_{\neg j,k}(X)$ denotes $\bigwedge_{i \neq j \vee k} (B_i^l \leq X_i < B_i^r)$.

Example 7.4.2 *The approach is summarized in* Figure 7.7. *The rectangle* R_0^0 *is the initial over-approximation of region* 1. *Four new overlapping rectangles* R_1^0, R_1^1, R_1^2 *and* R_1^3 *are added to over-approximate the "spill" outside region* R_0^0. *The procedure then iterates over each of them in the general case.*

Algorithm 6 Approximating with Many Hyper-Rectangles

1. As before, maintain the set of reachable hyper-rectangles $\mathcal{R}$ and the set of new hyper-rectangles $\mathcal{F}$ just added to the reachable set, representing the expanding frontier;

2. For each $P(X) \in \mathcal{F}$, compute the exact successor set of $\mathcal{R}$ thus:
$\mathcal{R}_E \equiv \bigvee_{\forall v}\{\exists Y \bigvee_{\forall u}\langle v, X\rangle \xrightarrow[D]{0} \langle u, Y\rangle \wedge P(Y) \bigvee\{\exists Y, h\ (0 < h \leq \mathcal{A}) \wedge \langle v, X\rangle \xrightarrow[C]{h} \langle v, Y\rangle \wedge P(Y);$

3. For each dimension X_i:

   (a) For the neighbor $Q(X)$ where $Q_i^l = N_i^r$, calculate Q_i^r: $\{\exists X\ (P_{\neg i}(X) \wedge X_i = Q_i^r) \wedge \mathcal{R}_E(X)\} \bigwedge \{\forall X\ (P_{\neg i}(X) \wedge X_i > Q_i^r) \Rightarrow \neg\mathcal{R}_E(X)\}$. If $Q_i^r < P_i^r$, skip the next two steps;

   (b) We now need to calculate the extremities l_j^{i+}, r_j^{i+} in each of the other dimensions X_j where $j \neq i$: $\{\exists X\ (P_{\neg i,j}(X) \wedge X_i > P_i^r \wedge X_i < Q_i^r \wedge X_j = l_j^{i+}) \wedge \mathcal{R}_E(X)\} \bigwedge \{\forall X\ (P_{\neg i,j}(X) \wedge X_i > P_i^r \wedge X_i < Q_i^r \wedge X_j < l_j^{i+}) \Rightarrow \neg\mathcal{R}_E(x)\}$ and $\{\exists X\ (P_{\neg i,j}(X) \wedge X_i > P_i^r \wedge X_i < Q_i^r \wedge X_j = r_j^{i+}) \wedge \mathcal{R}_E(X)\} \bigwedge \forall X\ (P_{\neg i,j}(X) \wedge X_i > P_i^r \wedge X_i < Q_i^r \wedge X_j > r_j^{i+}) \Rightarrow \neg\mathcal{R}_E(X)$.

   (c) The hyper-rectangle defined by $Q_i^l < X_i < Q_i^r \bigwedge_{j \neq i} l_j^{i+} < X_j < r_j^{i+}$ is added to the list of new hypercubes and also to the reachable set R;

   (d) Repeat the above three steps for the neighbor where $Q_i^r = P_i^l$ and $Q_i^l, l_j^{i-}(< X_j), (X_j <) r_j^{i-}$ need to be calculated;

4. Repeat $(ii) - (iii)$ until the procedure converges. $\square$

7.4.3 Battleship Strategy

In the "griddy" case, we inspect every possible neighbor and test transition. Alternately, we could have computed the exact successor of the entire set, and then

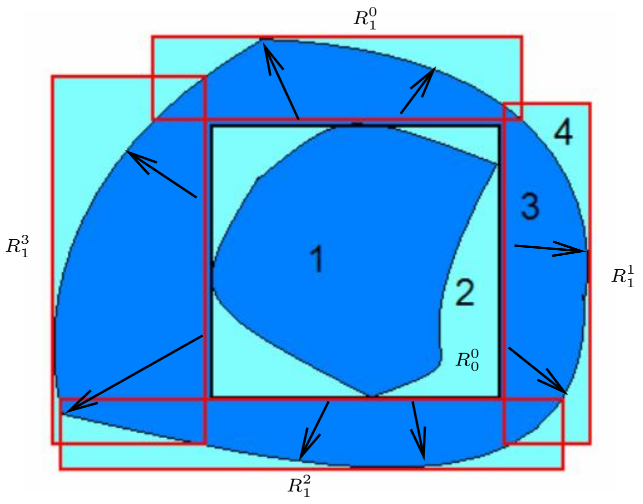

Figure 7.7: Over-Approximating using many Hyper-Rectangles

extracted the component hyper-rectangles. Such an approach would require a procedure for converting a semi-algebraic set (the exact successor) into an over- (or under-) approximating union of hyper-rectangles of fixed dimension.

In the "isothetic" case, we over approximated the "spill" outside the hyper-rectangle with a hyper-rectangle in each neighboring zone (with substantial overlap). Alternatively, we could compute the best non-overlapping but non-griddy hyper-rectangles that cover the newly reachable points, without having to compute the maximum and minimum values of each dimension in each neighboring zone. This approach again requires a general procedure for converting the exact successor into a union of hyper-rectangles of arbitrary dimension.

We solve this problem by actually testing if potential vertices (from a griddy or isothetic grid) are included in the exact reachable set (just as one uses the pictorial grid of hits and misses to guess where the opponent's fleet is located in the game "Battleship")[2]. We then use the resulting set of present and absent points to pick

[2]In this chapter, we will not go into the heuristics for picking the sample points that can be

candidate hyper-rectangles. Quantifier elimination is still necessary, since we may wish to guarantee that the hyper-rectangles we have picked are wholly inside (under-approximation) or that the hyper-rectangles we have omitted are wholly outside (over-approximation). Hence the approach we have suggested addresses the problem of minimizing the number of quantifier-elimination queries. We now provide the details of this new technique, which can be used in conjunction with both the algorithms presented before.

Our approach, detailed in *Algorithm* 7, follows from the following simple observations:

1. A given numerical point can be trivially tested for inclusion in a semi-algebraic set: just substitute in the expression and see if it simplifies to *True*.

2. By choosing various grid points and testing for set membership, we can get a rough idea of the shape of the real set.

In the under-approximation case, hyper-rectangles with at least one vertex not in R can be safely omitted. The hyper-rectangles with all vertices in R are the contenders for quantifier elimination. In both cases, one could use a "proof-by-example" approach, where one verifies the feasibility at some randomly selected points (center being the first choice) to see if quantifier elimination can be avoided. By randomizing or biasing the grid points, one can obtain non-griddy vertices. If, in addition, high-dimensional convex hull algorithms are used, one could build upon this method to derive general polyhedral representations as well.

Example 7.4.3 *The approach is summarized in* Figure *7.8. Every vertex of the rectangular grid is numerically evaluated for its presence in the dark blue set. The red*

developed given the hybrid system specification.

Algorithm 7 Battleship Strategy

1. Calculate i_{max} and i_{min}, the maximum and minimum values of X_i in the given set $\mathcal{R}$: $\{\exists X\ (X_i = i_{max}) \wedge \mathcal{R}(X)\} \bigwedge \{\forall X\ \mathcal{R}(X) \Rightarrow X_i \leq i_{max}\}$ and $\{\exists X\ (X_i = i_{min}) \wedge R(X)\} \bigwedge \{\forall X\ R(X) \Rightarrow X_i \geq i_{min}\}$;
2. Split each dimension into equidistant points of the desired resolution;
3. Evaluate membership in R for each grid point g by substitution: $\mathcal{R}(g)$;
4. The small hyper-rectangles created by the grid points which contain at least one vertex in $\mathcal{R}$ are immediately included in the over-approximation;
5. Hyper-rectangles where none of the vertices are in $\mathcal{R}$ are included only if $\exists x \in G\ \mathcal{R}(x)$ returns *true*. $\Box$

points were found to be "inside" the set. The points could then be used to guess an under / over approximate union of cells, or to suggest a polygonal approximation. In order to guarantee that the approximation suggested is strictly over or under, quantifier elimination will be necessary.

7.5 Time Discretization

Time discretization can be employed (in conjunction with most techniques) to approximate the hybrid system dynamics. Conventionally, the most restricted transition relation enforces continuous evolution for a fixed time-step Δ followed by one optional discrete transition. The typical "improvement" over the previous case could be allowing the discrete jump anywhere during the continuous evolution, as opposed to only at the end of it. This model could be made even more realistic by allowing N jumps

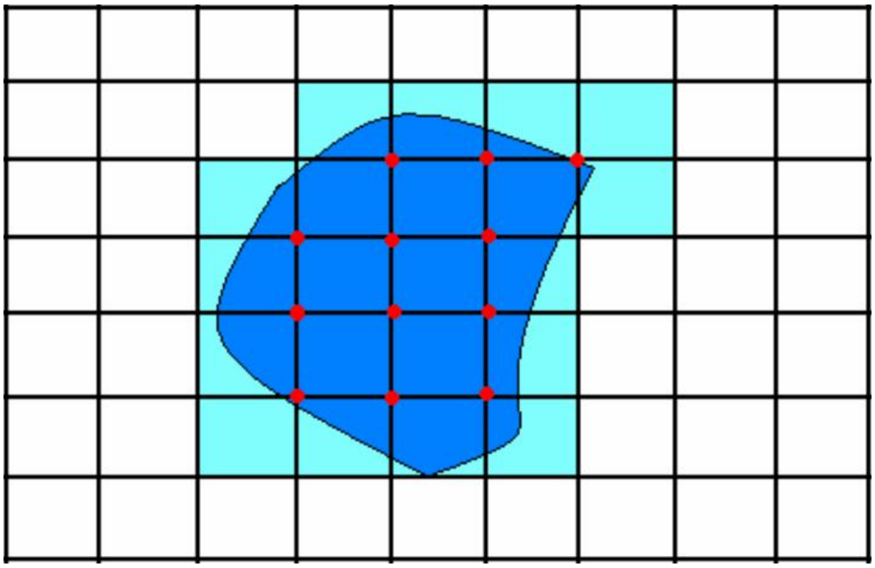

Figure 7.8: Battleship strategy for identifying candidate vertices

anywhere during the continuous evolution. Clearly, the only paths that get excluded here are those that involve more than N jumps in Δ time. All the restrictions described above are "fixed step", i.e., the system progresses in time-steps of Δ. Each of them could be relaxed by allowing the time-step to be in the range $[0, \Delta]$ to capture many other behaviors. Such restrictive transition relations greatly simplify fixpoint evaluations of temporal logic operators.

A completely different time-discretization-based under-approximating approach would be to ignore the behavior of the system *during* the continuous evolution. We simply use the end-points to verify the temporal query. Another simplifying over-approximation would be to assume that the state invariant needs to be true only at the beginning of (and not all along) the Δ time units of continuous evolution. This heuristic could prove particularly useful if we combine time discretization with the partitioning algorithm discussed earlier (which will accumulate complex state-invariants).

We present the simplifications that result from applying these concepts in this section.

7.5.1 Background: CTL

We now report the basic definitions[3] discrete-time temporal logic CTL and μ-Calculus, which is used to express the fix-point characterizations.

Definition 7.5.1 CTL[72] *It has the following syntactic structure:*

$$\phi ::= p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \exists\mathcal{X}\phi_2 \mid \phi_1 \forall\mathcal{X}\phi_2 \mid \phi_1 \exists\mathcal{U}\phi_2 \mid \phi_1 \forall\mathcal{U}\phi_2.$$

Its associated semantics are described below:

- $\forall\mathcal{X}\phi$ *and* $\exists\mathcal{X}\phi$*: universal and existential "next time" operators describe what must be true at the next time step of the system.*
- $\phi_1\forall\mathcal{U}\phi_2$ *and* $\phi_1\ \exists\mathcal{U}\ \phi_{\mathbf{2}}$*: universal (on all paths) and existential (on at least one path) "until" operators. For* $\phi_1\ \mathcal{U}\ \phi_2$ *to be true on a path,* ϕ_2 *is required to be true somewhere along the path, and* ϕ_1 *is required to be true all along the path up to (but not necessarily at) that location.* $\square$

Remark 7.5.1 *The basic notations are often extended by the following syntactic abbreviations [72].*

1. $\forall\mathcal{F}\ p \ \equiv\ true\ \forall\mathcal{U}\ p$ *and* $\exists\mathcal{F}\ \equiv\ true\ \exists\mathcal{U}\ p$*: "eventuality" operators.*
2. $\forall\mathcal{G}\ p \ \equiv\ \neg\exists\mathcal{F}\neg p$ *and* $\exists\mathcal{G}\ \equiv\ \neg\forall\mathcal{F}\neg p$*: "invariance" operators.*

Example 7.5.1 ***CTL Queries***

[3] *Table* .2 in *Sec. C* of the *Appendix* lists symbols used in the thesis

- $p \; \forall\mathcal{U} \; q$ *asks whether on* every path *leading off the state where the modal formula is being considered, p is true at all time steps until the state where q is true.*

- $\exists\mathcal{F} \; q$ *asks whether on* some path *leading off the state where the modal formula is being considered, there exists a time step when q is true.*

- $\exists\mathcal{G} \; q$ *asks whether q will be true at all time steps on* at least one path *leading off the state where the modal formula is being considered.*

Recall that unlike TCTL, no special "one-step until" operator needs to be defined. This is because the notion of a "next state" is well-defined in the discrete-time interpretation, and hence the $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{U}$ operators can be interpreted as fixpoints of the "next" state operator $\mathcal{X}$.

Definition 7.5.2 μ-Calculus Syntax *μ-calculus captures the discrete time properties of dynamical systems:* $\phi ::= X \mid p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \exists\mathcal{X}\phi \mid \forall\mathcal{X}\phi \mid \mu X.\phi$ *, where μ is the* least-fixpoint *operator. Thus,*

- *The* greatest-fixpoint ν *can be expressed as* $\neg\mu X.(\neg\phi[X := \neg X])$.

- Existential Until: *operator can be expressed as the least fixpoint of a $T\mu$-calculus formulæ as* $p \; \exists\mathcal{U} \; q \;=\; \mu Z.(q \vee (p \wedge \exists\mathcal{X} Z))$.

- Universal Until: $p\forall\mathcal{U}q \;=\; \neg(\neg q \; \exists\mathcal{U} \; (\neg p \wedge \neg q))$ □

7.5.2 Discrete-Time Model-Checking

We wish to use CTL for model-checking semi-algebraic hybrid automata. Since CTL model-checking involves iterations of the next-state operator, we must first understand its relation to the transition relation. We then document the different assumptions that can be made to simplify the transition relation. Another motivation for discretizing the transition relation is to be able to guarantee progress of time with the iterative procedure. This problem is encountered while model-checking dynamical systems that have zeno executions which can be ignored.

Recalling Continuous Mode Definitions The *transition relation* $\mathcal{T}$ of a hybrid system H connects the possible values of the system variables before and after *one step* - a discrete step for a time $h = 0$ or a continuous evolution for any time period $h > 0$:

$$\mathcal{T}(\ell \xrightarrow{h} \ell') = \ \{h = 0 \wedge \ell \underset{\mathcal{D}}{\xrightarrow{0}} \ell'\} \ \vee \ \{h > 0 \wedge \ell \underset{\mathcal{C}}{\xrightarrow{h}} \ell'\},$$

where the continuous reachability transition relation $\langle v, R\rangle \underset{\mathcal{C}}{\xrightarrow{h}} \langle v, S\rangle$ iff $\Big(\mathit{Flow}_v(R, S, t_0, h) \ \wedge \ \forall Z', h' \in [0, h) \ \mathit{Flow}_v(R, Z', t_0, h') \Rightarrow \mathit{Inv}_v(Z') \Big)$,

and the *discrete reachability transition relation* $\langle v, R\rangle \underset{\mathcal{D}}{\xrightarrow{0}} \langle u, S\rangle$ iff $\langle v, u\rangle \in E \ \wedge \ \mathit{Jump}_{v,u}(R, S)$.

Using CTL for Dense Time Systems The crux of TCTL model-checking was expressing the *one-step until* operator as a quantified semi-algebraic set, thus proving its decidability via real quantifier elimination. Similarly, we now have to show how the *next state* operator can be expressed in a decidable form. The next state operator is directly expressible in terms of the transition relation of the system as:

Definition 7.5.3 ***Next Operator*** $\mathcal{X}^D$

$$\mathbf{A}\mathcal{X}^{\mathbf{D}} \ \mathcal{P}(\ell) = \forall \ell' \ \{\mathcal{T}(\ell \xrightarrow{h} \ell') \Rightarrow \mathcal{P}(\ell')\} \tag{7.1}$$

$$\mathbf{E}\mathcal{X}^{\mathbf{D}} \ \mathcal{P}(\ell) = \exists \ell' \ \{\mathcal{T}(\ell \xrightarrow{h} \ell') \ \& \ \mathcal{P}(\ell')\} \quad \square \tag{7.2}$$

Note 7.5.1 *The notion of the next-state being h time later is not innate to the original CTL definition, but is a property of the transition relation. We have indicated it for clarity, and to reiterate the fact that we are interpreting a continuous system in discrete time.*

Discrete-time CTL model-checking can thus be performed algebraically on semi-algebraic hybrid systems, using an appropriate h (time-step) value. h could also be treated as a symbolic variable, thus yielding a symbolic expression in h representing possible evolutions and solutions to the temporal query. [4]

The Accurate Model With Variable Time-Step

The discrete-time transition relation that corresponds to the continuous time transition relation allows the system to take steps less than h in cases where a discrete state transition is being made. In other words, the system dynamics after the first jump is captured only in the "next" time-step. The system is allowed to evolve continuously for the full-time h in the current state, or to evolve to an intermediate point ($\leq h$) and then take a transition. Thus the effective length of each time step h varies. The 2 possible evolutions are:

- 1. *Evolve continuously in the current discrete state s for the entire h time units,*

  OR

- 1. *Evolve continuously in the current discrete state s for h_1 time units, where* $0 \leq h_1 \leq h$

[4] By specifying the nesting degree / time-bound of the iteration, unusual queries like "Is there a value of time-step h, $0.25 < h < 0.5$, for which a certain property holds for 5 steps?" could be answered.

2. *2. Then make a discrete state transition to state s'*

The semi-algebraic expression capturing this model can be expressed thus:

$$\mathcal{T}(\ell \xrightarrow{h} \ell') = \ \{\exists \ell'', h_1 \ 0 < h_1 < h \wedge \ell'' \xrightarrow[\mathcal{C}]{h_1} \ell' \wedge \ell \xrightarrow[\mathcal{D}]{0} \ell''\} \bigvee \{\ell \xrightarrow[\mathcal{C}]{h} \ell'\}.$$

The total number of quantified variables in the *accurate model* above is n_D+1, where n_D variables represent the state of the system just before the discrete transition(ℓ''), and h_1 stands for the time spent before the transition. This "continuous" model cannot guarantee the time for which the model has been verified, as the number of iterations does not correlate with the time elapsed. This is because many of the iterations could have corresponded to just a discrete transition, and many of the continuous steps could have been for time-lengths less than h. Further, *zeno-paths* are not excluded in the continuous-time formulation. A natural refinement of this transition relation is imposing a minimum on the time spent in each step. In other words, we can assume that if the hybrid system makes a discrete transition to a state, it spends at least h_{min} time in the new discrete state. This can guarantee that n iterations of the model-checking process would have covered at least nh_{min} amount of time.

For computational simplicity (in terms of number of quantified variables) and for ensuring time-progress, we are forced to explore transition relations which bound the number of discrete state changes inside one step of integration. We refer to such systems as "restricted discretized hybrid systems". In this section, we suggest three "discrete" interpretations of the continuous transition relation for semi-algebraic hybrid automata. We start with the *Timed-Jump-Model* that allows only one state transition at the beginning of the time-step, thus using one less quantified variable (no h_1 to represent the intermediate time). A simple extension - the *One-Jump-Anywhere-Model* allows the transition to occur anywhere during the h time, at the cost of reintroducing the quantified variable h_1 ($\exists h_1, 0 < h_1 < h, \cdots$). The advantage however is that the verification always progresses by h time units after each iteration. The *k-Jumps-Model* allows at most k transitions anywhere during the h time units. This definition is provided for theoretical completeness. The total number of quantified variables in this case is $2 * k * n_D + k$, as the k jumps involve k system states after continuous evolution, k system states after resets and k time-lengths for which they evolved continuously. All three models have the advantage that n iterations of the transition relation guarantee that the system has been model-checked for exactly time nh.

Model 1: Timed-Jump Model

The assumption that the hybrid system can make only **one** discrete state transition in the time step h is made. Further, this privilege to jump is provided only at the beginning of the time-step. Thus:

- 1. *Evolve continuously in the current discrete state s for the entire h time units*

  OR

- 1. Make a discrete state transition to state s'

  2. Then evolve in the new discrete state s' for h time units

The semi-algebraic expression capturing this model can be expressed thus:

$$\mathcal{T}(\ell \xrightarrow{h} \ell') = \quad \{\exists \ell'' \; \ell \xrightarrow[\mathcal{D}]{0} \ell'' \wedge \ell'' \xrightarrow[\mathcal{C}]{h} \ell'\} \bigvee \{\ell \xrightarrow[\mathcal{C}]{h} \ell'\}.$$

Limitations Equalities in the guard condition may lead to them being never satisfied as they are checked only at the start/end of the h-long time-step. It might be the case that a transition gets disallowed because its guard condition is satisfied only at intermediate points. This problem can be overcome if the hybrid system model can be rephrased so that all equalities are replaced by meaningful inequalities that successfully capture the same system dynamics. The other innate problem is that the system is not allowed to take multiple jumps in the time-step h. Reducing h can mitigate the problem, but it increases the computational complexity of evaluating long-term queries. Also, it is not a general solution.

Model 2: The One-Jump-Anywhere Model

The previous model is extended by allowing the discrete transition to occur at any intermediate point. Thus, the two possible evolutions of the "one-jump-anywhere" model in a time step h are:

- 1. Evolve continuously in the current discrete state s for the entire h time units

  OR

- 1. Evolve continuously in the current discrete state s for h_1 time units, where $0 \leq h_1 \leq h$
  2. Then make a discrete state transition to state s'
  3. Then evolve in the new discrete state s' for $h - h_1$ time units

The semi-algebraic expression capturing this model can be expressed thus:

$$\mathcal{T}(\ell \xrightarrow{h} \ell') = \{\exists \ell'', \ell''', h_1 \; \ell \xrightarrow[\mathcal{C}]{h_1} \ell'' \wedge \ell'' \xrightarrow[\mathcal{D}]{0} \ell''' \wedge \ell''' \xrightarrow[\mathcal{C}]{h-h_1} \ell'\} \bigvee \{\ell \xrightarrow[\mathcal{C}]{h} \ell'\}.$$

Limitations The expression that results is a reasonably accurate transition expression for most physical systems for most practical purposes. To guarantee accuracy, the hybrid system has to be hand-examined to ensure that not-allowing more than 1 discrete transition in a time-step h does not change the behavior one is interested in studying. Apart from just going in for a smaller h value to achieve this, one could introduce new transitions for every pair of possible consecutive discrete transitions. It is to be remembered that this can capture multiple jumps without intermediate continuous evolution, but is still not an exact depiction of the real system.

Model 3: The K-Jumps Model

We can now construct the expression for the *k-jumps-model*, where the hybrid automaton is allowed to take at most k jumps anywhere within the time-step h.

When $k = 1$, the *k-jumps model* is the same as the *one-jump model*:

$$\mathcal{T}^{\mathbf{1}}(\ell \xrightarrow{h} \ell') = \quad \{\exists \ell'' \; \ell \xrightarrow[\mathcal{D}]{0} \ell'' \wedge \ell'' \xrightarrow[\mathcal{C}]{h} \ell'\} \bigvee \{\ell \xrightarrow[\mathcal{C}]{h} \ell'\}.$$

The expression for the state of the system after at most k jumps is derived from the expression for the state of the system after at most $k-1$ jumps and then allowing *one* more jump:

$$\mathcal{T}^{\mathbf{k}}(\ell \xrightarrow{h} \ell') = \quad \{\exists \ell'', h_1 \; \mathcal{T}^{\mathbf{k-1}}(\ell \xrightarrow{h_1} \ell'') \wedge \mathcal{T}^{\mathbf{k}}(\ell \xrightarrow{h-h_1} \ell')\}.$$

Note that we can use the simpler *timed-jump* model instead of the *one-jump-anywhere* model as the time after which the system jumps h_i is a variable for each step. This effectively allows every possible jump to be taken. Also, as $k \to \infty$, the *k-jumps model* becomes exact.

Limitations Clearly, a good understanding of the hybrid system is necessary to fix the k ahead of time. A reasonable rule-of-thumb would be one more than the number of discrete states. It is easy to come up with systems where no matter what

k we fix, some behaviors get excluded. As an example, we can consider a system that counts the number of transitions taken in a variable say tr_{number}, where almost all the guard conditions are trivially satisfied. When $tr_{number} > n$, a special transition becomes possible. By picking $n > k$, we can ensure that the transition gets excluded no matter what k is used. It remains to be seen if such contrived systems have any parallels in common biochemical systems.

7.5.3 Simpler TCTL Expressions

For the dense-time logic TCTL to be model-checked, we described the one-step until operator as:

$$\begin{aligned} p \triangleright q \;=\; & q(R) \;\bigvee_{\forall v} \\ & \Big(\{\exists S \;\bigvee_{\forall u} \langle v, R\rangle \xrightarrow[\mathcal{D}]{0} \langle u, S\rangle \wedge q(S)\} \;\bigvee \\ & \quad \{\exists S, h \; (0 < h \leq \Delta) \wedge \langle v, R\rangle \xrightarrow[\mathcal{C}]{h} \langle v, S\rangle \wedge q(S) \wedge \\ & \quad \forall S', h' \; ((0 \leq h' < h) \wedge \langle v, R\rangle \xrightarrow[\mathcal{C}]{h'} \langle u, S'\rangle) \Rightarrow (p(S') \vee q(S'))\} \Big). \end{aligned}$$

The existential and universal *eventually* and *henceforth* operators are defined in terms of the corresponding *until* operator, which is computed by iterating (possibly indefinitely) over the *one-step until* operator.

In the previous section, we presented the approximations as being operational entirely in the discrete mode. However, it is possible to transfer the restricted transition relation models to the continuous TCTL domain as well. We demonstrate this approach with a simple instance of the application.

Let us try to devise a coarse approximation for TCTL model-checking, where time is continuous (TCTL) but the transition relation is restricted according to the *Timed-Jump Model*.

Definition 7.5.4 $\triangleright$ **For Restricted Semi-Algebraic Hybrid Systems With 1**

Timed Jump $p \triangleright q$ *is* True *if*

- *q is true now,* OR
- *For one of the possible current discrete states v, there exists a continuous transition (of exactly Δ time units) where p holds all along, and q holds at the end,* OR
- *For one of the possible current discrete states v, there exists a continuous transition (of exactly Δ time units) with p holding all along, after which there exists at least one state u to which a discrete transition can be taken such that q holds at the end.*

Thus:

$$\begin{aligned} p \triangleright q \;=\; & q \;\bigvee\nolimits_{\forall v} \\ & \quad p(r) \wedge \exists s \; \{(\langle v, r\rangle \xrightarrow[\mathcal{C}]{h} \langle v, s\rangle \wedge q(s)) \bigvee \\ & \qquad (p(s) \wedge \exists s' \; \bigvee\nolimits_{\forall u} \langle v, s\rangle \xrightarrow[\mathcal{D}]{0} \langle u, s'\rangle \wedge q(s'))\} \end{aligned}$$

Thus, depending on the application, we can do *accurate TCTL model-checking with restricted transition relations* as an alternative to *accurate CTL model-checking with restricted transition relations and time-discretization.*

7.6 Discussion

In this chapter, we have extended the theory of approximate verification of hybrid systems from the linear to the more expansive semi-algebraic domain. The algebraic model checking method presented in [222] was made more computationally practicable by extended bisimulation partitioning, approximation with general polyhedra and unions of simple polyhedra, and time discretization. For the extended bisimulation procedure suggested, we identified well-behaved subclasses based on some novel critical observations about the behavior of exactly onto linear and monotonic maps between arbitrary sets. For polyhedral approximations, we used the maximum and minimum values of the system variables and their possible growth in one step to expand the convex hull. We demonstrated how these same metrics (maximal growth along each dimension in one step) could be used to obtain a hyper-rectangular approximation of semi-algebraic sets. We also introduced a practical strategy to identify candidate hyper-rectangles that require quantifier elimination-based confirmation.

Time discretization was seen to simplify the problem by allowing fewer discrete jumps, excluding zeno paths, and by verifying the temporal property at certain sampling times rather than everywhere. The meaning of different restrictions on the number and timing of the discrete jumps was elaborated. While the "semi-algebraic" assumption limited the *continuous dynamics* by restricting the functions that can represent the flow, the requirement that a finite transition relation be available (for verification of temporal logic queries) lead to the restrictions on the *discrete dynamics*. More significantly, time-discretization could be coupled with the other methods described in this chapter to further improve the performance.

All these methods need to be refined to better handle discrete resets and symbolic approximations. More crucial is their actual implementation and performance analysis. On the purely algebraic side, approximate quantifier elimination and direct maximum-minimum estimation of a semi-algebraic set are those mathematical techniques, that need to be developed to further accelerate these methods. There are several approximating methods that are yet to be extended to semi-algebraic systems. These include: (1) piece-wise approximations of continuous dynamics; (2) problem domain transformation: optimal control using Pontryagin Maximum Principle, level sets of solutions to Hamilton-Jacobi-Bellman equations, sum of squares decomposition (semidefinite programming) and geometric programming; (3) predicate abstraction and qualitative simulation; (4) other geometric approximations: oriented rectangular hulls, zonotopes and ellipsoids. The next crucial research direction, which we do not cover in this thesis, is the determination of the practical applicability of these methods, trade-offs among them and suitable combinations of these approximations that work best with the available tools. Our implementation Tolque does not utilize any approximation algorithms at this point. While the theory of hybrid automata can be extended along many other lines, our contributions in this thesis come to a close at this point. We now return to the Systems Biology side and see how we can translate those problems into a format that can exploit the theory developed over the last four chapters.

Chapter 8

Metabolic Networks

Having carefully defined semi-algebraic hybrid automata and having worked out in detail the procedures for model checking and approximating them, we return to the Systems Biology portion of the problem statement. Earlier, we had discussed the kinds of problems Systems Biologists solve using hybrid automaton modeling and had provided a list of well-studied biochemical systems along with their kinetic mass action (KMA)-based ordinary differential equations (ODEs). However, we had not specified the details of the problems and how they can be transformed to a format which can exploit the theory we developed. In this chapter, we delve into the details of modeling biochemical networks. We start with the well-studied KMA-based networks, typically used for genetic regulatory networks. We then move to the complex realm of metabolism, where fast and slow reactions operate simultaneously, with much of the system remaining in pseudo-equilibrium. While the numerical handling of this scenario has already been well-documented in literature, we introduce a general framework for analyzing metabolic networks algebraically. Our second contribution here is extending an equilibrium-estimation technique called Flux Balance Analysis (FBA), also to the algebraic domain. Here we introduce the application of algebraic optimization to the Systems Biology domain. Thus, we show how the biochemistry problem description can be transformed into an entirely algebraic dynamical system specification. In this process, we have shown how techniques developed in previous chapters can be exploited for their analysis. One problem that we do not address is the conversion of a one-state dynamical system into a multi-state hybrid system.

8.1 Introduction

Problem Statement

We focus on the dynamical systems that naturally arise in a very important problem in Systems Biology: the analysis of metabolic networks. Metabolism [119] is used to denote the complete array of enzyme catalyzed biochemical reactions (involving "metabolites") continuously operating in a living cell, with the genetic regulatory and signal transduction aspects alone excluded. The fundamental law of chemical kinetics[91] provides ordinary differential equations governing the rate of change of the concentrations of the interacting metabolites (refer *Section* 3.2.2).

The system of ODEs lends itself to simplification because of three fascinating properties of metabolic networks: (1) A subset of the metabolites interact with each other through reactions much faster than the rest of the system; (2) These fast reactions always reach an equilibrium state, which is local (involving only this subset of metabolites) and momentary (it is modulated by the slower reactions in the rest of the system); (3) Mass is conserved during this equilibrium recomputation, and the position of the equilibrium is determined given the total concentration of the metabolites. These inherent properties of the "quasi"-equilibrium subnetwork lead to two broad questions that need to be addressed: (1) How can we characterize this equilibrium state algebraically? (2) We are forced to simulate / analyze the entire system in very small time-steps appropriate for the fast reactions. How can we move in larger time-steps sufficient for the slow reactions, and yet get a reasonable symbolic picture of the dynamics of the fast reactants?

State of the Art

Slow reactions have been modeled using the law of mass action. Their numerical simulation has been used to understand their temporal properties. In order to model a metabolic network accurately, the kinetic parameters should have been estimated under reversible conditions, in the presence of all metabolites likely to affect the activity in vivo, at the same temperature, pH, ionic strength, degree of macromolecular crowding, etc. and preferably by the same research group [94]. As such criteria are almost impossible to meet, the ability to handle unknown parameters or their believed ranges becomes an extremely powerful feature of any analysis tool, and an almost essential feature for a tool attempting to analyze vast networks of slow and fast reactions. Towards this end, the kinetic mass action models have been extended to the algebraic domain. These efforts include the techniques described in the earlier chapters where a rigorous algebraic formalism with decidable and approximable techniques was developed (also see AAMC I, II & III [239, 222, 221]). Models of biochemical networks derived from the differential equations dictated by the kinetic mass action laws have been successfully used for numerical simulation based analysis of temporal properties including reachability [31, 278, 77]. Similarly, the algebraic estimation of the equilibrium concentrations has also been extensively studied [54, 55, 78, 210, 302].

The fast reaction systems are typically subject only to a dynamical equilibrium characterization with minimal dynamic characterization. There have also emerged methods that try to integrate the two by dynamic KMA simulation of the slow reactions, under the assumption that the fast reactions quickly respond by reaching equilibrium. In other words, the sub-network of fast reversible reactions is assumed to be a self-contained metabolic network, whose response to the external stimuli has been characterized in advance; so it is reasonable to abstract out the time-curve which it follows to reach that new equilibrium. We assume that they are very fast reactions compared to the external switches – so the metabolic network very quickly moves to the new equilibrium in response to the stimuli. Though the assumptions necessary for such a model to be realistic seem very restraining, surprisingly, they are not unreasonable for many metabolic networks. Though the scheme for splitting the entire network into slow and fast reactions is problematic and under development, this approach is founded on sound biochemical principles, and is likely to be more useful in understanding the systems-level properties. Methods that venture in this direction include tendency modeling [288], dynamic flux balance analysis [200] and hybrid static / dynamic simulation [304].

Algebraic Approach

There has been no effort to handle both KMA-based simulation and direct equilibrium estimates (via KMA or FBA) algebraically. In this chapter, we present an entirely symbolic algebraic framework for the unified analysis of metabolic networks. We proceed by first mathematically characterizing the dynamical system to which metabolic networks correspond. The assumptions and constraints that capture the inherent biochemical structure are presented. We then show that the algebraic equilibrium description is decidable, both using kinetic mass action and its popular alternative flux balance analysis [173]. This is thus an algebraic generalization of approaches like tendency modeling [288], dynamic flux balance analysis [200] and hybrid static / dynamic simulation [304], as mentioned previously.

Our proof of the decidability of the algebraic approach will be based on the well-established Gröbner basis approach [73] for solving polynomial equations, and the decidability of semi-algebraic optimization using real quantifier elimination [276]. We then show how to go from the equilibrium description to its rate of change, which can then be combined with the ODEs of the slow reactants to get a complete algebraic description of the metabolic network. This directly leads to efficient algebraic model-checking, since we have ensured that all the interactions operate at roughly the same time scale. Hence a big time-step suitable for the slow interactions is sufficient.

The existing tools for metabolic networks are based primarily on numerical simulation based analysis, such as Gepasi [209], Systems Biology Workbench [154], E-Cell [159] and BioSpice [179]; so the ideas suggested in this chapter represent a step forward. Further, we can directly obtain the expression governing the dependence of the temporal property on the different parameters. This allows more immediate and generalized reasoning, as well as the estimation of the sensitivity of the equilibria to errors in rate parameters. The approach can also be used to estimate parameter ranges that would allow a certain observed phenomenon in the network. Unfortunately, key questions like what chemical species can be modeled as reaching equilibrium quickly and what species need explicit dynamic modeling, still remain in the domain of biochemists studying the specific system of interest. Our approach has the potential to yield a symbolic description of the external switches which can be used to alter the flux of the metabolites, and hence the position of its dynamic equilibrium. The seemingly inevitable pitfall is, once again, the computational complexity of the approach.

8.2 Background: Metabolism

Here, we introduce metabolic networks and the various means of analyzing them. Informally, metabolism stands for the complex enzyme-catalyzed pathways that produce and consume the metabolites in any living cell. Two independent properties of these enzymes are [119]: (1) *Control* – the ability to increase or decrease the net flux of a metabolite in response to external triggers, (2) *Regulation* – the ability to ensure that the net flux remains constant in the face of external perturbations. From the point of view of modeling, a very crucial aspect of metabolism is that different mechanisms for regulation and control exist on different time scales [119]:

1. On long time scales, amounts of enzymes are changed by enzyme synthesis and degradation.

2. On medium time scales, the activities of ready-formed enzyme are altered by covalent modification.

3. On short time scales, the activities of enzymes alter in response to changes in metabolite concentrations.

The second key characteristic is that these biochemical processes quickly attain dynamic equilibrium. In other words, the rates of production and consumption of these "metabolites" become equal; so their net concentration does not change.

8.2.1 Analyzing Metabolism

In contrast with the detailed kinetic mass action models, biochemists have focused on characterizing these steady-state fluxes and developed several approaches [130, 174]:

1. Metabolic Control Analysis (MCA) [149] provides a rigorous framework for characterizing the control that the different enzymes exert over the effective flux

along a pathway. It is sensitivity analysis in a sense, and uses detailed KMA equations to obtain the equilibrium concentrations.

2. Metabolic Flux Analysis (MFA) [95, 192] exploits the stoichiometric matrix to determine the minimal number of experimental measures necessary to fully describe the flux distribution.

3. Flux Balance Analysis (FBA) [173] tries to estimate the distribution of the flux during steady-state conditions. It uses the stoichiometric matrix and the input / output fluxes of the system to constrain the solution space. By assuming that the biochemical network would have so evolved as to optimize a biochemically meaningful function, the actual flux distribution is estimated. Common objective functions include maximization of biomass, maximization of ATP or reducing power, and maximization of the rate of synthesis of a particular product. (see next section for details)

4. Cybernetic approaches[227] extend the optimization assumption to states not in equilibrium as well, thus obtaining a complete dynamical description of the system. Now the response of microorganism to external stimuli is assumed to optimize some biochemically meaningful function at all times.

5. Metabolic Pathway Analysis (MPA) [257, 256] tries to solve the flux distribution by characterizing elementary modes and extreme pathways. The different optima can then be seen as the different extremities of the solution space, or as combinations of them.

8.2.2 Flux Balance Analysis

Flux Balance Analysis (FBA) is one of the means fo estimating the fluxes (rate of variation of concentration) of a metabolic network which has reached equilibrium. FBA only requires the knowledge of biochemical pathways. The enzyme kinetics and regulatory networks of all enzymes in a cell are *not* required to go through with this analysis.

The Mass Balance Equation is typically written as $S_n.V_n - b = -\frac{dX_n}{dt}$, where S is the stoichiometric matrix, v is the rate vector and b is the transportation vector. As per the steady-state assumption, there is no net accumulation of intermediate metabolites in the cell, i.e., $-\frac{dX_n}{dt} = 0$ for all intermediates. Thus $S_n.V_n - b = 0$. Since the number of fluxes is always greater than the number of metabolites in the cell, this system of linear equations (above) is under-determined, and so no unique solution exists for the fluxes v. Hence, optimization is required to find a particular solution of the system, where optimization effectively performs minimization or maximization of a particular flux. Common objective functions include maximization of biomass (cellular growth), maximization of ATP or reducing power, and maximization of the rate of synthesis of a particular product.

Several studies established this optimization phenomenon in metabolic networks: the growth and metabolic by-product secretion in wild-type *Escherichia coli* W3110 were found to be maximizing growth [287]; Computational models of *E. coli* metabolism based on physicochemical constraints were used to interpret mutant behavior [111], again assuming optimal growth objective function; *E. coli* metabolic network was shown to be maximizing growth under the experimental conditions considered [112].

FBA Evolution

We briefly review various extensions of the basic FBA technique have been suggested [173].

Mahadevan et al. [200] introduced Dynamic FBA and two kinds of optimization schemes. They concluded that the static optimization approach was computationally simpler to implement provided all of the constraints were linear, whereas the dynamic optimization approach was more flexible and should be quite suitable for the incorporation of experimental data. Beard and coworkers [56] explored the impact of a full energy balance analysis on the predictions of a flux balance analysis. Beard and coworkers found that combination of the energy balance analysis with FBA gave the same optimal growth rate, but the observed fluxes were substantially different.

Lee et al. [191] presented a multi-objective linear programming in FBA for simultaneous consideration of several objectives, by which Pareto solutions are generated and the corresponding flux distribution profiles are explored to understand how the internal fluxes are changed when pathways are switched for another purpose. Minimization of metabolic adjustment (MOMA) [260] was suggested to improve the prediction efficiency of FBA for studying *E. coli* mutants. In contrast to FBA, MOMA does not assume optimality of growth or of any other metabolic function. Instead, for perturbations such as gene deletions, MOMA approximates metabolic phenotype by performing distance minimization in flux space. FBA has also been extended by incorporating exploration of alternative classes of objective functions [199]. Regulatory On-Off Minimization (ROOM) [263] is a model for predicting the behavior of metabolic networks in response to gene knockouts. It is based on minimizing the number of significant flux changes with respect to the wild-type.

8.3 Algebraic Analysis of a Biochemical Dynamical System

8.3.1 Motivation

The most accurate simulation of a metabolic network will naturally require the complete kinetic mass action equations and the allosteric models. The most common strategy to handle the complexity is to identify the different time-scales at which different reactions occur, and then make reasonable assumptions. For example, while studying a VLSI circuit, the underlying components such as transistors can be assumed to have a certain "instantaneous" response, even though the detailed characterization of their voltage response is available. In biochemistry, fast reactions are those that reach equilibrium quickly. Indeed, when a network of fast reactions reaches dynamic equilibrium, the rate of production of the metabolites is equal to their rate of consumption, and hence their concentrations remain constant. So while analyzing / simulating a big network, it is assumed that the fast sub-networks reach equilibrium in one time-tick of the global-simulation.

While such a theoretical model has already been presented in one form or another [288, 200, 304], we now present the new analytical framework for modeling and studying such a biochemical network symbolically. This involves the following steps:

1. Start with a complete kinetic mass action model of the entire network, with variables (parameters) substituted in place of unknowns.

2. Identify subnetworks of fast reactions (from biochemistry literature).

3. Compute the dynamic equilibrium concentrations and fluxes of the fast subnetworks. This can be done accurately from given the KMA model, using Gröbner basis. This can also be obtained from the FBA approach, using al-

gebraic optimization. We will thus get an (semi) algebraic description of the equilibrium state of the system.

4. Now the entire system is simulated using algebraic KMA models, with the fast reactants updated as per the equilibrium equation previously derived.

8.3.2 An Algebraic Framework

In a typical model of a network of interacting biochemicals, the variables represent the concentrations of the biochemical species. The differential equations governing their time evolution are given by the kinetic mass action laws, with the parameters being the rate constants [91]. Let k_is denote the rate constants, n_is the number of molecules of the metabolite that appear in the reactions, and W_js their concentrations.

Definition 8.3.1 ***Biochemical Dynamical System***

$$\dot{W}_h = +\Sigma_{i,j,..} k_{i,j,..\neq h} n_{i,j,..} W_i W_j \cdots - \Sigma_{i,..\neq h} n_{i,..} k_{h,i} W_h W_i \cdots . \quad \square$$

Here, the first summation represents all processes producing W_h, and the second represents all processes consuming it. The number of W_js multiplied in each term depends on the number of molecules of reactants / products in that reaction. Note that even higher order terms like $W_i^3 W_j^{10} W_k^5$ are possible.

Example 8.3.1 *For the reaction $aA + bB \longleftrightarrow cC + dD$, the rate of the forward reaction $v_f \equiv k_f[A]^a[B]^b$ and the rate of the backward reaction $v_b \equiv k_b[C]^c[D]^d$, where k_f and k_b are the forward and backward rate constants respectively. The rates of the reactions are related to the rate of individual reactants via the number of molecules. Thus,*

$$\frac{1}{c}\dot{C} = \frac{1}{d}\dot{D} = -\frac{1}{a}\dot{A} = -\frac{1}{b}\dot{B} = (v_f - v_b).$$

At equilibrium, all flows are zero since the forward and backward rates are equal, and $\frac{k_f}{k_b} \equiv K_{eq}$*, the equilibrium constant.* $\square$

Having characterized the KMA dynamical system as a system of ODEs, recall that the algebraic temporal logic analysis of such systems involves constructing the *Flow* expression (see *section* 5.2).

Definition 8.3.2 ***Flow Relation*** *of a dynamical system* $Flow(V, V', t, h, K)$ *is a relation linking the symbolic state of system* V *at time* t *with the symbolic state* V' *at time* $t + h$. $\square$

Thus, the metabolic dynamical system corresponds to a simple one-state semi-algebraic hybrid automaton with flows of the form $W' = W + h\dot{W}$. Thus the techniques developed in *Chapters* 5,6 and 7 may be applied for analysis.

8.4 Algebraic Analysis of Metabolic Dynamical Systems

Having discussed general biochemical dynamical systems, we now detail the special sublcass "metabolic dynamical systems", where a subset of the system's variables are involved in fast interactions that quickly reach a local equilibrium. We first formalize these assumptions about metabolic dynamical systems.

Definition 8.4.1 ***Metabolic Network***

Three Types of Reactants *A Metabolic Network has three kinds of metabolites* X, Y *and* Z*, and symbolic (rate) parameters* K*, such that:*

1. Dynamic Reactants *All the reactions involving these metabolites (denoted by* X*) are modeled using detailed kinetic mass action-based differential equations. They may be involved in only one reaction. Typically, these reactions will be slow and irreversible.* $\dot{X} = F(X, Z, K)$

2. Pseudo-Equilibrium Reactants *All the reactions involving these metabolites (denoted by Y) are modeled in terms of their dynamic equilibrium alone. They always partake in at least one reaction as a substrate and in at least one reaction as a product. Typically, these reactions will be fast and reversible.* $\dot{Y} = G(Y, Z, K)$

3. Interface Reactants *These reactants (denoted by Z) interact with both the Dynamic Reactants and the Pseudo-Equilibrium Reactants. Thus, their kinetic mass action based flow equations (from slow reactions) will be modified because of the fast reactions with the Pseudo-Equilibrium Reactants. They may be involved in only one reaction.* $\dot{Z} = D(X, Z, K) + P(Y, Z, K)$

The Equilibrium Constraints *The underlying biochemical structure can be expressed thus:*

1. *Each substrate of each reaction involving at least one interface metabolite also as a substrate, is associated with mass conservation equation. The total concentration of these substrate metabolites in their many chemical forms at equilibrium is captured using* Equilibrium Pool Variables T.

2. *The* mass conservation equations $T = \mathcal{M}(Z, Y)$ *have the form:* $T_i = \Sigma_{j \in Pool_i} W_j$, *where* $Pool_i$ *represents the set of the different chemical forms* W_j *that the* i*-th substrate metabolite exists in.*

3. *The equilibrium concentrations of* Z *and* Y *are expressible in terms of the equilibrium pool concentrations, i.e., semi-algebraic relations* $E_Z(Z, T, K)$ *and* $E_Y(Y, T, K)$ *exist.* $\square$

Note 8.4.1 *One of the assumptions that becomes valid in metabolic dynamical systems is that: during the momentary recomputation of the equilibrium point, the total* *concentrations of the pool variables T do not change. This is because the time required to recompute equilibrium is negligible compared to the time-step used for simulating the slow reactions. Hence, the change in the concentrations due to the slow reactions is negligible compared to the effect of the equilibrium recomputation.*

Mathematically, the dynamical system that such a metabolic network corresponds to may be summarized thus:

Definition 8.4.2 ***Metabolic Dynamical System*** *A Metabolic Dynamical System has three kinds of variables X, Y and Z, and symbolic (rate) parameters K, such that:*

$$\begin{array}{rcl} \dot{X} & = & F(X,Z,K) \\ \dot{Y} & = & G(Y,Z,K) \\ \dot{Z} & = & D(X,Z,K) + P(Y,Z,K) \\ T & = & \mathcal{M}(Z,Y), \ \textit{where}\ T_i = \Sigma_{j \in Pool_i} W_j \\ \dot{T} & = & H(X,Z,K) \\ Z = E_Z(T,K) & \textit{and} & Y = E_Y(T,K) \quad \square \end{array}$$

Example 8.4.1 *Consider a simple metabolic network composed of just two reactions:*

- *A slow irreversible reaction $A + B \xrightarrow{k_s} R + S$*
- *A fast reversible reaction $E + S \underset{k_r}{\overset{k_f}{\rightleftharpoons}} C$.*

This could represent say an enzyme (E) and substrate (S) interacting to produce the enzyme substrate complex (C). We want to study how an external slow reaction producing the substrate can control the position of equilibrium. Let us denote $[E]$ by e, $[S]$ by s, $[C]$ by c, $k_f/k_r = K$. The dynamic reactants X are A,B and R. The *pseudo-equilibrium reactants Y are E and C. The interface reactant is S. Their flow equations are:*

$$\begin{aligned}
\dot{a} &= \dot{b} = -\dot{r} = -k_s ab \\
\dot{e} &= -\dot{c} = k_r c - k_f es \\
\dot{s} &= k_s ab + k_r c - k_f es
\end{aligned}$$

The only reaction with an interface metabolite as a substrate is $E+S \underset{k_r}{\overset{k_f}{\rightleftharpoons}} C$. The mass-conservation equations can be written for the two substrates E and S as $e_T = e + c$ and $s_T = s + c$, where e_T and s_T are the new equilibrium pool variables. □

Having clearly defined the dynamics and constraints, we now show how to compute the equilibrium description, and then extract the *Flow* expression. This will reduce the metabolic dynamical system to a *biochemical dynamical system* where the fast ODEs of the pseudo-equilibrium reactants have been approximated by slow reactions.

8.4.1 Detailed Kinetic Mass Action Based Approximation

We want to characterize the momentary pseudo-equilibria that the fast variables (interface and pseudo-equilibrium metabolites) reach in response to a change in the slow interactions (dynamic reactants).

Equilibrium Characterization

Let Z and Y be the initial concentrations (at the start of the time-step) of the interface and pseudo-equilibrium metabolites. Let their equilibrium concentrations be Z' and Y'. The complete set of constraints which must be true at equilibrium are given by:

Definition 8.4.3 ***KMA Equilibrium Relation***

$$\begin{aligned} E(Z',Y',T',Z,Y,T,K) \;&\equiv \\ \{P(Z,Y,K)=0 \;&\wedge\; G(Y,Z,K)=0 \wedge \\ T=\mathcal{M}(Z,Y) \;&\wedge\; T'=\mathcal{M}(Z',Y') \wedge T'=T\} \quad \square \end{aligned}$$

This equilibrium characterization lends itself to simplification by algebraic methods. In other words, the equilibrium equations (and inequalities) in *Flow* can be solved before being fed into the TCTL model checker.

Equilibrium Simplification using Gröbner Bases Since the KMA-based approach involves only equalities ("algebraic sets"), the equilibrium relation E is effectively just a system of polynomial equations, which needs to be solved for the Z' and Y' (in terms of Z, Y, t, h and K). In the case of metabolic dynamical systems, we know that Z' and Y' are expressible in terms of the equilibrium pool variables T. Clearly, the well-established method for solving such systems of simultaneous multivariate polynomial equations with symbolic parameters is Buchberger's 20-year old Gröbner Basis algorithm [73]. Its many implemented forms include CoCoA [88] and Macaulay-2 [133].

Remark 8.4.1 *Algebraic methods for enzyme kinetics have been elaborately studied since the advent of computer algebra systems (for instance, see Bayram's 1993 thesis [54]). The issue of simultaneous solution of polynomial equations, especially in the context of biochemical networks, has been addressed by Bayram, Barnett and colleagues [55, 78, 210] (see Barnett [46, 47] for an exhaustive summary of recent work that applies computer algebra to the life sciences).*

Definition 8.4.4 ***Simplified KMA Equilibrium Relation***

$$E(Z', Y', Z, Y, T, K) \equiv \{T = \mathcal{M}(Z, Y) \wedge Z' = E_Z(T, K) \;\wedge\; Y' = E_Y(T, K)\},$$

where E_Z and E_Y represent the solutions obtained using the Gröbner basis technique over

$$\{P(Z, Y, K) = 0 \;\wedge G(Y, Z, K) = 0 \wedge T = \mathcal{M}(Z, Y)\}. \quad \square$$

Note 8.4.2 *The existence of functions E_Z and E_Y follows from the assumptions about metabolic networks. However, we could characterize the algebraic requirements for such a T-basis, in terms of the rank of the matrix and so on. We do not delve into this here.*

Flow Description

Let us now construct $Flow(X, Y, Z, T, X', Y', Z', T', t, h, K)$ – the continuous flow expression, which connects the state of the system X, Y, Z at time t and the state of the system X', Y', Z' at time $t + h$.

The ODEs for Z and Y can be directly derived from E_Z and E_Y by differential calculus. The equilibrium concentrations of Z and Y are expressible in terms of the equilibrium pool variables T. Thus,

$$\dot{Z} = \frac{dE_Z(T, K)}{dt} = \frac{\delta E_Z(T, K)}{\delta T}.\frac{dT}{dt} = \frac{\delta E_Z}{\delta Z}H.$$

The same applies to Y as well. Thus we have our final result:

Definition 8.4.5 ***Approximated Metabolic Dynamical Systems***

$$\begin{aligned}
\dot{X} &= F(X, Z, K) \\
\dot{T} &= H(X, Z, K) \\
\hat{\dot{Z}}(T, X, Z, K) &\approx \frac{\delta E_Z}{\delta T} H \\
\hat{\dot{Y}}(T, X, Z, K) &\approx \frac{\delta E_Y}{\delta T} H \quad \square
\end{aligned}$$

Having derived the approximate ODE description, the dynamical system is now amenable to direct analysis by the methods suggested for *Biochemical Dynamical Systems*. Recall that the flow expression is simply:

$$\begin{aligned}
\mathit{Flow}(\{X, Y, Z, T\}, \{X', Y', Z', T'\}, t, h, K) \equiv & \\
\{(X' &= X + hF(X, Z, K)) \\
\wedge(T' &= T + hH(X, Z, K) \\
\wedge(Z' &= Z + h\hat{\dot{Z}}(T, X, Z, K) \\
\wedge(Y' &= Y + \hat{\dot{Y}}(T, X, Z, K))\}
\end{aligned}$$

The approach is summarized in *Figure* 8.1.

Example 8.4.2 *Let us continue with the example introduced earlier.*

If we wanted to simulate this system directly, the time-step of integration would need to be small enough for the approximation error of the fast reactions to be reasonable. For example, if all concentrations are initially 1*, then the rate constant of the fast reactions (k_f, k_r) would typically be a* 100 *times that of the fast reactions (k_s). So we will be forced to pick a time-step say* 0.001 *units. However, if we compute the rate of change of equilibrium instead, we can simulate the system with a time-step of* 0.1.

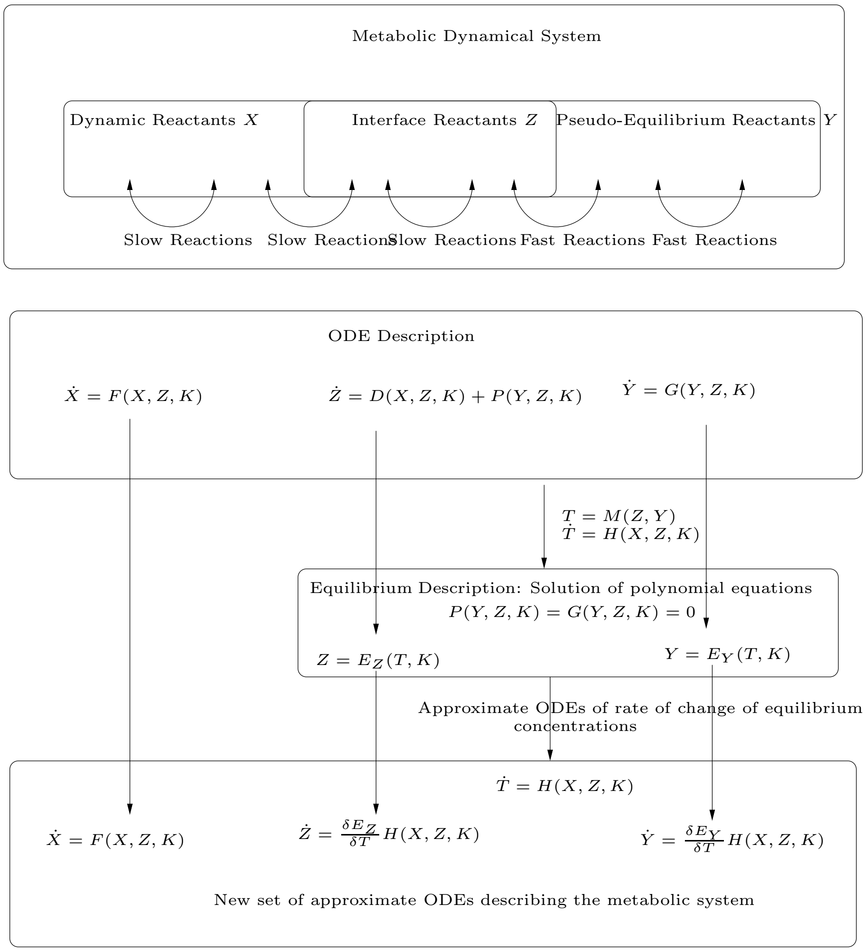

Figure 8.1: Kinetic Mass Action based approximate analysis of a metabolic dynamical system

At equilibrium, only one equation needs to hold: $k_f es = k_r c$. *The total amount of enzyme* e_T *is assumed to be fixed throughout the simulation, and is a parameter, along with* k_f, k_r *and* k_s. *The total amount of substrate changes with time, but is conserved during the period when the equilibrium is recomputed. Thus the equilibrium relation* *is given by:*

$$
\begin{aligned}
E(\{s'\},\{e',c'\},\{s\},\{e,c\},K) \;\equiv& \\
&\{k_f es - k_r c = 0 \quad\wedge\quad (e+c=e'+c') \wedge (s+c=s'+c') \\
&\wedge e \geq 0 \wedge c \geq 0 \quad\wedge\quad s \geq 0 \wedge e' \geq 0 \wedge c' \geq 0 \wedge s' \geq 0\}.
\end{aligned}
$$

This reduces to the quadratic equation

$$c^2 - c(s' + 2c' + e' + 1/K) + (s' + 2c' + e') = 0,$$

with the all positive constraint leading to the solution:

$$
\begin{aligned}
c &= \frac{(s' + 2c' + e' + 1/K) - \sqrt{(s' + 2c' + e' + 1/K)^2 - 4(s' + 2c' + e')}}{2} \\
e &= e' + c' - c \\
s &= s' + c' - c
\end{aligned}
$$

The modified flow expression would thus be:

$$
\begin{aligned}
&\hat{Flow}(\{s'\},\{e',c'\},\{s\},\{e,c\},t,h) \equiv \\
&\{\exists s'',\ (s'' = s + hk_s ab) \wedge c^2 - c(s'' + 2c' + e' + 1/K) + (s'' + 2c' + e') = 0 \\
&\wedge (c = e' + c' - c) \wedge (s = s' + c' - c) \\
&\wedge e \geq 0 \wedge c \geq 0 \wedge s \geq 0 \wedge e' \geq 0 \wedge c' \geq 0 \wedge s' \geq 0 \wedge s'' \geq 0\}
\end{aligned}
$$

Alternatively, we can express the equilibrium using the equilibrium pool variables as

$$\frac{(s_T + e_T + 1/K) - \sqrt{(s_T + e_T + 1/K)^2 - 4(s_T + e_T)}}{2}.$$

8.4.2 Flux Balance Analysis Based Approximation

FBA [173] is the most popular alternative to detailed kinetic modeling of the fast metabolic reactions, because it only requires knowledge of the wirings of biochemical pathways and not their kinetic parameters or regulatory functions. The essence of flux balance analysis is optimizing a function under a set of constraints. Since we are stressing the ability to treat variables and parameters as symbols, we first formalize the established theory that algebraic optimization is decidable. The aim of the optimization can be reworded as: for all values U that differ from the optimal value $\check{U}$ and still satisfy the constraints $C(U,V)$, the value of the function $f(U,V)$ is less than $f(\check{U},V)$. This immediately leads to the following characterization of $\{\check{U},V\}$:

Definition 8.4.6 ***Optimization Relation*** $\mathit{Optimize}(f(U,V),C(U,V),\check{U}) \equiv C(\check{U},V)\bigwedge\{\forall U,\ (U\neq\check{U}\wedge C(U,V))\Rightarrow(f(U,V)<f(\check{U},V))\}$ □

Theorem 8.4.1 ***Decidability of Semi-Algebraic Optimization*** *The set of optimal values $\check{U}$ of the n-length vector U satisfying the semi-algebraic constraints $C(U,V)$ where V represents the vector of variables not being optimized (i.e., the parameters), which leads to the maximum value of a polynomial function $f(U,V)$ is decidable.*

Proof 8.4.1 *Clearly, if C is semi-algebraic and if f is polynomial, the expression $\mathit{Optimize}(f(U,V),\check{U},V,) \equiv C(\check{U},V)$ is a quantified semi-algebraic set, which is decidable [276].* □

A simpler approach to obtaining the optimal value of the function being optimized is well-established in literature. This involves first characterizing the function f to be optimized by a new variable z. This is done by conjuncting a constraint of the form $z - f(U) > 0$ to the original system, and eliminating all U. The result will express z as $g(V)$, a function of the parameters alone. The task is to now use this bound on the range of f and calculate the bound on the variables Z. This can be done by conjuncting with the system invariants. However, the procedure outlined in our proof is a more direct way of capturing the optimal values (and not the value of the function being optimized).

Now that the central computational engine is in place, it just remains to reformat the flux balance analysis problem so it can exploit this algebraic device.

Equilibrium Characterization

Let $C(Z', Y', Z, Y, K)$ represent the semi-algebraic constraints on the kinetic parameters, rates of change, bounds on parameters, energy balance equations, etc. Let $O(Z', Y', Z, Y)$ represent the function[1] that the metabolic network is assumed to be optimizing. Thus, the complete set of equations and inequalities that need to be true at equilibrium may be represented thus:

Definition 8.4.7 ***FBA Equilibrium Relation***

$$\hat{E}(Z', Y', Z, Y, T) \equiv \mathit{Optimize}(O(Z', Y', Z, Y, T), M(Z', Y', Z, Y, T), \{Z', Y'\}),$$

where $M(Z', Y', Z, Y, T) \equiv \exists K, \{C(Z', Y', Z, Y, K) \wedge E(Z', Y', Z, Y, T, K)\}$ *with* E *being the* KMA Equilibrium Relation. $\square$

Note 8.4.3 *Since FBA assumes that kinetic parameters K are unavailable, the effective set of constraints over which the optimization must be performed is obtained by eliminating K. The existential quantifier captures the assumption that the network would have so evolved that the kinetic parameters effectively make the optimization possible.*

[1] The primed variables may be necessary to capture functions involving the rate of change of concentrations

Algebraic Optimization using Quantifier Elimination Here, we need to optimize a function under semi-algebraic constraints (i.e., equations and inequalities). Gröbner bases cannot be used as they can handle only equalities. Instead, the general technique of real quantifier elimination [276, 151, 109] has to be employed to perform the algebraic optimization [20]. One ideal system is the Maple toolbox Symbolic-Numeric toolbox for Real Algebraic Constraints (SyNRAC) developed by Anai and Yanami [23, 300]. SyNRAC includes a collection of symbolic, numerical, and symbolic-numeric solvers based on quantifier elimination. These handle semi-algebraic constraints efficiently, and can perform even non-convex parametric Semi-Definite Programming required by algebraic FBA.

Linking Fluxes and Concentrations The flux balance analysis approach is typically defined using the fluxes rather than the concentrations. Each flux term has a typical form $k_{i,j}X_iX_j$. Thus, these new quadratic equations in the X_is needs to be solved. While the flux-approach is simpler if we are not interested in concentrations, the introduction of new variables is unnecessary if time-profiles of the equilibria are desired. Thus, the flux balance equations which are typically written as $\Sigma_i F_i = 0$, are now written explicitly as $\Sigma_i k_i X_i Y_i = 0$. These rate parameters are removed during optimization, i.e.,

$$\exists K,\ C(\check{X}, Y) \bigwedge \{\forall X,\ (X \neq \check{X} \wedge C(X, Y)) \Rightarrow (f(X, Y) < f(\check{X}, Y))\}.$$

This is equivalent to the assumption that the kinetic parameters are such that the function f is maximized. Thus, the flow may be expressed as:

Definition 8.4.8 ***FBA Flow Relation***
$\hat{Flow}(\{X, Y, Z, T\}, \{X', Y', Z', T'\}, t, h, K) \equiv \{(X' = X + hF(X, Z, K)) \wedge (T' = T + H(X, Z, K)) \wedge \hat{E}(Z', Y', Z, Y, T')\}$

Note 8.4.4 *In some cases, the solution after optimization might be an algebraic equation of the form $Z' = E_Z(Z, Y, T, K)$ and $Y' = E_Y(Z, Y, T, K)$. Thus, we can write:*

$$\begin{aligned}\dot{Z} &= \frac{\delta E_Z}{\delta Z}\dot{Z} + \frac{\delta E_Z}{\delta Y}\dot{Y} + \frac{\delta E_Z}{\delta T}\dot{T} \\ \dot{Y} &= \frac{\delta E_Y}{\delta Z}\dot{Z} + \frac{\delta E_Y}{\delta Y}\dot{Y} + \frac{\delta E_Y}{\delta T}\dot{T}.\end{aligned}$$

By solving these two equations, one can obtain the general solution:

$$\begin{aligned}\dot{Y} &= \frac{\frac{\frac{\delta E_Y}{\delta Z}\frac{\delta E_Z}{\delta T}}{1-\frac{\delta E_Z}{\delta Z}} + \frac{\delta E_Y}{\delta T}}{1 - \frac{\frac{\delta E_Y}{\delta Z}\frac{\delta E_Z}{\delta Y}}{1-\frac{\delta E_Z}{\delta Z}} - \frac{\delta E_Y}{\delta Y}}\dot{T} \\ \dot{Z} &= \frac{\frac{\delta E_Z}{\delta Y}\dot{Y} + \frac{\delta E_Z}{\delta T}\dot{T}}{1 - \frac{\delta E_Z}{\delta Z}}.\end{aligned}$$

Also note that $\dot{Z} = \frac{\delta E_Z}{\delta T}\dot{T}$ and $\dot{Y} = \frac{\delta E_Y}{\delta T}\dot{T}$ derived in the KMA based approximation is just a special case where $\frac{\delta E}{\delta Y} = \frac{\delta E}{\delta Z} = 0$.

8.5 Discussion

It has been reiterated that a systems level analysis (as opposed to a reductionist approach) of biology is crucial for giant leaps in our understanding of life to be possible [93], and to contribute more directly to the drug discovery process [76, 92]. The framework outlined in this chapter demonstrates how richer and deeper symbolic queries can be phrased about metabolic networks, in a format that makes them amenable to the algebraic temporal logic analysis techniques described in the earlier chapters. There are several avenues of extending this research:

Solving Polynomial Equations Efficiently

This will necessitate incorporating more efficient and less general techniques for equilibrium estimation and heuristics for choosing between them[236]. These include direct algebraic simplification using quantifier elimination, the Wu-Ritt resultant algorithm for solving simultaneous equations[204, 55], resultant computation followed by eigen decomposition of a generalized companion matrix[291], and linearization-based techniques for solving quadratic systems.

Performing Algebraic Optimization

While quantifier elimination based algebraic optimization is relatively general, less general but more efficient quantifier elimination algorithms could be applicable in many scenarios. Some approaches that merit attention include constraint logic programming with first-order constraints $CLP(RL)$ [271] based on Redlog, systems theoretic algebraic optimization [161] and semidefinite programming [234]. Dolzmann et al. [108] survey three implemented real quantifier elimination methods: partial cylindrical algebraic decomposition, virtual substitution of test terms, and a combination of Gröbner basis computations with multivariate root counting. Weispfenning[296] has described an optimized method for the elimination of linear variables from a Boolean combination of polynomial equations and inequalities. Hong et al. [152] show how to write all common stability problems as quantifier-elimination problems, while Jirstrand [162] analyzes dynamical systems described by polynomial differential equations subjected to constraints on control and system variables using quantifier elimination.

Systems Biology Extensions

Algebraic analysis can be used to perform the classification of the dynamical system interactions as fast and slow, and the decomposition into sub-modules of a large network. The newer constraints (see *Section* 8.2.2) that have been defined in an effort to bring the results of the optimization-based simulation closer to experimental values also need to be extended to the algebraic domain. Other approximate methods to estimate the equilibrium fluxes (e.g., cybernetic modeling[227]) also need to be extended to the algebraic domain. Also, the problems in modeling and analyzing biochemical networks are becoming increasingly related to more general scenarios handled much more mathematically rigorously in the field of Control Theory[301, 158, 255].

Conclusion

The foundation of *Algorithmic Algebraic Model Checking* is the decidability of real quantifier elimination [276]. Over the last couple of chapters, we have established that reachability analysis [239], TCTL model-checking [222] and many popular approximation methods [221] can be reduced to iterative quantifier elimination calls. For this procedure to be decidable, only Boolean combinations of polynomial equations and inequalities can appear in the description of the system's dynamics. This motivated the definition of the *semi-algebraic hybrid automaton* class. In this chapter, we did not use the discrete aspects of hybrid systems. We instead focused on dynamical systems which have only one discrete state and no notion of discrete transitions. We showed how a new class of problems that appear in metabolic network analysis may be reformulated as an algebraic biochemical system. Our algebraic framework allowed the description of the behavior of the metabolic network symbolically, in terms of when it will be in which equilibrium, and when it will move. Having completed the documentation of the Systems Biology, Hybrid Automata and Model Checking theory, we now discuss the preliminary implementation of some of these algorithms in the symbolic algebraic dense-time model-checker Tolque [222, 224].

Chapter 9

Tolque: An Algebraic Model Checker

It is natural to hope that the theory hitherto developed can immediately be applied to the Systems Biology domain, leading to the solution of important problems that radically alter our understanding of life. However, such naïveté is thwarted by the double-exponential complexity of the quantifier elimination procedure. In this chapter, we find that Qepcad is unable to solve "big" quantifier elimination problems in reasonable time on a reasonable computer. Nevertheless, we are able to get a sense of the power of a dense-time TCTL model-checker that can handle semi-algebraic hybrid automata. We document many simple examples where symbolic solutions are obtained, and some cases where Tolque stalls quickly due to lack of sufficient memory. On the Systems Biology side, we discuss the repressilator and the Delta-Notch examples. The purpose of this chapter is to motivate the development of a powerful tool that can exploit several existing computer algebra libraries, approximation schemes and paralellizable algorithms to solve general semi-algebraic TCTL queries over hybrid automata. We also include a review of popular software tools for model-checking / verification.

9.1 Tolque: A Preliminary Prototype

A preliminary version of a symbolic algebraic model checker that uses the TCTL based approach outlined in this thesis has been implemented. This quantifier-elimination-centric model checker, christened Tolque [222] (an acronym for "TempOral Logic via QUantifier Elimination"), takes as input a semi-algebraic hybrid automaton specification (with the flow equations already approximated if necessary) and an *Existential Until* ($p \; \exists\mathcal{U} \; q$) query. It then computes the fixpoint $p \; \exists\mathcal{U} \; q \;=\; \mu X.(q \vee (p \triangleright X))$ [143] by using Qepcad [151] (an acronym for "Quantifier Elimination by Partial Cylindrical Algebraic Decomposition") to perform the quantifier elimination in $p \triangleright X$. The entire process is automated in this *C/C++* implementation that runs in *Linux*.

A basic module reads in the definition of the hybrid system - the states, the transitions, the flows, etc. and generates its transition relation. A *lex/yacc*-based parser/generator is used to read / write the semi-algebraic formulæ. The CTL query on this system is translated into a series of quantifier elimination problems which are solved by calls to Qepcad [151]. Qepcad is an implementation of quantifier elimination by partial cylindrical algebraic decomposition, based on the original CAD algorithm[90] (see *Section* 5.1.2 for technical details and summary). A typical input formula looks like: $(Ex)[ax^2 + bx + c = 0]$., with the output generated being: $4ac - b^2 \leq 0 \wedge [c = 0 \vee a \neq 0 \vee 4ac - b^2 < 0]$.

Both discrete and continuous modes of operation are supported. The discretization scheme that has been chosen is the *Euler Forward* one, though this is not assumed by the implementation. In the continuous mode, no restrictions are imposed and conventional TCTL model checking is performed. In the discrete mode, the discrete-time *one-jump-anywhere* model of the transition relation has been chosen, and the state invariant is checked only at the start of each time-step.

9.2 Survey of Computational Tools

As discussed in *Sec.* 6.4, fundamental algorithmic concerns have limited the development and application of temporal analysis techniques to severely restricted hybrid automaton subclasses, and in addition, the methods often sacrifice accuracy or symbolic capabilities. In this section, we briefly document some of the major software tools for analyzing the dynamics of discrete and hybrid automata.

9.2.1 Discrete Model Checkers

Purely discrete dynamical systems do not present problems comparable with those faced in the analysis of hybrid automata. Thus, we survey the approaches developed for these systems only very briefly:

1. `smv` [208] The current user-friendly industrial quality version of this seminal tool is the Cadence `SMV` model checker, which supports both extended SMV and synchronous verilog. CTL and LTL are both handled, in addition to other forms of specification.

2. `NuSMV` [85, 84] This `SMV` rewrite supports the original `SMV` input language, and also has CTL / LTL modelcheckers.

3. **Spin** [150] is an LTL model checker for distributed software systems (described in the promela formalism), based mostly on Vardi and Wolper's pioneering work [286].

4. DSSZ is a suite comprising BDD-based CTL and LTL model checkers for safe Petri Nets [229].

5. **Bogor** [110] is a model checker specifically designed for software verification.

6. **SLAM** is the software verification and debugging project of Microsoft [44].

7. PVS (Prototype Verification System) [233] is a comprehensive verification / theorem-proving tool with a powerful specification language, based on decision procedures for different theories.

8. CVC Lite [48] is an interactive library with natural-language support for verifying quantifier-free first-order formulas over different theories.

9. **ASTRE** [96] is a static program analyzer for checking for run-time errors in C programs.

9.2.2 Hybrid Model Checkers

As seen in *section* 9.2.1, a plethora of model checking, verification, automated reasoning and theorem prover tools for discrete systems have become available in the short span of two decades. Different tools focus on different problem domains, make differing assumptions and employ one of the many efficient and optimal algorithms developed for that restrictive sub-domain. Hence, it becomes very difficult to justify one tool or approach as being better than the others.

In order to better gauge the relative merit of tools comparable to Tolque, we observe the key characteristics of the algorithmic algebraic model checking approach:

- Hybrid automata, with continuous and discrete dynamics, can be analyzed.

- Semi-algebraic sets can appear in the description of the continuous dynamics. If the differential equations are polynomial, they can be approximated using symbolic version of discretization schemes. If the dynamics are innately semi-algebraic, no approximation is necessary.

- Semi-algebraic sets can appear in the description of the discrete dynamics, with constraints on the resets.

- All expressions can be symbolic, i.e., with algebraic parameters.

- The initial condition can be unspecified or symbolic.

- The TCTL query should be of the form $p\exists\mathcal{U}\backslash\sqcup\rangle\updownarrow q$, where p and q are semi-algebraic sets.

- The algorithm is iterative quantifier elimination which is not guaranteed to terminate unless the solution is *true*. There are no heuristics or optimizations implemented to exploit systems with simpler dynamics (though they have been theoretically developed [221]).

- There is no user-friendly interface currently implemented. It is a command-line tool that uses a text file as the input (with integration into Simpathica in progress).

Based on these observations, we now survey the relevant features of some of the popular tools for analyzing the temporal dynamics of hybrid systems, which are relatively in their early stage of development. For the reader's convenience, a comparison of these features has been tabulated in *Table* 9.1.

1. Kronos [100, 303] handles real-time systems modeled as timed automata, and model checks queries described in TCTL.

Table 9.1: Tools for the Analysis of Hybrid Automata

Tool	Details
Kronos [100, 303]	timed automata, TCTL
Uppaal [58, 59]	timed automata extended with data types
HyTech [16, 142, 146]	HA with piecewise constant polyhedral differential inclusions
HyperTech / ADIODES [145]	HA with general continuous dynamics, over-approximation, interval numerical methods
VeriShift [70]	ellipsoidal over-approximation, $\dot{x}(t) = Ax(t) + u(t),\ x(t) \in I, u(t) \in U$
Coho [134]	reachability, two dimensional projections of polyhedra, non-linear, ordinary differential equations, linear programming, linear systems theory, local linearizations for each face of the projectahedron
Step [65]	deductive verification, clocked transition systems
CheckMate / MATLAB [265, 83]	polyhedral-invariant hybrid automata (PIHA), polyhedral set of initial continuous states, flow-pipe approximation procedure, ACTL, approximate quotient transition system (AQTS)
PHAVer / PPL [124]	safety properies, hybrid systems, exact arithmetic, piecewise constant bounds on the derivatives, over-approximation of piecewise affine dynamics
Maple / Qepcad[127]	Hybrid System, Symbolic Reachability, Abstraction
Requiem / Mathematica[218, 186]	nilpotent linear differential equation, quantifier elimination, exact reachability
Charon / Requiem [18, 6]	Specification language, agents and modes, hierarchical interacting hybrid systems
Maple / PVS [1]	computer algebra, automated theorem proving, symbolic computation problems
SAL [261, 60, 125, 103]	Theorem-prover, abstraction, intermediate language, LTL/CTL model checkers
SAL / MATLAB / Qepcad [125]	predicate abstraction, reachable set computation, quantifier elimination, hybrid automata, delta-notch
d/dt [36, 41]	orthogonal polyhedral approximation, reachability, HA with linear differential inclusions, numerical integration,Maximum Principle from optimal control
Tolque[222]	Semi-Algebraic Hybrid Automata, TCTL, AAMC

2. Uppaal [58, 59] is another popular tool for analyzing timed automata, with support for bounded integers, arrays and other variable types.

3. HyTech [16, 142, 146] was one of the early model-checkers to proceed beyond timed systems to hybrid automata where the continuous dynamics are defined by piecewise constant polyhedral differential inclusions. It thus expanded the domain of application of formal methods to interesting problems in digital controllers, schedulers and distributed algorithms.

4. HyperTech [145] extends HyTech by being able to handle hybrid automata with dynamics defined by non-linear differential equations of the form $dx_i/dt = f(x_1, ..., x_n)$, where f is a composition of polynomials, exponentials and trigonometric functions. The tool overcomes the pitfalls of traditional numerical integration by resorting to interval numerical methods (specifically, the ADIODES library), which over-approximate the result of integrating differential equation by a set of points guaranteed to include the true solution.

5. VeriShift [70] builds on the ellipsoidal over-approximation technique of Kurzhanski and Varaiya [180, 181, 183] and can handle differential equations of the form $\dot{x}(t) = Ax(t) + u(t),\ x(t) \in I, u(t) \in U$.

6. Coho [134] relies on a technique for representing high dimensional objects (non-convex polyhedra) by their projections onto the two dimensional subspaces ("projectahedron") defined by every pair of variables. At each time step of integration, ideas from computational geometry, linear programming and linear systems theory are used to keep track of the faces of each projection of the polyhedron. The tool also handles non-linear differential equations by approximate linearization along each face of each projectahedron.

7. Step [65] performs deductive verification over *clocked transition systems* with discrete and clock variables.

8. CheckMate [265, 83] uses MATLAB procedures to model-check ACTL queries over polyhedral-invariant hybrid automata (PIHA) where all invariants and guards are defined by linear inequalities, with the initial state also being polyhedral. The linear and non-linear terms in the continuous dynamics are approximated using appropriate flow-pipes with controllable error bounds. Interfacing with Simulink and Stateflow is supported. Model-checking is performed over the approximate quotient transition system (AQTS) extracted by conservative abstraction.

9. **PHAVer** (Polyhedral Hybrid Automaton Verifier) [124] is a tool for verifying hybrid systems with piecewise constant bounds on the derivatives, with affine dynamics being handled by overapproximation and state-space partitioning. The Parma Polyhedra Library (PPL) is used for performing robust exact arithmetic over non-convex polyhedra.

10. `d/dt` [36, 41] is a tool for analyzing hybrid systems with linear continuous dynamics of the form $dx/dt = Ax + u$, where u is input taking values in a bounded convex polyhedron U (i.e., linear differential inclusions). The invariants and guards are also assumed to be convex polyhedra. The polyhedral approximation, in combination with the Maximum Principle from optimal control, allows numerical integration based reachability verification.

11. SAL (Symbolic Analysis Laboratory) [261, 60, 125, 103] is a powerful framework providing tools for abstraction, program analysis, theorem proving an model checking. The tool-set was designed to abstract a continuous dynamical system

into a finite-state problem amenable to conventional model checking. Logical reasoning over the first-order theory of real closed fields is performed using quantifier elimination based on cylindrical algebraic decomposition. The set of polynomials that appear in the description of the continuous and discrete dynamics and their higher order derivatives are used to identify partitions of the original state space that are sign-invariant to all polynomials in the set. An intermediate language for describing transition systems is also defined. SAL supports many features of PVS, including model-checkers for LTL and CTL.

12. Ghosh et al. [125] used MATLAB to automate the process of modeling the Delta-Notch system in SAL, obtaining the abstract state corresponding to the hybrid automaton, performing symbolic backward reachability computation, and simplifying the solution using Qepcad.

13. Ghosh et al. [127] repeat their analysis using Maple and Qepcad, rather than MATLAB and SAL, by assuming piecewise affine continuous dynamics. An iterative refinement procedure is used to obtain the symbolic discrete abstraction, which is used to compute the under-approximate parametric backward reachable sets from the equilibria of the hybrid automaton.

14. Adams et al. [1] integrate the computer algebra system Maple with PVS and provide a new approach to symbolic computation problems.

15. Requiem [218] is a Mathematica notebook for reachability computation, based on the characterization of differential equations amenable to exact symbolic analysis [186]. It uses the quantifier elimination package in Mathematica.

16. Charon [18, 6] is a language for describing hierarchical concurrent hybrid systems, characterized using agents and modes. The continuous evolution can be

described using differential constraints, algebraic constraints or invariants. Requiem is used for reachability analysis.

9.3 A Case Study: The Delta-Notch Protein Signaling

Here we examine the Delta-Notch protein interaction system, the primary basis of biological pattern formation (see *Section* 3.4.1 for details). Ghosh et al. [126, 155] analyzed a simplified piece-wise linear hybrid automaton model (derived from the work of Collier et al. [89]) with the following properties: (1) The Delta (concentration v_D) production is turned on by low Notch concentration (v_N) in the same cell, i.e., when $-v_N > h_D$; (2) The Notch production is turned on by high Delta concentration in the cell environment (neighbors), i.e., when $\Sigma_i u_D^i > h_N$. Here, h_D and h_N are the thresholds, and u_D^i denotes the Delta concentration in each (i-th) neighbor.

In this section, we show how some interesting properties of the one-cell and two-cell Delta-Notch model of Ghosh et al. [126, 155] can be formulated as temporal logic queries, that Tolque can answer. Unfortunately, Qepcad cannot support the queries necessary to analyze system properties more complex than those documented here. Approximate methods (such as those discussed in AAMC-III [221]), reduction in the computational complexity of quantifier elimination, and greater computing power will help overcome this computational bottleneck. Rather than providing new insights[1] about the model, at this point, Tolque is only seen to support a more elegant and general way of thinking about system properties.

[1] A summary of the Ghosh-Tomlin analysis of this Delta-Notch pathway is presented in *Sec.* 9.3.3

9.3.1 One-Cell Delta-Notch Analysis in **Tolque**

In the hybrid automaton modeling the one-cell system [126], there are 2 dynamic variables v_D and v_N corresponding to the Delta and Notch concentration in the cell, 4 discrete states corresponding to the 2×2 possibilities resulting from Delta and Notch production being switched "on" or "off". The external variable u_N is assumed to be static. We will denote the upper bound on the continuous time-step by Δ.

The one-cell Delta-Notch hybrid automaton is depicted in *Figure* 9.1

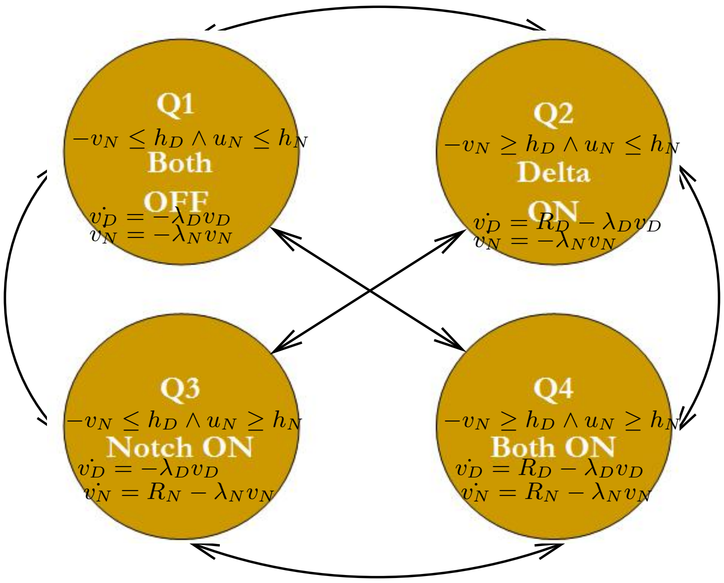

Figure 9.1: One-Cell Delta-Notch Hybrid Automaton

Note 9.3.1 *Since we model the external delta concentration* $u_N = \Sigma_i u_D^i$ *as a parameter and not a dynamic variable, the hybrid automaton naturally assumes these external variables do not change in the course of its evolution. Transitions are possible only between states* 1 *and* 2*, and states* 3 *and* 4 *because* u_N*'s relation to* h_N *cannot change in the course of the evolution of this one-cell hybrid system. Also, strong inequalities are sometimes changed to weak inequalities to correctly capture the*

Delta-Notch dynamics as per the standard semantics of hybrid automata.

Similarly, for simplification, we replace all inequalities with their weaker forms, i.e., $<$ to $\leq$ and $>$ to $\geq$. This is just to allow computation of the discrete state transitions using the simplest transition relation possible. For example, if the invariant of a state is $x \leq 5$ and the jump condition to another state is $x > 5$, the standard hybrid system semantics do not allow the system to ever take the jump because it will have to violate the state invariant for a time > 0. However, if the state invariant is $x < 5$ and the jump condition is $x \geq 5$ then the standard semantics allow the jump because the state invariant is violated for zero time, i.e., in one instant when $x = 5$ it violates the invariant - but it immediately jumps to the other state (in zero time). Hence its value does not violate its current (new) state's invariant. In the one-cell Delta-Notch model, we encounter this problem for example when we want to jump from state 4 ($-v_N \leq h_D$) to state 3 ($-v_N > h_D$). It will never be allowed according to the standard semantics. To overcome this "technical" problem, we change all invariants to the $\leq$ and $\geq$ forms.

1. **Pruned Transition Map** Only 8 out of the 12 discrete state transitions are mathematically possible [126]. This inference can be automated using the following query: for each discrete transition from state i to state j, ask if there exists a path where the invariant of state i is true until the invariant of state j becomes true.

   Definition 9.3.1 Jump-Possibility Query *When the state invariants are non-overlapping, an evolution path from discrete state i to j is possible iff $Inv_i \wedge \{Inv_i \; \exists\mathcal{U} \; Inv_j\}$.*

   Notice that invariants can be made non-overlapping by introducing a new environmental variable "discrete-state" that is reset to the destination discrete state number during discrete state transitions, with flow always 0. For the four transitions that are not possible, Tolque correctly replies that the query is *False*.

- **Discrete Transition** $1 \xrightarrow[\mathcal{D}]{0} 2$
  $[-v_N \leq h_D \wedge u_N \leq h_N]\ \exists\mathcal{U}\ [-v_N \geq h_D \wedge u_N \leq h_N]$
  After k iterations, we get the requirement $v_N \leq -h_D/(1-\Delta l_N)^k$ which is *True* when $k\ (\geq -\log(h_D/v_N)/\log(1-\Delta l_N))$. Thus the transition from 1 to 2 is possible.

- **Discrete Transition** $2 \xrightarrow[\mathcal{D}]{0} 1$
  $[-v_N \geq h_D \wedge u_N \leq h_N]\ \exists\mathcal{U}\ [-v_N < h_D \wedge u_N < h_N]$ converges after two iterations to *False*. Thus, it is not possible to jump to state 1 from state 2.

2. **Estimating Continuous-State Equilibrium Concentrations** The equilibrium values are computed in [126] by setting the *derivatives* to 0, solving for the equilibrium concentrations and substituting them in the invariants to see if they are satisfiable. In Tolque, this process can be automated by asking what needs to be initially true for no path to exist along which the values of the dynamic variables change. The fact that the state invariant has to hold at the starting point will eliminate the other terms resulting from the negation.

   When the state invariants are non-overlapping, an equilibrium of the continuous state exists in state i iff $Inv_i\ \wedge\ \neg\{Inv_i\ \exists\mathcal{U}\ (v'_D \neq v_D\ \vee\ v'_N \neq v_N)\}$, where v'_D and v'_N are the values after one step of the hybrid automaton.

Remark 9.3.1 *We have extended the TCTL notation to allow more complex temporal queries that can describe the values of the variables before and after* *one step of evolution. The semi-algebraic quantifier elimination based model-checking supports these queries without* any *additional work.*

State 1: $\neg\{[-v_N \leq h_D \wedge u_N \leq h_N] \; \exists\mathcal{U} \; [v'_D \neq v_D \vee v'_N \neq v_N]\}$ *converges to* False – *implying the non-existence of an equilibrium in this state.*

State 2: $\neg\{[-v_N \geq h_D \wedge u_N \leq h_N] \; \exists\mathcal{U} \; [v'_D \neq v_D \vee v'_N \neq v_N]\}$ *converges to* $v_D l_D - r_D = 0 \wedge v_N \leq 0$. *Thus we get the equilibrium concentrations as* $v_D^* = r_D/l_D, v_N^* = 0$.

State 3: $\neg\{[-v_N \leq h_D \wedge u_N \geq h_N] \; \exists\mathcal{U} \; [v'_D \neq v_D \vee v'_N \neq v_N]\}$ *converges to* $v_D \leq 0 \wedge v_N l_N - r_N = 0$. *Thus* $v_D^* = 0, v_N^* = r_N/l_N$ *are the equilibrium values.*

State 4: $\neg\{[-v_N \geq h_D \wedge u_N \geq h_N] \; \exists\mathcal{U} \; [v'_D \neq v_D \vee v'_N \neq v_N]\}$ *converges to the equilibrium condition* $v_N^* l_N - r_N = 0 \wedge h_D + v_N^* \neq 0 \wedge v_D^* l_D - r_D = 0 \wedge h_N - u_N \neq 0$.

3. **Discrete State Equilibria** When the invariants are non-overlapping, a system can stay forever in the discrete state i iff $Inv_i \wedge \neg \{Inv_i \; \exists\mathcal{U} \; \neg Inv_i\}$.

State 1: $[-v_N \leq h_D \wedge u_N \leq h_N] \; \exists\mathcal{U} \; [-v_N > h_D \vee u_N > h_N]$ returns $v_N \leq -h_D/(1-\Delta l_N)^k$ after k iterations, effectively evaluating to *True*. Thus the system always evolves out of state 1 and hence it does not correspond to any equilibrium.

State 2: $[-v_N \geq h_D \wedge u_N \leq h_N] \; \exists\mathcal{U} \; [-v_N < h_D \vee u_N > h_N]$ converges to *False*. Thus there is no path out of state 2 and hence it corresponds to an equilibrium. Note that the transition from 2 to 4 recorded in [126] is not possible in a one-cell model where u_N is not modeled as a dynamic variable.

State 3: $[-v_N \leq h_D \wedge u_N \geq h_N]$ $\exists\mathcal{U}$ $[-v_N < h_D \vee u_N > h_N]$ is non-convergent and returns $v_N \leq (-h_D - \Delta r_N)/(1 - \Delta l_N)$ after one iteration. So, for such a path out of state 3 to not exist, there should be no way of satisfying the above inequality when $-v_N < h_D$. So we get $(-h_D - \Delta r_N)/(1 - \Delta l_N) < -h_D$ which simplifies to $h_D > -r_N/l_N$.

State 4: $[-v_N \geq h_D \wedge u_N \geq h_N]$ $\exists\mathcal{U}$ $[-v_N < h_D \vee u_N < h_N]$ is non-convergent and returns $l_N h_D + r_N > 0 \wedge h_D - \Delta v_N l_N + \Delta r_N + v_N \geq 0$ after the second iteration. The second term is just a lower bound on the starting value of v_N which continues to drop with each iteration - effectively being *True*. Hence, for an equilibrium to exist in State 4, the first term must not be satisfiable, i.e., $l_N h_D + r_N \leq 0$ which is equivalent to $h_D \leq -r_N/l_N$.

4. **Backward Reachability From An Equilibrium State** We could now find the initial conditions that force the system to choose between the equilibria, by asking if there is a path of evolution that ends in the equilibrium state of interest.

Definition 9.3.2 State Reachability Query *When the invariants are non-overlapping, the initial conditions that lead to a discrete state i are given by:*

True $\exists\mathcal{U}$ Inv_i

Reaching:

State 2 *True* $\exists\mathcal{U}$ $[-v_N \geq h_D \wedge u_N \leq h_N]$ is non-convergent and returns $v_N \leq -h_D/(1-\Delta 1_N)^k \wedge h_N - u_N \geq 0$ after k iterations. The first term is the same slowly increasing upper bound on the initial Notch concentration and is effectively *True*. Hence we get the initial condition $u_N \leq h_N$ necessary to force this equilibrium.

State 3 After 2 (non-convergent) iterations of *True* $\exists\mathcal{U}$ $[-v_N \leq h_D \wedge u_N \geq h_N]$ we get $h_N - u_N \leq 0 \wedge [h_D + \Delta^2 v_N l_N^2 - \Delta^2 r_N l_N - 2\Delta v_N l_N + 2\Delta r_N + v_N \geq 0 \vee h_D + v_N \geq 0 \vee h_D - \Delta v_N l_N + \Delta r_N + v_N \geq 0 \vee [h_D + \Delta^2 v_N l_N^2 - 2\Delta v_N l_N + \Delta r_N + v_N \geq 0 \wedge h_N - u_N = 0]]$. The second term in the conjunction which is non-convergent, lists all the possible initial conditions that can lead to $-v_N \leq h_D$ and can be seen to be eventually *True* leaving just the first term $u_N \geq h_N$.

9.3.2 Two-Cell Delta-Notch Analysis in Tolque

The above exercise can be repeated for a two cell model, where there are 4 dynamic variables n_1, d_1, n_2 and d_2, which stand for the Notch and Delta concentrations in cell 1 and 2 respectively. Due to the limitations of Qepcad, we use the numerical parameter values courtesy Hwang et al. [155] to demonstrate our approach. In particular, we set $\lambda_N = \lambda_D = r_N = r_D = 1, h_D = -\frac{1}{2}, h_N = \frac{1}{5}, \Delta = \frac{1}{2}$. In other words

- *Cell-1* produces Delta when $n_1 < \frac{1}{2}$ and Notch when $d_2 > \frac{1}{5}$
- *Cell-2* produces Delta when $n_2 < \frac{1}{2}$ and Notch when $d_1 > \frac{1}{5}$

The two-cell Delta-Notch hybrid automaton is depicted in *Figure* 9.2

1. **Equilibrium Concentration Estimation** Again, we ask if there exists some path out of the state i where the variables change. The negation gives the condition for the system to remain in the same continuous state forever.

   State q_{10} (3,2): $\neg\{[-2n_1 > -1 \wedge 5d_2 < 1 \wedge -2n_2 < -1 \wedge 5d_1 > 1]\exists\mathcal{U}[d_1' \neq$

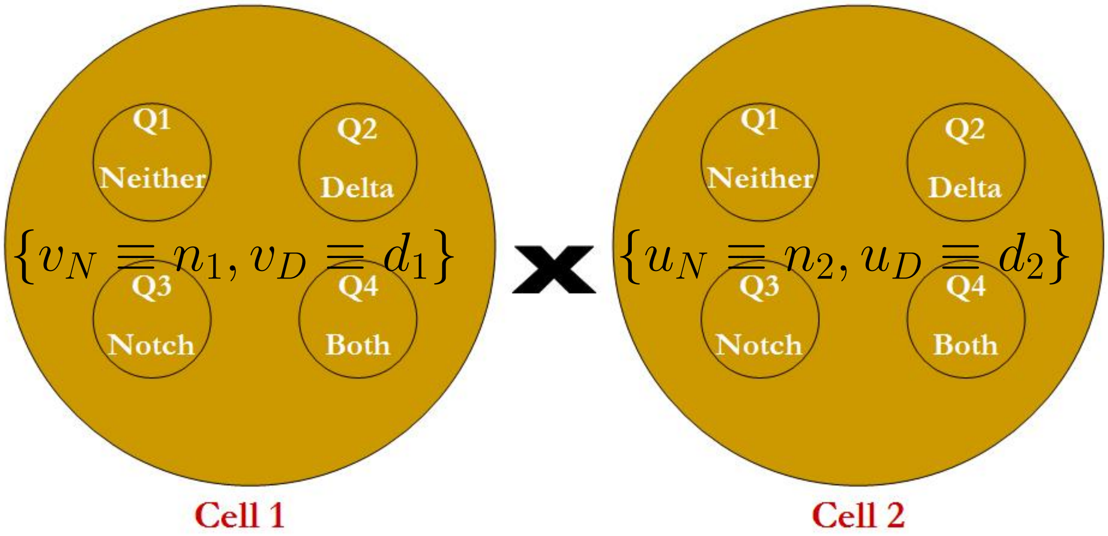

Figure 9.2: Two-Cell Delta-Notch Hybrid Automaton

$d_1 \vee n_1' \neq n_1 \vee d_2' \neq d_2 \vee n_2' \neq n_2]\}$ converges to $[n_1 \leq 0 \wedge d_2 \leq 0 \wedge d_1 - 1 = 0 \wedge n_2 - 1 = 0]$. Thus $n_1^* = d_2^* = 0$ and $d_1^* = n_2^* = 1$.

State q_7 **(2,3):** $\neg\{[-2n_1 < -1 \wedge 5d_2 > 1 \wedge -2n_2 > -1 \wedge 5d_1 < 1]\exists\mathcal{U}[d_1' \neq d_1 \vee n_1' \neq n_1 \vee d_2' \neq d_2 \vee n_2' \neq n_2]\}$ converges to $[n_2 \leq 0 \wedge d_1 \leq 0 \wedge d_2 - 1 = 0 \wedge n_1 - 1 = 0]$. Thus $n_2^* = d_1^* = 0$ and $d_2^* = n_1^* = 1$.

State q_{15} **(4,3):** $\neg\{[-2n_1 > -1 \wedge 5d_2 > 1 \wedge -2n_2 < -1 \wedge 5d_1 > 1]\exists\mathcal{U}[d_1' \neq d_1 \vee n_1' \neq n_1 \vee d_2' \neq d_2 \vee n_2' \neq n_2]\}$ converges to *False*, implying that in this discrete state the variables can never be in equilibrium.

2. **Are Equilibria Reversible?** We can find out if the system can ever leave an equilibrium state by asking of there is a path from its inside to its outside. We assume a numerical value of 0.5 for Δ, the upper-bound on the continuous time-step.

State q_7 **(2,3):** $[-2n_1 > -1 \wedge 5d_2 < 1 \wedge -2n_2 < -1 \wedge 5d_1 > 1]\ \exists\mathcal{U}\ [-2n_1 =$

$-1 \vee 5d_2 = 1 \vee -2n_2 = -1 \vee 5d_1 = 1]$ converges to *False* after 2 iterations implying that this is an irreversible discrete state equilibrium.

State q_{10} **(3,2):** $[-2n_1 < -1 \wedge 5d_2 > 1 \wedge -2n_2 > -1 \wedge 5d_1 < 1]\ \exists\mathcal{U}\ [-2n_1 = -1 \vee 5d_2 = 1 \vee -2n_2 = -1 \vee 5d_1 = 1]$ also converges to *False* after 2 iterations implying that the equilibrium is irreversible.

State q_{16} **(4,4):** $[-2n_1 > -1 \wedge 5d_2 > 1 \wedge -2n_2 > -1 \wedge 5d_1 > 1]\ \exists\mathcal{U}\ [-2n_1 = -1 \vee 5d_2 = 1 \vee -2n_2 = -1 \vee 5d_1 = 1]$ converges to *True* implying that the two-cell Delta-Notch system will always leave this discrete state.

3. **Choice Of Equilibrium** Now, suppose, we wish to find out what initial condition can lead to one equilibrium or the other. While we should ideally ask what initial condition always guarantees a specific equilibrium ($\forall\mathcal{U}$), we compute only $\exists\mathcal{U}$ and then check if the initial conditions overlap. This is because Qepcad is not able to proceed beyond 2 iterations. We can "verify" that the wrong equilibrium cannot be reached from a given initial relation between n_1 and n_2, and d_1 and d_2. When the invariants are non-overlapping, the initial conditions that allow a path to discrete state i but not to discrete state j are given by $\{\mathit{True}\ \exists\mathcal{U}\ \mathit{Inv}_i\} \wedge \neg\{\mathit{True}\ \exists\mathcal{U}\ \mathit{Inv}_j\}$.

State q_7 **(2,3):** At iteration 2 of *True* $\exists\mathcal{U}\ [-2n_1 > -1 \wedge 5d_2 < 1 \wedge -2n_2 < -1 \wedge 5d_1 > 1]$, we get: $n_1 - 1 \leq 0 \wedge [[2n_1 - 5d_1 \leq 0 \wedge 5d_2 - 1 \leq 0 \wedge 8n_2 - 5d_2 - 3 \geq 0 \wedge n_2 + n_1 - 1 = 0] \vee [8n_1 - 5d_1 - 3 \leq 0 \wedge 4d_2 + d_1 - 1 = 0 \wedge 2n_2 - 1 \geq 0 \wedge 8n_2 + 5d_1 - 5 \geq 0] \vee [5d_1 - 1 \geq 0 \wedge 2n_1 - 5d_1 \leq 0 \wedge 5d_2 + 2n_1 - 2 \leq 0 \wedge 2n_2 - 1 \geq 0] \vee [5d_1 - 1 \geq 0 \wedge 2n_1 - 1 \leq 0 \wedge 5d_2 - 1 \leq 0 \wedge 8n_2 - 5d_2 - 3 \geq 0] \vee [2n_1 - 1 \leq 0 \wedge 5d_2 - 1 \leq 0 \wedge 8n_2 - 5d_2 - 3 \geq 0 \wedge 8n_2 + 5d_1 - 5 \geq 0] \vee [2n_1 - 5d_1 \leq 0 \wedge 5d_2 - 1 \leq 0 \wedge 2n_2 - 1 \geq 0 \wedge 8n_2 + 5d_1 - 5 \geq 0]] \equiv f_7$.

Iteration 1 says "to get inside the state, you must first reach the border", while iteration 2 then lists the different states from where one could reach those bordering values.

State q_{10} (3,2): At iteration 2 of *True* $\exists\mathcal{U}$ $[-2n_1 < -1 \wedge 5d_2 > 1 \wedge -2n_2 > -1 \wedge 5d_1 < 1]$, we get: $n_2 - 1 \leq 0 \wedge [[2n_1 - 1 \geq 0 \wedge 5d_2 + 8n_1 - 5 \geq 0 \wedge d_2 + 4d_1 - 1 = 0 \wedge 2n_2 + 5d_1 - 2 \leq 0] \vee [2n_1 - 1 < 0 \wedge 8n_1 - 5d_1 - 3 \geq 0 \wedge 5d_2 + 8n_1 - 5 \geq 0 \wedge n_2 + n_1 - 1 = 0] \vee [8n_1 - 5d_1 - 3 \geq 0 \wedge 5d_2 + 8n_1 - 5 < 0 \wedge 5d_2 + 2n_1 - 2 \geq 0 \wedge n_2 + n_1 - 1 = 0] \vee [2n_1 - 1 \geq 0 \wedge 5d_2 - 1 \geq 0 \wedge 2n_2 + 5d_1 - 2 \leq 0 \wedge n_2 + n_1 - 1 < 0] \vee [5d_1 - 1 \leq 0 \wedge 2n_1 - 1 \geq 0 \wedge 5d_2 + 8n_1 - 5 \geq 0 \wedge 2n_2 - 5d_2 \leq 0] \vee [5d_1 - 1 \leq 0 \wedge 2n_1 - 1 \geq 0 \wedge 5d_2 + 8n_1 - 5 \geq 0 \wedge 2n_2 - 1 \leq 0] \vee [8n_1 - 5d_1 - 3 \geq 0 \wedge 5d_2 - 1 \geq 0 \wedge 2n_2 + 5d_1 - 2 \leq 0 \wedge 2n_2 - 1 \leq 0]] \equiv f_{10}$.

State q_7 and not State q_{10}: The initial conditions that lead only to q_7 and not q_{10} are thus given by $f_7 \wedge \neg f_{10} = n_1 - 1 \leq 0 \wedge [[2n_1 - 5d_1 \leq 0 \wedge 5d_2 - 1 < 0 \wedge 8n_2 - 5d_2 - 3 \geq 0 \wedge n_2 + n_1 - 1 = 0] \vee [2n_1 - 1 \leq 0 \wedge 5d_2 - 1 \leq 0 \wedge 8n_2 - 5d_2 - 3 \geq 0 \wedge 2n_2 + 5d_1 - 2 > 0] \vee [2n_1 - 1 \leq 0 \wedge 5d_2 + 2n_1 - 2 \leq 0 \wedge 4d_2 + d_1 - 1 = 0 \wedge n_2 + n_1 - 1 > 0] \vee [2n_1 - 5d_1 \leq 0 \wedge 5d_2 - 1 \leq 0 \wedge n_2 + n_1 - 1 > 0 \wedge 2n_2 - 1 \geq 0] \vee [2n_1 - 1 \leq 0 \wedge 5d_2 - 1 < 0 \wedge 8n_2 - 5d_2 - 3 \geq 0 \wedge 8n_2 + 5d_1 - 5 \geq 0] \vee [8n_1 - 5d_1 - 3 < 0 \wedge 4d_2 + d_1 - 1 = 0 \wedge 2n_2 - 1 \geq 0 \wedge 8n_2 + 5d_1 - 5 \geq 0] \vee [5d_1 - 1 \geq 0 \wedge 2n_1 - 5d_1 < 0 \wedge 5d_2 + 2n_1 - 2 \leq 0 \wedge 2n_2 - 1 \geq 0] \vee [5d_1 - 1 \geq 0 \wedge 2n_1 - 1 \leq 0 \wedge 5d_2 - 1 < 0 \wedge 8n_2 - 5d_2 - 3 \geq 0] \vee [2n_1 - 1 < 0 \wedge 5d_2 - 1 \leq 0 \wedge 8n_2 - 5d_2 - 3 \geq 0 \wedge 8n_2 + 5d_1 - 5 \geq 0]]$

Since we have assumed no upper bound on the initial values and since we have been able to compute only two iterations, this formula does *not* evaluate to *True* given the correct initial partition $n_1 < n_2 \wedge d_1 > d_2$. However, when Qepcad simplifies the above formula assuming that $n_1 >$

$n_2 \wedge d_1 < d_2$, it evaluates to *False*.

State q_{10} and not State q_7 Similarly, $\neg f_7 \wedge f_{10} = n_2 - 1 \leq 0 \wedge [[5d_1 - 1 \leq 0 \wedge 8n_1 - 5d_1 - 3 > 0 \wedge 5d_2 + 8n_1 - 5 \geq 0 \wedge 2n_2 - 1 \leq 0] \vee [8n_1 - 5d_1 - 3 > 0 \wedge 5d_2 - 1 \geq 0 \wedge n_2 + n_1 - 1 = 0 \wedge 2n_2 - 5d_2 = 0] \vee [8n_1 - 5d_1 - 3 \geq 0 \wedge 5d_2 - 1 \geq 0 \wedge 2n_2 - 1 < 0 \wedge 2n_2 + 5d_1 - 2 \leq 0] \vee [2n_1 - 1 \geq 0 \wedge 5d_2 + 8n_1 - 5 > 0 \wedge d_2 + 4d_1 - 1 = 0 \wedge 2n_2 + 5d_1 - 2 \leq 0] \vee [2n_1 - 1 \geq 0 \wedge 5d_2 + 8n_1 - 5 \geq 0 \wedge d_2 + 4d_1 - 1 = 0 \wedge n_2 + n_1 - 1 < 0] \vee [2n_1 - 1 \geq 0 \wedge 5d_2 - 1 \geq 0 \wedge n_2 + n_1 - 1 < 0 \wedge 2n_2 + 5d_1 - 2 \leq 0] \vee [8n_1 - 5d_1 - 3 \geq 0 \wedge 2n_1 - 1 < 0 \wedge 5d_2 + 2n_1 - 2 > 0 \wedge n_2 + n_1 - 1 = 0] \vee [5d_1 - 1 \leq 0 \wedge 2n_1 - 1 \geq 0 \wedge 5d_2 - 1 > 0 \wedge 2n_2 - 5d_2 \leq 0] \vee [8n_1 - 5d_1 - 3 \geq 0 \wedge 5d_2 - 1 \geq 0 \wedge 2n_2 - 1 \leq 0 \wedge 2n_2 + 5d_1 - 2 < 0]]$, evaluates to *False* assuming $n_1 < n_2 \wedge d_1 > d_2$. This concurs with the result of Ghosh et al. [125].

9.3.3 Summary of other Efforts

For the sake of completeness, the series of symbolic analyses of the Delta-Notch system conducted by Ghosh and Tomlin, which clearly outperforms Tolque, is summarized here from [127].

One-Cell Delta-Notch Automaton

Representing Delta and Notch concentrations as x_1 and x_2 respectively, a three-state abstract state space was derived:

$$\begin{aligned}
q_1 : Inv(q_1) &= x_1 > 0 \wedge x_2 + h_D > 0 \ \textit{and} \ q_1 \rightarrow q_2 \\
q_2 : Inv(q_2) &= x_1 > 0 \wedge x_2 + h_D = 0 \ \textit{and} \ q_2 \rightarrow q_3 \\
q_3 : Inv(q_3) &= x_1 > 0 \wedge x_2 + h_D < 0 \ \textit{and} \ q_3 \rightarrow \phi : \textit{contains equilibrium}
\end{aligned}$$

From backward reachability computation of the equilibrium state q_3 ($Reach(q_3) = q_1 \cup q_2 \cup q_3$), there were able to conclude that an isolated cell will adopt the "Delta" fate over the "Notch" fate.

Two-Cell Delta-Notch Automaton

In addition to x_1 and x_2, in the second cell x_3 and x_4 are now used to denote the Delta and Notch concentration respectively. The equilibria and their backward reachability states were characterized thus:

1. **Equilibrium 1:** $(x_1 = 0, x_2 = \frac{R_N}{\lambda_N}, x_3 = \frac{R_D}{\lambda_D}, x_4 = 0)$

   $(x_3 - x_1 \geq 0 \wedge x_4 - x_2 \leq 0 \wedge ((x_3 - x_1 > 0 \wedge h_D + x_4 \geq 0) \vee (x_3 - x_1 > 0 \wedge h_D + x_2 \leq 0) \vee (x_4 - x_2 < 0 \wedge h_N - x_3 \geq 0) \vee (x_4 - x_2 < 0 \wedge h_N - x_1 \leq 0))) \vee (x_3 - x_1 < 0 \wedge x_4 - x_2 < 0 \wedge ((-x_2 - h_D < 0 \wedge x_1 - h_N \leq 0 \wedge -x_4 - h_D > 0 \wedge (1 - \frac{\lambda_D}{R_D}(x_3 - x_1))^{\lambda_N} \leq (\frac{x_2}{-h_D})^{\lambda_D}) \vee (-x_2 - h_D \leq 0 \wedge x_3 - h_N \geq 0 \wedge -x_4 - h_D > 0 \wedge (1 - \frac{\lambda_D}{R_D}(x_3 - x_1))^{\lambda_N} \leq (\frac{R_N - \lambda_N x_4}{R_N + \lambda_N h_D})^{\lambda_D}))) \vee (x_3 - x_1 > 0 \wedge x_4 - x_2 > 0 \wedge ((-x_2 - h_D < 0 \wedge x_3 - h_N > 0 \wedge x_1 - h_N \leq 0 \wedge x_4 - x_2 \leq \frac{-R_N}{\lambda_N}(1 - (\frac{x_3}{h_N})^{\frac{\lambda_N}{\lambda_D}})) \vee (x_3 - h_N \geq 0 \wedge -x_4 - h_D \geq 0 \wedge x_1 - h_N < 0 \wedge x_4 - x_2 \leq \frac{-R_N}{\lambda_N}(1 - (\frac{x_1}{h_N})^{\frac{\lambda_N}{\lambda_D}})))$

2. **Equilibrium 2:** $(x_1 = \frac{R_D}{\lambda_D}, x_2 = 0, x_3 = 0, x_4 = R_N \lambda_N)$

   $(x_3 - x_1 \leq 0 \wedge x_4 - x_2 \geq 0 \wedge ((x_3 - x_1 < 0 \wedge h_D + x_2 \geq 0) \vee (x_3 - x_1 < 0 \wedge h_D + x_4 \leq 0) \vee (x_4 - x_2 > 0 \wedge h_N - x_1 \geq 0) \vee (x_4 - x_2 > 0 \wedge h_N - x_3 \leq 0))) \vee (x_3 - x_1 < 0 \wedge x_4 - x_2 < 0 \wedge ((x_3 - h_N \leq 0 \wedge -x_4 - h_D < 0 \wedge x_1 - h_N > 0 \wedge (1 - \frac{\lambda_N}{R_N}(x_4 - x_2))^{\lambda_D} \leq (\frac{x_1}{h_N})^{\lambda_N}) \vee (-x_2 - h_D \geq 0 \wedge x_3 - h_N < 0 \wedge x_1 - h_N \geq 0 \wedge (1 - \frac{\lambda_N}{R_N}(x_4 - x_2))^{\lambda_D} \leq (\frac{R_D - \lambda_D x_3}{R_D - \lambda_D h_N})^{\lambda_N}))) \vee (x_3 - x_1 > 0 \wedge x_4 - x_2 > 0 \wedge ((-x_2 - h_D \geq 0 \wedge x_3 - h_N < 0 \wedge x_1 - h_N < 0 \wedge x_3 - x_1 \leq \frac{-R_D}{\lambda_D}(1 - (\frac{x_4}{-h_D})^{\frac{\lambda_D}{\lambda_N}})) \vee (x_3 - h_N > 0 \wedge -x_4 - h_D \leq 0 \wedge x_1 - h_N \geq 0 \wedge x_3 - x_1 \leq \frac{-R_D}{\lambda_D}(1 - (\frac{x_2}{-h_D})^{\frac{\lambda_D}{\lambda_N}})))$

Thus, they were able to prove that the two-cell system adopts cell-fate dictated by the amplification of the initial differences in Delta and Notch concentrations, except when certain semi-algebraic relations hold between the various symbolic parameters.

Four-Cell Delta-Notch Automaton

Adding four more variables, $x_1, \cdots, x_8$ now represent the Delta and Notch concentrations in the four cells, which now have three possible equilibria:

1. **Equilibrium 1:** $(x_1 = 0, x_2 = \frac{R_N}{\lambda_N}, x_3 = 0, x_4 = \frac{R_N}{\lambda_N}, x_5 = \frac{R_D}{\lambda_D}, x_6 = 0, x_7 = 0, x_8 = \frac{R_N}{\lambda_N})$

   $(h_D + x_6 \leq 0 \wedge h_D + x_4 \geq 0 \wedge h_D + x_2 \geq 0 \wedge h_D + x_8 \geq 0 \wedge h_N - x_7 - x_5 - x_1 \leq 0 \wedge h_N - x_5 - x_3 \leq 0 \wedge h_N - x_7 - x_3 - x_1 \geq 0 \wedge ((x_5 - x_3 > 0 \wedge x_7 - x_3 + x_1 \leq 0 \wedge h_D + x_4 \leq 0 \wedge h_N - x_7 - x_3 - x_1 > 0) \vee (x_5 - x_3 > 0 \wedge x_7 - x_3 + x_1 \leq 0 \wedge h_D + x_6 \geq 0 \wedge h_N - x_7 - x_3 - x_1 > 0) \vee (x_5 - x_3 > 0 \wedge x_7 - x_3 + x_1 \leq 0 \wedge h_N - x_7 - x_5 - x_1 \geq 0) \vee (x_5 - x_3 > 0 \wedge x_7 - x_3 + x_1 \leq 0 \wedge h_D + x_8 \leq 0 \wedge h_N - x_7 - x_3 - x_1 > 0) \vee (h_D + x_2 \leq 0 \wedge h_D + x_8 > 0 \wedge h_N - x_7 - x_3 - x_1 > 0) \vee (x_7 - x_3 + x_1 \geq 0 \wedge x_7 - x_5 + x_1 < 0 \wedge h_D + x_4 \leq 0 \wedge h_N - x_7 - x_3 - x_1 > 0) \vee (x_7 - x_3 + x_1 \geq 0 \wedge x_7 - x_5 + x_1 < 0 \wedge h_D + x_6 \geq 0 \wedge h_N - x_7 - x_3 - x_1 > 0) \vee (x_7 - x_3 + x_1 \geq 0 \wedge x_7 - x_5 + x_1 < 0 \wedge h_N - x_5 - x_3 \geq 0) \vee (h_D + x_6 < 0 \wedge h_D + x_4 > 0 \wedge h_D + x_2 > 0 \wedge h_D + x_8 > 0 \wedge h_N - x_7 - x_3 - x_1 \leq 0) \vee (x_7 - x_3 + x_1 \geq 0 \wedge x_7 - x_5 + x_1 < 0 \wedge h_D + x_8 \leq 0 \wedge h_N - x_7 - x_3 - x_1 > 0)))$

2. **Equilibrium 2:** $(x_1 = 0, x_2 = \frac{R_N}{\lambda_N}, x_3 = \frac{R_D}{\lambda_D}, x_4 = 0, x_5 = 0, x_6 = \frac{R_N}{\lambda_N}, x_7 = 0, x_8 = \frac{R_N}{\lambda_N})$

   $(h_D + x_4 \leq 0 \wedge h_D + x_6 \geq 0 \wedge h_D + x_2 \geq 0 \wedge h_D + x_8 \geq 0 \wedge h_N - x_7 - x_5 - x_1 \geq 0 \wedge h_N - x_5 - x_3 \leq 0 \wedge h_N - x_7 - x_3 - x_1 \leq 0 \wedge ((x_5 - x_3 < 0 \wedge x_7 - x_5 + x_1 \leq 0 \wedge h_D + x_6 \leq 0 \wedge h_N - x_7 - x_5 - x_1 > 0) \vee (x_5 - x_3 < 0 \wedge x_7 - x_5 + x_1 \leq 0 \wedge h_D + x_4 \geq 0 \wedge h_N - x_7 - x_5 - x_1 > 0) \vee (x_5 - x_3 < 0 \wedge x_7 - x_5 + x_1 \leq 0 \wedge h_N - x_7 - x_3 - x_1 \geq 0) \vee (x_5 - x_3 < 0 \wedge x_7 - x_5 + x_1 \leq 0 \wedge h_D + x_8 \leq 0 \wedge h_N - x_7 - x_5 - x_1 > 0) \vee (h_D + x_4 < 0 \wedge h_D + x_2 \leq 0 \wedge h_D + x_8 > 0 \wedge h_N - x_7 - x_5 - x_1 > 0) \vee (x_7 - x_5 + x_1 \geq 0 \wedge x_7 - x_3 + x_1 < 0 \wedge h_D + x_6 \leq 0 \wedge h_N - x_7 - x_5 - x_1 > 0) \vee (x_7 - x_5 + x_1 \geq 0 \wedge x_7 - x_3 + x_1 < 0 \wedge h_D + x_4 \geq 0 \wedge h_N - x_7 - x_5 - x_1 > 0) \vee (x_7 - x_5 + x_1 \geq 0 \wedge x_7 - x_3 + x_1 < 0 \wedge h_N - x_5 - x_3 \geq 0) \vee (h_D + x_4 < 0 \wedge h_D + x_6 > 0 \wedge h_D + x_2 > 0 \wedge h_D + x_8 > 0 \wedge h_N - x_7 - x_5 - x_1 \leq 0) \vee (x_7 - x_5 + x_1 \geq 0 \wedge x_7 - x_3 + x_1 < 0 \wedge h_D + x_8 \leq 0 \wedge h_N - x_7 - x_5 - x_1 > 0)))$

3. **Equilibrium 3:** $(x_1 = \frac{R_D}{\lambda_D}, x_2 = 0, x_3 = 0, x_4 = \frac{R_N}{\lambda_N}, x_5 = 0, x_6 = \frac{R_N}{\lambda_N}, x_7 =$

$\frac{R_D}{\lambda_D}, x_8 = 0)$

$(h_D + x_4 \geq 0 \wedge h_D + x_6 \geq 0 \wedge h_N - x_5 - x_3 \geq 0 \wedge h_N - x_7 - x_5 - x_1 \leq 0 \wedge h_N - x_7 - x_3 - x_1 \leq 0 \wedge ((h_D + x_4 > 0 \wedge h_D + x_6 > 0 \wedge h_D + x_2 \geq 0 \wedge h_D + x_8 < 0) \vee (h_D + x_2 \leq 0 \wedge h_N - x_5 - x_3 > 0 \wedge h_N - x_7 - x_3 - x_1 \geq 0) \vee (h_D + x_2 \leq 0 \wedge h_N - x_5 - x_3 > 0 \wedge h_N - x_7 - x_5 - x_1 \geq 0) \vee (h_D + x_4 > 0 \wedge h_D + x_6 > 0 \wedge h_D + x_2 < 0 \wedge h_N - x_5 - x_3 \leq 0) \vee (x_7 - x_3 + x_1 > 0 \wedge x_7 - x_5 + x_1 > 0 \wedge h_D + x_4 \leq 0 \wedge h_D + x_8 \leq 0 \wedge h_N - x_5 - x_3 > 0) \vee (x_7 - x_3 + x_1 > 0 \wedge x_7 - x_5 + x_1 > 0 \wedge h_D + x_8 = 0 \wedge h_N - x_5 - x_3 > 0) \vee (x_7 - x_3 + x_1 > 0 \wedge x_7 - x_5 + x_1 > 0 \wedge h_D + x_6 \leq 0 \wedge h_D + x_8 \leq 0 \wedge h_N - x_5 - x_3 > 0) \vee (h_D + x_2 \leq 0 \wedge h_D + x_8 \geq 0 \wedge h_N - x_5 - x_3 > 0)))$

Based on this characterization, they were able to verify that the amplification of initial variations dominates cell-fate choice even in the four-cell system. However, the results are more complex in that the semi-algebraic relations that need to hold to guarantee a certain equilibrium are not easily interpretable as the product of 2 two-cell systems. The third equilibrium, where two cells adopt the Delta-fate as opposed to just one cell, was seen to have a larger backward reachable set.

9.4 Other Examples

9.4.1 One-State Harmonic Oscillator Example

Consider a basic harmonic oscillator defined by the equation $\ddot{x} = -x$. We can encode it as a *one-state* hybrid system in two variables u and v where:

$$\begin{aligned} u &= x \\ v &= \dot{u}(= dx/dt) \\ \dot{v} = \ddot{u} = \ddot{x} &= -x = -u \end{aligned}$$

In other words, $\mathbf{u} = \mathbf{x}, \mathbf{du/dt} = \mathbf{v}, \mathbf{dv/dt} = -\mathbf{u}$ and the solution is $x = Asin(t) + Bcos(t)$. Using $h = 1/10$, the transition relation is:

$$10u' - v - 10u = 0 \wedge 10v' - 10v + u = 0$$

To test the system, let us start with the *initial condition* $u = 0, v = 1$. This refers to the solution $x = Asin(t)$. Now we ask if u ever becomes < 0, i.e., $\mathbf{EF}u < 0$? We know that after π units of time, the *sinusoidal curve* becomes negative and hence the query should become true then. Indeed, after **31** iterations, $t = 1/10 * 31 = 3.1 < \pi$ and *result* is :

$u < 0 \vee$

$6048617925506247085134663964 99v - 1165190467951638865229623660531 0u < 0$.

Checked against initial conditions, we get:

$$0 < 0 \vee 604861792550624708513466396499 < 0 \equiv \mathit{False}.$$

After the **32**nd iteration, $t = 1/10 * 32 = 3.2 > \pi$ and *result* is :

$u \neq 0 \vee$

$560328675401014156716157264032 0v + 1171239085877145112314758324495 99u \neq 0$

Checked against initial conditions, we get:

$$0 \neq 0 \vee 560328675401014156716157264032 0 \neq 0 \equiv \mathit{True}.$$

9.4.2 The Repressilator Example

The repressilator is an artificial network of 6 proteins and their corresponding mRNAs (see 3.4.4 for details). Protein-1 represses protein-2 which represses protein-3, which in turn represses protein-1. More specifically, each protein suppresses the production of its target mRNA. As the mRNA concentration drops, the protein levels also drop.

The flow equations may be expressed as:

$$
\begin{aligned}
(1+P_3^2)(M_1' - M_1) &= ((\gamma - M_1)(1+P_3^2)+\alpha)h \\
(1+P_1^2)(M_2' - M_2) &= ((\gamma - M_2)(1+P_1^2)+\alpha)h \\
(1+P_2^2)(M_3' - M_3) &= ((\gamma - M_3)(1+P_2^2)+\alpha)h \\
P_1' &= P_1 - \beta(P_1 - M_1)h \\
P_2' &= P_2 - \beta(P_2 - M_2)h \\
P_3' &= P_3 - \beta(P_3 - M_3)h
\end{aligned}
$$

We further simplify our problem by assuming $\gamma = \alpha = 0, \beta = 1$. The initial conditions are: $[p_1 = 2] \wedge [p_2 = 2] \wedge [p_3 = 2] \wedge [m_1 = 2] \wedge [m_2 = 2] \wedge [m_3 = 2]$. We want to check if the concentration of p_1 never drops below 1. So we ask $EHp_1 > 1$.

The transition relation of this one state hybrid system is just their value as given by the discretization equations:
$10p_3^2m_1' + 10m_1' - 9m_1p_3^2 - \gamma p_3^2 - 9m_1 - \gamma - \alpha = 0 \wedge 10p_1^2m_2' + 10m_2' - 9m_2p_1^2 - \gamma p_1^2 - 9m_2 - \gamma - \alpha = 0 \wedge 10p_2^2m_3' + 10m_3' - 9m_3p_2^2 - \gamma p_2^2 - 9m_3 - \gamma - \alpha = 0 \wedge 10p_1' + \beta p_1 - 10p_1 - \beta m_1 = 0 \wedge 10p_2' + \beta p_2 - 10p_2 - \beta m_2 = 0 \wedge 10p_3' + \beta p_3 - 10p_3 - \beta m_3 = 0$

1. At time $t = 0$, we check if $P_1(0) > 1$ which is indeed true according to the initial condition $P_1 = 2$.

2. Evaluating query with future-length = 1 :
   $(\exists m_1')(\exists m_2')(\exists m_3')(\exists p_1')(\exists p_2')(\exists p_3')[[[p_3^2m_1' + m_1' - 50 = 0 \wedge p_1^2m_2' + m_2' - 50 = 0 \wedge p_2^2m_3' + m_3' - 50 = 0 \wedge p_1' + 4p_1 - 5m_1 = 0 \wedge p_2' + 4p_2 - 5m_2 = 0 \wedge p_3' + 4p_3 - 5m_3 = 0] \wedge [[p_1' - 1 > 0]]]], assume = [[m_1 >= 0] \wedge [m_2 >= 0] \wedge [m_3 >= 0] \wedge [p_1 >= 0] \wedge [p_2 >= 0] \wedge [p_3 >= 0] \wedge [1h - 1 = 0]]$
   $= 4p_1 - 5m_1 + 1 < 0$

$= [[[p_1 - 1 > 0]] \wedge [4p_1 - 5m_1 + 1 < 0]]$

Checked against initial conditions, $2 - 1 = 1 > 0 \wedge 8 - 10 + 1 = -1 < 0$ is also true.

3. Evaluating query with future-length $= 2$:

   $(\exists m_1')(\exists m_2')(\exists m_3')(\exists p_1')(\exists p_2')(\exists p_3')[[[p_3^2 m_1' + m_1' - 50 = 0 \wedge p_1^2 m_2' + m_2' - 50 = 0 \wedge p_2^2 m_3' + m_3' - 50 = 0 \wedge p_1' + 4p_1 - 5m_1 = 0 \wedge p_2' + 4p_2 - 5m_2 = 0 \wedge p_3' + 4p_3 - 5m_3 = 0] \wedge [[[[p_1' - 1 > 0]] \wedge [4p_1' - 5m_1' + 1 < 0]]]]]$

   $= [[[p_1 - 1 > 0]] \wedge [4p_1 - 5m_1 + 1 < 0 \wedge 16p_1p_3^2 - 20m_1p_3^2 - p_3^2 + 16p_1 - 20m_1 + 249 > 0]]$

   Checked against initial conditions, $2 - 1 = 1 > 0 \wedge 8 - 10 + 1 = -1 < 0 \wedge 128 - 160 - 4 + 32 - 40 + 249 = 205 > 0$ is also true.

4. Evaluating query with future-length $= 3$:

   $(\exists m_1')(\exists m_2')(\exists m_3')(\exists p_1')(\exists p_2')(\exists p_3')[[-\gamma p_3^2 - 9m_1p_3^2 + 10p_3^2 m_1' - \alpha - \gamma - 9m_1 + 10m_1' = 0] \wedge [-\gamma p_1^2 - 9m_2p_1^2 + 10p_1^2 m_2' - \alpha - \gamma - 9m_2 + 10m_2' = 0] \wedge [-\gamma p_2^2 - 9m_3p_2^2 + 10p_2^2 m_3' - \alpha - \gamma - 9m_3 + 10m_3' = 0] \wedge [-\beta m_1 + \beta p_1 - 10p_1 + 10p_1' = 0] \wedge [-\beta m_2 + \beta p_2 - 10p_2 + 10p_2' = 0] \wedge [-\beta m_3 + \beta p_3 - 10p_3 + 10p_3' = 0] \wedge [[5p_1' - 1 < 0] \vee [-\beta m_1' + \beta p_1' - 10p_1' + 2 > 0]]]$.

   Qepcad, because of computational limitations, cannot handle this query, and is not able to proceed.

9.5 Discussion

Under ordinary computational settings, Qepcad failed to support fully symbolic analysis of the two-cell Delta-Notch system and other interesting examples. However, it is to be noted that even this preliminary version of Tolque was able to support a very uniform way of asking about a good spectrum of interesting temporal properties of a biologically significant hybrid system. The performance on the harmonic oscillator example was promising, as oscillating biochemical networks are prevalent in nature. We are in the process of rewriting Tolque in *Lisp* and integrating it with Simpathica[33] (see [224] for a progress report). These modifications will allow biochemical networks to be easily represented, stored and analyzed in keeping with our "Systems Biology" motivation. Eventually, we plan to implement our own symbolic algebra system to work hand in hand with the different *quantifier elimination*, *Gröbner basis* and *characteristic set* tools that can systematically simplify the formulæ at each fixpoint iteration. Similarly, it is hoped that we can also build upon the efforts at parallelizing these computational algebra algorithms [254, 107, 207]. It is hoped that the possibility of such a powerful and universal tool will engender the development of new techniques and more efficient implementations of quantifier elimination and other algorithms from real algebraic geometry.

Chapter 10

Conclusion

Summary

Systems Biology was seen to be a very broad and challenging field attempting to hasten the understanding of life by integrating computational, mathematical, statistical, AI, machine learning and temporal reasoning techniques with tradition wet-lab experimentation. One important strand of research is modeling, simulation and analysis of biochemical pathways to evaluate mutual consistency of biological facts, to validate or falsify hypotheses, to aid experiment design and refine existing models. The fundamental problem in biochemical networks continues to be reasoning about the emergent temporal properties and their modulation by external signals. The direct detailed kinetic mass action based numerical analysis of biochemical networks is intractable because of the complexity of the network in terms of the number of interacting species, the number of interconnections and the different types of interactions. More importantly, it becomes extremely difficult to reason about the system-level properties of the entire network from the time-course analysis of its many molecular components. A related problem of tremendous interest to the pharmaceutical industry is the identification of mechanisms by which a biochemical network can be enticed to shift from one equilibrium (causing ill-health, or not producing an invasion sensor, etc.) to a target equilibrium (enhancing health, producing a protein that can detect cancer cells, etc.). Effectively, a strategy that a new drug or therapy should try to mimic can be discovered.

The computer science formalism *Hybrid Automaton* has been found to be very useful in modeling biological systems, especially at the subcellular level in terms of interactions of biochemical species. In this thesis, we improved our fundamental understanding of hybrid automata. We considered hybrid dynamical systems which are described by sets of ODEs, which govern the evolution of the system variables with time. One common approach to understanding how the system properties vary with time is by inspecting the answers (or the counter-examples) to temporal logic queries like "Is a certain unfavorable property ever satisfiable?" and "Is a certain essential property always satisfied?". When the system description involves symbolic parameters, deeper questions and richer answers become necessary; e.g., "For what ranges of parameters is a certain unfavorable property never satisfiable?", "Are there any parameters which can guarantee that a certain essential property will always be true?", etc.

Our first important result was proving that the open subclass HPCD is closer to the decidability and undecidability frontiers for the reachability query, than was previously characterized. Our most significant contribution, however, was the extension of Fränzle's ideas to construct the new *semi-algebraic hybrid automaton* subclass. It is a substantially expansive class of hybrid automata amenable to rigorous symbolic temporal analysis. Moreover, the fundamental mathematical machinery – real quantifier elimination, handles symbolic parameters in a very natural way. In a sense, we have partially answered the question: "Given the existence of the real algebraic geometry decision procedure quantifier elimination, what is the most complex temporal logic analysis one can perform?". After going from bounded reachability to an enhanced version of dense-time TCTL that can answer semi-algebraic queries, we also characterized the semi-decidability properties. The intractability of reachability was also proved in the real Turing Machine formalism. We developed many approximation strategies based on bisimulation partitioning, approximation with rectangular grids, approximation with polytopes and time discretization. On the Systems Biology side, we showed how the biochemistry problem description can be translated into the dynamical systems theory format, thus making the biochemical pathway models ready for the sort of analyses devised for semi-algebraic hybrid automata. We also contributed by fundamentally extending techniques like Flux Balance Analysis to the algebraic domain. On the software side, we were able to provide a glimpse into what could be eventually possible, by presenting the proof-of-concept tool Tolque. The double-exponential complexity of quantifier elimination continues to be the computational bottleneck of this approach.

Future Work

This thesis has opened up several interesting avenues for research. First of all, how can the new characterization of the HPCD class lead to better semi-decidable algorithms? Do chaotic dynamical systems have a say in the decidability of this class? The next and more fundamental question is: have we indeed characterized the most expansive temporal decision procedure based on real quantifier elimination? Can the semi-algebraic class be extended? What extensions are possible that will still yield a semi-decidable procedure? The characterization of decidable subclasses is an open problem. What are biochemically sound assumptions we can use to simplify the possible dynamics of semi-algebraic hybrid automata? Similarly, there do not exist efficient methods that convert a one-state dynamical system into a semi-algebraic hybrid automaton. Bisimulation partitioning may be one approach, but are there more general approaches that also utilize the properties valid for biochemical systems? On the software side, the obvious step is to integrate all available quantifier elimination tools, develop the necessary parallelizable algorithms, and utilize a cluster of machines of substantial power to see if practically useful results are extractable. Here again, we will face questions such as: What approximation schemes will work in practice? When is there a net gain?

Characterizing recursive paths and system invariants, extending concepts from BMC to the algebraic domain, deriving assumptions that will guarantee fixpoint convergence and extension of Cousot's widening technique from polyhedra to the more general semi-algebraic sets (to hasten convergence by over-approximation) are some of the many issues waiting to be explored. The analysis of perturbed and robust systems also warrants investigation. The semi-decidability results for the TCTL operators and the introduction of the Blum-Shub-Smale model are expected to spark further investigations of the relations between dynamical systems, topology and complexity. To demonstrate the generality of the AAMC approach, the techniques need to be extended beyond TCTL model-checking to dense-time LTL. The enhancement achievable by allowing non-linear (but polynomial) expressions in the temporal queries that can involve the values before and after one step of the hybrid system needs to be captured semantically and formally, possibly leading to a definition of a new subclass of temporal logic.

As we enlarge the scope of the biological models by considering metabolic processes, signal transduction processes and subcellular biochemical processes that are specific to locations and transportation between cellular compartments, the challenges to the algorithmic complexity and approximability deepen the need for better algorithmic algebraic techniques. In the process, we are also forced to explore the connection among constructive approaches for differential algebra, commutative algebra, Tarski-algebra, etc. It is hoped that all these ideas will coalesce into a powerful algebraic tool for the better analysis of biochemical systems. Although the state of the art of algebraic hybrid systems model-checking can only be compared to that of *Boolean finite-state model-checking* in the early 80s, we believe that the approach will make quick and important strides, and yield deep insights in biological areas before the end of this decade.

Appendix

A. Detailed Proofs for Chapter 4

Lemma 4.3.1 *A 1-dim PAM is bounded.*

Proof *Consider the 1-dim PAM* $f(x) = a_i x + b_i$, $x \in I_i(\equiv [l_i, r_i))$, $i = 1, 2, \cdots, n$. *Since it is a well-defined function, each interval has a post image which has to be covered by other intervals. Thus, if the* n *intervals are arranged in ascending order on the real line,* l_1 *is the smallest value that any trajectory of the PAM can ever reach. Similarly,* r_n *is the largest value that any trajectory can ever tend to. By definition, all intervals* I_i *have rational end points, hence* $\pm\infty$ *are excluded.* $\square$

Lemma 4.3.2 *Every 1-dim PAM is equivalent to a 1-dim "positive" PAM where all intervals are positive.*

Proof *Consider a 1-dim PAM* $f(x) = a_i x + b_i$, $x \in I_i(\equiv [l_i, r_i))$, $i = 1, 2, \cdots, n$. *Shifting the axis by* D *units,* $x' = x + D$ *and hence* $f'(x') = a_i(x' - D) + b_i + D$, $x \in I'_i(\equiv [l_i + D, r_i + D))$ *i.e.,* $a'_i = a_i$ *and* $b'_i = D(1 - a_i) + b_i$. *By picking* $D > |l_1|$, *all intervals become positive in this new PAM.* $\square$

Theorem 4.3.3 *A 1-dim PAM can be simulated by a 2-dim HPCD with comparative guards, 3 different flows* $+1, -1, 0$ *and no resets.*

Proof *Consider a 1-dim PAM* $f(x) = a_i x + b_i$, $x \in I_i$, $i = 1, 2, \cdots, n$. *Once again, we use the "taking-turns" idea. Unlike the previous case, we cannot make a variable start at* b_i *as we do not have resets. So, we now have* p *evolving from* x_{n-1} *to* x_{n+1}, *while* q *remains stationary at* x_n. *The guard condition* $p = a_i q + b_i$ *makes the HA jump to the next state at the correct time. Since* x_{n+1} *could be greater or less* *than x_{n-1}, the flow will need to be $+1$ or -1 respectively. Hence, each P (and Q) state now corresponds to two states: P^+ and P^-.*

We will construct a HPCD with $4n$ states of the form $P_j^{\pm}$ and $Q_j^{\pm}$ that simulates this PAM. Just as before, p and q will again take turns simulating x i.e., $p_{2i,2i+1} = x_{2i}$ and $q_{2i\pm1,2i+1\pm1} = x_{2i\pm1}$, at the end of each discrete transition.

Consider a state $P_j^{\pm}$ defined as follows:

- *Since $p_0 \in I_j$, q flows from $q_0(\in$ some $I_i)$ to $q' = a_j p_0 + b_j$.*

- *q's flow is $\dot{q} = +1$ since $q' > q_0$ in P_j^+. Simplifying $a_j p_0 + b_j > q_0$ yields $q_0 < (b_j + a_j b_i)/(1 - a_i a_j)$. In P_j^-, $q' > q_0$ and $\dot{q} = -1$. These must be incorporated into the guard of transitions leading to this state.*

- *p remains stationary i.e., $\dot{p} = 0$ in this state, in order to ensure that q flows to the correct amount.*

- *The guard for jumping out of this state is that q has reached q' i.e., $q = a_j p_0 + b_j = a_j p + b_j$*

- *The transitions out of this state are of the form $P_j^+ \to Q_k^{\pm}$ only. In the next state, q stays fixed at the value computed in this state, while p flows to the next iterant of x. In particular, the guard for Q_k^+ will be $q = a_j p + b_j \wedge q \in I_k \wedge p < (b_k + a_k b_j)/(1 - a_j a_k)$ (The last term will be $p \geq$ if we are jumping to Q_k^-).*

- *The $Q_j^{\pm}$ states are defined exactly as above, with p and q interchanged.*

Clearly, the above HPCD without resets simulates the given PAM. In particular, the reachability query "Is x_f reachable from x_0" *is true* iff $(p = x_f, q = x_{f-1})$ or $(p = x_{f-1}, q = x_f)$ is reachable from $(p = x_0, q = x_1)$, *where x_{f-1} is some pre-image of x_f. The proof follows from the fact that if (x_f, x_{f-1}) is reachable, then:*

- x_f *is the fixed value and* x_{f-1} *is the changing value, implying that* x_f *is indeed reachable.*
- x_f *is an intermediate changing value, while* x_{f-1} *is the fixed value. In this case* x_{f-1} *is reachable. But this guarantees that* x_f *can be reached in one more step, since post-images are unique. Hence, in this case also,* x_f *is reachable.*

Note that there can be at most n *pre-images of the target* x_f*, and hence we have* n *reachability queries that need to be asked. A factor of two crops up, because we never know if* x_f *requires odd or even iterations to reach (so we do not know if* p *is going to reach it, or if* q *is).* $\square$

Theorem 4.3.4 *A 1-dim PAM can be simulated by an initialized PCD with comparative guards.*

Proof *After a discrete transition, the variable carrying the current iterate of* x has *to flow at the same rate as it did in the previous state. If it did not, the "initialized" requirement would force it to be reset to a constant during the transition. So, we need* $2n^2$ *states of the form* P_{ij} *and* Q_{ij} *to simulate the PAM. Using the same trick as before, we ensure that* $p_{2i,2i+1} = L_{p_{2i,2i+1}} + x_{2i}$ *while* $q_{2i\pm1,2i+1\pm1} = L_{q_{2i\pm1,2i+1\pm1}} + x_{2i\pm1}$ *at the end of each discrete transition. In state* P_{ij}*:*

- p *flows from* p_0 *to* $p_0 + a_i p_0$ *with* $\dot{p} = a_i$
- q *flows from* b_j *to* $a_j p_0 + b_j$ *with* $\dot{q} = a_j$
- *Discrete transitions are of the form* $P_{ij} \rightarrow Q_{jk}$ *with guard* $\{(1 + a_i)(q - L_j) - a_j(p - L_i) - b_j(1 + a_i) = 0\} \wedge q \in I_k$ *and reset* $q' = q \wedge p' = b_k + L_k$*.* q *has its base at* L_j *in the next state, while* p *has its base already corrected to* L_k*. Clearly, only variables whose flow changes are initialized.* $\square$

Theorem 4.3.5 *A 1-dim PAM can be simulated by a 2-dim HPCD with rectangular guards, i.e., $p = 0 \wedge q \in I_i$ instead of $ax + by + c = 0 \wedge q \in I_i \wedge p \in I_j$ with constant or identity resets of the form $q' = a_j \wedge p' = p$ (rather than affine resets).*

Proof *The following HPCD with $2n$ states of the form P_j and Q_j simulates the equivalent positive PAM. The state P_j is defined as follows, with the other states defined symmetrically:*

- *While entering this discrete state, p has the current value of x.*
- *p flows from p_0 to 0 with $\dot{x} = -1$*
- *q flows from b_j to $b_j + p_0 a_j$ with $\dot{q} = a_j$*
- *The discrete state transitions are of the form $P_j \to Q_k$ with guard $p = 0 \wedge q > 0$ and reset $p' = b_k \wedge q' = q$. Since q is not reset even though its flow changes in the next state, this HA is not an "initialized" automata.* $\square$

Theorem 4.3.6 *A 1-dim bounded PAM can be simulated by a PCD with just clocks and translation resets when comparative guards are allowed.*

Proof *The following HPCD with $2n$ states of the form P_j and Q_j simulates the equivalent positive PAM. The trick is exactly as before: associate a number L_{P_i/Q_i} with each discrete state that is separated from every such number by at least L in the positive and negative directions. The state P_j is defined as follows:*

- *p flows from $L_{P_j} + p_0$ to $L_{P_j} + p_0 + a_j p_0 + b_j$ with $\dot{p} = +1$*
- *q flows from $L_{P_j} + 0$ to $L_{P_j} + a_j p_0 + b_j$ with $\dot{q} = +1$*

- *The discrete transitions will be of the form* $P_j \to Q_k$ *with guard* $a_j p - (1 + a_j)q + b_j + L_j = 0 \wedge q \in I_k$ *and reset* $p' = 0 + L_k \wedge q' = q - L_j + L_k$ □

Theorem 4.3.7 *A 1-dim PAM can be simulated by a 2-dim HPCD with the simple reset* $(x', y') = (0, y)$ *or* $(x, 0)$ *(i.e., one variable is not reset while the other is reset to* 0*) and all flows being* $+1$*, as long as comparative guards are allowed.*

Proof *The following HPCD with* $2n$ *states of the form* P_j *and* Q_j *simulates the equivalent positive PAM.* $p_{2i,2i+1} = x_{2i}$ *while* $q_{2i\pm1,2i+1\pm1} = x_{2i\pm1}$ *at the end of each discrete transition. The state* P_j *is defined as follows:*

- p *flows from* p_0 *to* $p_0 + a_j p_0 + b_j$ *with* $\dot{p} = +1$
- q *flows from* 0 *to* $a_j p_0 + b_j$ *with* $\dot{q} = +1$
- *The discrete transitions will be of the form* $P_j \to Q_k$ *with guard* $a_j p - (1 + a_j)q + b_j = 0 \wedge q \in I_k$ *and reset* $p' = 0 \wedge q' = q$ □

Theorem 4.3.9 *A 1-dim PAM can be simulated by a 2-dim HPCD without comparative guards or affine resets using origin dependent flows.*

Proof *The following HPCD with* $2n$ *states of the form* P_j *and* Q_j *simulates the equivalent positive PAM. The state* P_j *is defined as follows, with other states defined symmetrically:*

- p *flows from* p_0 *to* 0 *with* $\dot{p} = -1$
- q *flows from* 0 *to* $a_j p_0 + b_j$ *with* $\dot{q} = a_j + b_j/p_0$
- *Discrete state transitions are of the form* $P_j \to Q_k$ *with guard being* $p = 0 \wedge q \in I_k \wedge q > 0$ *and no resets* □

Theorem 4.4.2 *Reachability over HPCDs with zeno-paths ($HPCD_{zeno}$) is undecidable.*

Proof *Given an MM $\mathcal{M}$ with s states (program-lines) $q_1, q_2, \cdots, q_s$ and two counters m and n, we construct the HPCD $\mathcal{H}^s_{\mathcal{M}}$ with two variables x and y as follows:*

- *A discrete state l_i (with an associated integer value i) corresponds to the program-state q_i of $\mathcal{M}$*
- *The value of the counters is captured in state q_i as $x = i + 2^{-(m+1)}$ and $y = i + 2^{-(n+1)}$. This bounds the possible values of $\{x, y\}$ in l_i to $\{(i, i + \frac{1}{2}], (i, i + \frac{1}{2})\}$ effectively making the rectangles corresponding to the different discrete states bounded and non-overlapping (not necessary)*
- *MM computations:*
  - *$(q_i : m++, goto\ q_j)$ corresponds to a transition from l_i to l_j with reset $x' = (x - i)\frac{1}{2} + j$*
  - *$(q_i : m--, goto\ q_j)$ corresponds to a transition from l_i to l_j with reset $x' = (x - i)2 + j$*
  - *$(q_i : If\ m == 0\ goto\ q_j\ else\ goto\ q_k)$ corresponds to a transition from l_i to the state l_j with guard $x = i + \frac{1}{2}$ and reset $x' = x - i + j$, and a transition to the state l_k with guard $x < i + \frac{1}{2}$ and reset $x' = x - i + k$*

  The operations on n can be obtained by substituting x with y in the transformations above.

Clearly the HPCD $\mathcal{H}^s_{\mathcal{M}}$ simulates the given (and hence every) 2-counter MM $\mathcal{M}$. Since reachability is undecidable for the MM, it has to be undecidable for the HPCD *as well (if not, an "undecidable" MM can be converted to a HPCD, which can then be "decided").* □ **Alternate Construction** We can also construct a **1**-discrete-state HPCD $\mathcal{H}^1_{\mathcal{M}}$ where the value of the counters at line l_i of the MM is captured as $x = i + 2^{-(m+1)}$ and $y = i + 2^{-(n+1)}$. This bounds the possible values of $\{x, y\}$ in l_i to $\{(i, i + \frac{1}{2}], (i, i + \frac{1}{2})\}$ effectively storing the value of the next state (program-line) i in the value of x (and y).

- The guards are used to find out which instruction of the 2-counter machine we need to execute next, while the computations are trivially performed using the resets.
  - $(q_i : m++, goto\ q_j)$ corresponds to the self-loop with guard $i < x \leq i + \frac{1}{2}$ and reset $x' = (x - i)\frac{1}{2} + j$
  - $(q_i : m--, goto\ q_j)$ corresponds to the self-loop with guard $i < x \leq i + \frac{1}{2}$ and reset $x' = (x - i)2 + j$
  - $(q_i : If\ m == 0\ goto\ q_j\ else\ goto\ q_k)$ corresponds to two self-loops: one with guard $x = i + \frac{1}{2}$ and reset $x' = x - i + j$, and the other with guard $x < i + \frac{1}{2}$ and reset $x' = x - i + k$

  (substitute y for x to get the rules for n)

Theorem 4.4.3 *Reachability over HPCDs with access to the integer-check oracle in the guards is undecidable.*

Proof *Given an MM $\mathcal{M}$ with s states (program-lines) $q_1, q_2, \cdots, q_s$ and two counters m and n, we construct the HPCD $\mathcal{H}^s_{\mathcal{M}}$ with variables x and y as follows:*

1. HPCD_{fn-int}

- *A discrete state l_i corresponds to the program-state q_i of $\mathcal{M}$*
- *The value of the counters is captured in the variable x as $2^m 3^n$ while y is a dummy variable typically flowing from 0 to 1 in each state. Note that the rectangles corresponding to the different discrete states are bounded (because of y) but could possibly overlap.*
- *MM computations:*
  - $(q_i : m{+}{+}, goto\ q_j)$ *corresponds the state l_i with $\dot{x} = 0$ and $\dot{y} = 1$, with a transition to l_j with guard $y = 1$ and reset $x' = 2x \wedge y' = 0$.*
  - $(q_i : m{-}{-}, goto\ q_j)$ *corresponds the state l_i with $\dot{x} = 0$ and $\dot{y} = 1$, with a transition to l_j with guard $y = 1$ and reset $x' = \frac{1}{2}x \wedge y' = 0$.*
  - $(q_i : If\ m == 0\ goto\ q_j\ else\ goto\ q_k)$ *requires that we check if $x/2$ is an integer.*

    (a) State l_{i_1} has $\dot{x} = 0$ and $\dot{y} = 1$ and jumps to state l_{i_2} with guard $y = 1$ and reset $x' = \frac{1}{2}x$

    (b) State l_{i_2} has $\dot{x} = 0$ and $\dot{y} = 1$ and jumps to state l_j with guard $y = 1 \wedge integer(x)$ and reset $x' = 2x$, and jumps to state l_k with guard $y = 1 \wedge \neg integer(x)$ and reset $x' = 2x$

    The operations on n are obtained as usual by substituting y for x in the above rules.

Clearly the HPCD $\mathcal{H}^s_{\mathcal{M}}$ simulates the given (and hence any) 2-counter MM $\mathcal{M}$. Since reachability is undecidable for the MM, it has to be undecidable for the HPCD as well.

2. HPCD$_{zeno-int}$

The only difference is that instead of allowing a function $integer(x)$ in the guard, *we allow a zeno-computation that performs this check. To check if $x/2$ is an integer:*

(a) State l_{i_1} has $\dot{x} = 0$ and $\dot{y} = 1$ and jumps to state l_{i_2} with guard $y = \frac{1}{2}x$ and reset $x' = x \wedge y' = y$

(b) State l_{i_2} has $\dot{x} = 0$ and $\dot{y} = 0$ (zeno-state) and has a self-loop with guard $y > 1$ and reset $y' = y - 1$. It jumps to state l_k with guard $y = 1$ and reset $y' = 0 \wedge x' = x$, and jumps to state l_j with guard $y < 1 \wedge 0 \leq x < 1$ and reset $y' = 0 \wedge x' = x$ $\square$

Alternate Construction Alternatively, the two counters could be stored as $x = m + 2^{-(1+n)}$. m can be modified by addition and checked for 0 by $x < 1$. The fractional part ($2^{-(1+n)}$) can be accessed by letting y *flow* ($\dot{y} = +1$) from 0 until x flows down ($\dot{x} = -1$) and becomes an integer (guard: *integer*(x)). Since y now holds the fractional part, n can be modified by multiplication ($y = 0 \Rightarrow n = 0$). In the next state, x can flow ($\dot{x} = +1$) until y becomes 0 ($\dot{y} = -1$).

Lemma 4.5.1 *1-dim PAMs that can check if a given number x can be expressed as p^{-i} where p is a given prime number and i is an unknown positive integer can simulate a MM.*

Proof *The idea is to encode the two counters m and n, and the line-number (MM instruction) l in one real number as $x = l + 2^{-m}3^{-n}$. Thus when x lies in the range $I_l \equiv (l, l+1]$, the i-th instruction of the MM needs to be executed. Thus the MM instructions can be encoded as follows:*

- $(q_i : m++, goto\ q_j)$ *corresponds to* $x' = (x - i)\frac{1}{2} + j,\ x \in I_i$

- $(q_i : m--, goto\ q_j)$ *corresponds to* $x' = (x-i)2+j,\ x \in I_i$

- $(q_i : If\ m == 0\ goto\ q_j\ else\ goto\ q_k)$ *corresponds to* $x' = x-i+j,\ x \in I_i \wedge x-l = 3^{-n}$ *and to* $x' = x-i+k,\ x \in I_i \wedge x-l \neq 3^{-n}$ $\square$

Theorem 4.5.2 *Reachability is decidable for 1-dim oPAMs.*

Proof *We just have to show how we can verify the reachability query after the partitioning algorithm converges. After convergence, we will be left with several "interval-node-paths" of the form* $P_i \rightarrow P_{i+1} \cdots \rightarrow P_j \rightarrow P_{j+1} \cdots \rightarrow P_j$ *i.e., a cycle of interval-nodes possibly preceded by a linear path of interval-nodes. All these interval-node-paths will clearly be non-overlapping. If the given* x_0 *and* x_f *do not lie on the same interval-node-path, then* x_f *is unreachable from* x_0*. Otherwise, we just numerically iterate from* x_0 *until* x_f *is reached. If this does not happen in* $2k$ *steps where* k *is the length of the interval path containing* x_0*, we can conclude that* x_f *is unreachable using* Lemma 3 *above. Hence,* x_f *is reachable* iff *it is encountered on this* $2k$*-long path starting at* x_0*.* $\square$

Theorem 4.6.1 *A 1-dim PAM is decidable if its PCD-Graph is planar.*

Proof *A discrete reset operation can be interpreted as a continuous flow operation if we can connect the starting and initial values by a sequence of convex regions with constant flow (i.e., a series of PCD states). If we perform this transformation for all the resets in the system, we get a 2-dim PCD system that simulates the original 1-dim PAM. This 2-dim PAM is decidable for the reachability query [201]. However, since the invariants corresponding to the different states cannot overlap, we must be able to arrange them on a plane without any intersections. Hence the original reset lines* *have to be planar for the 1-dim PAM to be decidable. While the circular arrangement captures the fact that the order of the output ports cannot be changed, the central node prevents the interiors of the different states from overlapping.* $\square$

B. Detailed Proofs for Chapter 7

Theorem 7.2.1 *The standard bisimulation partitions are computable for semi-algebraic hybrid automata.*

Proof *Each state s (source) needs to be split into two states s_d and $s_{\bar{d}}$ depending on whether or not the guard of the transition to each d (destination) can ever be satisfied. Since real quantifier elimination is decidable [276, 151], these partitions can be computed thus:* $Inv_{s_d}(X) \equiv \exists h, X' \ \langle s, X \rangle \xrightarrow[C]{h} \langle s, X' \rangle \wedge Guard_{s,d}(X')$ *and* $Inv_{s_{\bar{d}}}(X) \equiv Inv_s(X) \wedge \neg Inv_{s_d}(X)$.

Theorem 7.2.2 *There are only a finite number of 1-to-1 coupled linear maps possible between two sets with finite axes of symmetry.*

Proof *Consider two sets S_1 and S_2 between which* onto *linear maps exist. Maps of the form $x' = \Sigma a_i x_i + a_0$ correspond to a rotation, stretch and shift of the coordinate axes. In other words, S_2* has *to be a stretched, rotated and shifted image of S_1 for such an onto map to exist. There are 2^{s_l} maps possible because of the s_l axes of linear symmetry, on* each *of the s_r axes of rotational symmetry. Hence the total number of coupled linear onto maps is $s_r 2^{s_l}$.* $\square$

Corollary 7.2.1 *Given a cycle of n d-dimensional sets with s_l axes of linear symmetry and s_r axes of rotational symmetry each, where each set maps exactly onto its* *successor, the number of unique successors of any point is at most* $ns_r2^{s_l}$.

Proof *Let* $m(= s_r2^{s_l})$ *be the number of possible onto maps between* S_1 *and* S_2. *Let* $x_1 \in S_1$ *map to one of* $y_1, \cdots, y_m \in S_2$. *Let* y_1 *map to one of* $x_1, \cdots, x_m \in S_1$ *(*x_1 *has to appear because the inverse of a linear map is linear). Suppose* x_2 *maps to* y_{m+1}. *Since linear maps are closed under composition and retain their ontoness property, by following the linear maps from* $x_1 \rightarrow y_{1\leq i\leq m} \rightarrow x_{1\leq i\leq m} \rightarrow y_{m+1}$, *we get a new linear map that takes* x_1 *to* y_{m+1}. *However, we know from symmetry arguments that only* m *linear maps can exist. This contradiction proves that no matter how many times we compose the two given onto linear maps, we remain within the set of* $2n$ *points (for example, rectangles have 8 possible linear onto maps while cubes have 24). Extending this argument to a cycle of* n *states, each point can have only one of* m *different successors in each of the* n *states. Hence, the length of the biggest cycle is* nm. $\square$

Theorem 7.2.3 *Reachability over a semi-algebraic hybrid system with coupled linear resets and flows is decidable if the partitioning algorithm converges into states with finite axes of symmetry.*

Proof *The continuous evolution can be treated as a linear map from the initial value to the final value that first satisfies the guard. Further,* time *does not appear in the equation as in deterministic systems, an initial value corresponds to a unique final value. No restriction on the guard is necessary as we assume that there is exactly* 1 *successor to each discrete state. Thus a cycle of* n *states corresponds to a cycle of* $2n$ *linear 1-to-1 maps with only the values before and after a reset sufficing to capture the dynamics. If* m *is the maximum number of possible onto maps between any two* *consecutive sets, the number of unique successors is* $\leq 2nm$. *Since* x_0 *has a finite number of successors, if the target* x_f *is not reached before the system begins to cycle, we conclude exact unreachability. If* x_f *is eventually reached, then it is indeed exactly reachable.* □

Theorem 7.2.4 *If there exist 1-to-1 monotonic maps between two sets, all points converge to one of a finite number of fixed points or limit cycles.*

Proof *Let the sequence be* $X_0,\ X_1 = f(X_0),\ X_0' = g(X_1) = g(f(X_0)), \cdots$. *Since* f *and* g *are monotonic,* X *will continue to move in the same direction. The process can continue ad infinitum if* X *approaches a fixed point* ($g(f(X)) = X$). *The other mathematical alternative is that it approaches a limit cycle* ($f(g(f(g(..(X)..)))) = X$). *Monotonicty ensures progress while ontoness ensures finiteness of the number of iterations required to reach the neighborhood of a fixed point or a limit cycle.* □

Theorem 7.2.5 *The following problem is decidable, independent of the degree of accuracy needed: Reachability over a semi-algebraic hybrid system with monotonic resets and flows that converges upon partitioning.*

Proof *Just as in the linear case, the continuous flow can also be thought of as a monotonic function from the initial value (from a reset that brought the system to this state) to the final value (when a guard is first satisfied). Thus any cycle of* n *discrete states corresponds to a cycle of* $2n$ *monotonic maps (*n *flow-maps and* n *reset maps). Further, from the previous theorem, we know that iterative evolution along such a cycle of exactly onto maps has to approach a fixed point or a limit cycle. We can stop iterating when* $\bigwedge(|X_i - X_i'| < \epsilon_i)$, *where* X *is the d-dimensional value of the* *system variables, ϵ_i is the desired accuracy in the i-th dimension and X' is the value after one cycle (reached the neighborhood of a fixed point) or after $2, 3, \cdots, d$ cycles (reached the neighborhood of a limit cycle) . The monotonic resets guarantee that this condition will be satisfied eventually in a finite number of steps and that once this happens, the system cannot escape out of it.* $\square$

C. Symbols and Abbreviations

Abbreviations frequently used in the thesis have been tabulated in *Table* .1, while the symbols have been defined in *Table* .2.

Table .1: Abbreviations used in this thesis

Abbreviation	**Expansion**
AAMC	Algorithmic Algebraic Model Checking
alg.	algorithm
BDD	Binary Decision Diagram
BMC	Bounded Model Checking
CAD	Cylindrical Algebraic Decomposition
defn.	definition
dim	dimension
FBA	Flux Balance Analysis
fig.	figure
HA	Hybrid Automaton
HPCD	Hierarchical Piecewise Constant Derivative System
IDA	Independent Dynamics Automata
KMA	Kinetic Mass Action
LTL	Linear Temporal Logic
MCA	Metabolic Control Analysis
MM	Minsky Machine
ODE	Ordinary Differential Equation
oPAM	onto Piecewise Affine Map
PAM	Piecewise Affine Map
PCD	Piecewise Constant Derivative System
Qepcad	Quantifier Elimination by Partial Cylindrical Algebraic Decomposition
SACoRe	Semi-Algebraic Constant Reset Automata
sec.	section
TCTL	Timed Computation Tree Logic
TL	Temporal Logic
TM	Turing Machine
Tolque	TempOral Logic via QUantifier Elimination

Table .2: Symbols used in this thesis

Symbol	**Expansion**
$\mathcal{F}$	TL Eventually / Future operator
$\mathcal{G}$	TL Generally / Henceforth operator
$\mathcal{U}$	TL Until operator
$\triangleright$	TCTL One-Step Until operator
$\forall$	for all (universal quantifier)
$\exists$	exists (existential quantifier)
$\neg$	Boolean not
$\vee$	Boolean or
$\wedge$	Boolean and
$\cup$	set union
$\cap$	set intersection
$^{-}$	set complement
$\xrightarrow[\mathcal{C}]{t}$	Continuous evolution expression
$\xrightarrow[\mathcal{D}]{0}$	Discrete evolution expression

Bibliography

[1] Andrew Adams, Martin Dunstan, Hanne Gottliebsen, Tom Kelsey, Ursula Martin, and Sam Owre. Computer algebra meets automated theorem proving: Integrating maple and pvs. In R.J. Boulton and P.B. Jackson, editors, *TPHOLs*, volume 2152 of *LNCS*, page 2742. Springer-Verlag Berlin, 2001.

[2] Alan Aderem. Systems biology: Its practice and challenges. *Cell*, 121(4):511–513, May 2005.

[3] Participating investigators & scientists of the Alliance for Cellular Signaling AfCS. Overview of the alliance for cellular signaling. *Nature*, 420:703–706, December 2002.

[4] E. Alm and A.P. Arkin. Biological networks. *Current Opinion in Structural Biology*, 13:193–202, 2003.

[5] R.E. Altenbaugh, K.J. Kauffman, and J.S. Edwards. Suitability and utility of computational analysis tools: characterization of erythrocyte parameter variation. In *Pac Symp Biocomput.*, pages 104–15, 2003.

[6] R. Alur, C. Belta, F. Ivancic, V. Kumar, M. Mintz, G. J. Pappas, H. Rubin, and J. Schug. Hybrid Modeling and Simulation of Biomolecular Networks. In

Hybrid Systems: Computation and Control, volume 2034 of *LNCS*, pages 19–32. Springer, 2001.

[7] R. Alur, C. Courcoubetis, and D. Dill. Model-Checking for Real-Time Systems. In *International Symposium on Logic in Computer Science*, 5, pages 414–425. IEEE Computer Press, 1990.

[8] R. Alur, C. Courcoubetis, and D.L. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

[9] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The Algorithmic Analysis of Hybrid Systems. *Theoretical Computer Science*, 138:3–34, 1995.

[10] R. Alur, T. Dang, and F. Ivančić. Progress on Reachability Analysis of Hybrid Systems using Predicate Abstraction. In O. Maler and A. Pnueli, editors, *Sixth International Workshop on Hybrid Systems: Computation and Control (HSCC 2003)*, volume 2623 of *LNCS*. Springer, 2003.

[11] R. Alur and D. Dill. The Theory of Timed Automata. In J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, editors, *Real-Time: Theory in Practice (REX Workshop)*, volume 600 of *LNCS*, pages 45–73. Springer-Verlag, 1992.

[12] R. Alur and D.L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126:183–235, 1994.

[13] R. Alur, T. Feder, and T.A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43:116–146, 1996.

[14] R. Alur and T. A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.

[15] R. Alur and T. A. Henzinger. A really temporal logic. *Journal of the Association for Computing Machinery*, 41(1):181–204, 1994.

[16] R. Alur, T. A. Henzinger, and P.-H. Ho. Automatic Symbolic Verification of Embedded Systems. *IEEE Transactions on Software Engineering*, 22:181–201, 1996.

[17] R. Alur, T. A. Henzinger, and Pei-Hsin Ho. Automatic Symbolic Verification of Embedded Systems. In *IEEE Real-Time Systems Symposium*, pages 2–11. IEEE Press, 1993.

[18] Rajeev Alur, Radu Grosu, Yerang Hur, Vijay Kumar, and Insup Lee. Modular specification of hybrid systems in charon. In *Hybrid Systems: Computation and Control, Proceedings of the Third International Conference*, volume 1790 of *Lecture Notes in Computer Science*, pages 6–19. Springer, 2000.

[19] K. Amonlirdviman, R. Ghosh, J.D. Axelrod, and C.J. Tomlin. A hybrid systems approach to modeling and analyzing planar cell polarity. In *Proceedings of the 3rd International Conference on Systems Biology*, 2002.

[20] Hirokau Anai. On solving semidefinite programming by quantifier elimination. In *Proceedings of the American Control Conference*, June 1998.

[21] Hirokazu Anai. Algebraic approach to analysis of discrete-time polynomial systems. In *ECC Karlsure (Germany)*, 1999.

[22] Hirokazu Anai and Volker Weispfenning. Reach set computations using real quantifier elimination, 2000.

[23] Hirokazu Anai and Hitoshi Yanami. Synrac: A maple-package for solving real algebraic constraints. In *International Conference on Computational Science*, volume 2657 of *Lecture Notes in Computer Science*, pages 828–837. Springer, 2003.

[24] Thomas Anantharaman, Venkatesh Mysore, and Bud Mishra. Fast and cheap genome wide haplotype construction via optical mapping. In R.B. Altman, A.K. Dunker, L. Hunter, T.A. Jung, and T.E.Klein, editors, *The Pacific Symposium on Biocomputing: PSB*. World Scientific, January 2005.

[25] B.D.O. Anderson and P.J. Moylan. Structure result for nonlinear passive systems. In *Int Symp Operator Theory of Networks and Syst, Montreal, Canada*, Aug 1975.

[26] A. Annichini, E. Asarin, and A. Bouajjani. Symbolic techniques for parametric reasoning about counter and clock systems. In *Computer Aided Verification*, volume 1855. SpringerVerlag Heidelberg, Germany, 2000.

[27] A. Annichini, A. Bouajjani, and M. Sighireanu. Trex: a tool for reachability analysis of complex systems. In *Computer Aided Verification*, volume 2102. SpringerVerlag Heidelberg, Germany, 2001.

[28] M. Antoniotti, P.E. Barbano, W. Casey, J.-W. Feng, N. Ugel, and B. Mishra. Multiple biological model classification: From system biology to synthetic biology. *BioConcur'04, 2nd Workshop on Concurrent Models in Molecular Biology, The Royal Society, London, Transactions on Computational Systems Biology*, 2005.

[29] M. Antoniotti, B. Mishra, C. Piazza, A. Policriti, and M. Simeoni. Modelling

cellular behavior with hybrid automata: Bisimulation and collapsing. In *International workshop on Computational Methods in Systems Biology, CMSB'03, (Ed. C. Priami)*, volume 2602 of *Lecture Notes in Computer Science*, pages 57–74. Springer-Verlag, 2003.

[30] M. Antoniotti, B. Mishra, C. Piazza, A. Policriti, and M. Simeoni. Taming the complexity of biochemical models through bisimulation and collapsing: Theory and practice. *Theoretical Computer Science*, 325(1):45–67, 2004.

[31] M. Antoniotti, F. C. Park, A. Policriti, N. Ugel, and B. Mishra. Foundations of a Query and Simalation System for the Modeling of Biochemical Processes. In *Proceedings of the Pacific Symposium on Biocomputing 2003 (PSB 2003)*, January 2003.

[32] M. Antoniotti, A. Policriti, N. Ugel, and B. Mishra. XS-systems: eXtended S-Systems and Algebraic Differential Automata for Modeling Cellular Behavior. In *Proceedigs of the International Confernce on High Performance Computing, HiPC 2002*, Bangalore, India, December 2002.

[33] M. Antoniotti, A. Policriti, N. Ugel, and B. Mishra. Reasoning about Biochemical Processes. *Cell Biochemistry and Biophysics*, 38:271–286, 2003.

[34] E. Asarin and A. Bouajjani. Perturbed turing machines and hybrid systems. In *LICS*, 2001.

[35] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate Reachability Analysis of Piecewise-linear Dynamical Systems. In *Third International Workshop on Hybrid Systems: Computation and Control (HSCC 2000)*, volume 1790 of *LNCS*, pages 21–31. Springer-Verlag, 2000.

[36] E. Asarin, T. Dang, and O. Maler. `d/dt`: a Verification Tool for Hybrid Systems. In *Proceedings of the 40th IEEE Conference on Decision and Control*. IEEE, December 2001.

[37] E. Asarin, T. Dang, O. Maler, and O. Bournez. Approximate Reachability Analysis of Piecewise-Linear Dynamical Systems. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control*, volume 1790 of *LNCS*, pages 20–31. Springer, 2000.

[38] E. Asarin, O. Maler, and A. Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138:35–65, 1995.

[39] E. Asarin and G. Schneider. Widening the boundary between decidable and undecidable hybrid systems. In *CONCUR'2002, Brno, Czech Republic*, volume 2421 of *LNCS*, pages 193–208. Springer-Verlag, August 2002.

[40] E. Asarin, G. Schneider, and S. Yovine. On the decidability of the reachability problem for planar differential inclusions. In *In: Hybrid Systems: Computation and Control*, pages 89–104. LNCS 2034, March 2001.

[41] Eugene Asarin, Thao Dang, and Oded Maler. The d/dt tool for verification of hybrid systems. In D. Brinksma and K. G. Larsen, editors, *CAV*, volume 2404 of *LNCS*, page 365370. Springer-Verlag Berlin, 2002.

[42] A.R. Asthagiri and D.A. Lauffenburger. Bioengineering models of cell signaling. *Annu Rev Biomed Eng.*, 2:31–53, 2000.

[43] George S. Avrunin. Symbolic model checking using algebraic geometry. In Rajeev Alur and Thomas A. Henzinger, editors, *Computer Aided Verification, 8th*

International Conference, volume 1102, pages 26–37. Springer-Verlag, August 1996.

[44] Thomas Ball and Sriram K. Rajamani. The slam project: Debugging system software via static analysis. In *POPL'02*, pages 1–3, January 2002.

[45] P. Barbano, M. Spivak, J. Feng, M. Antoniotti, and B. Mishra. A coherent framework for multi-resolution analysis of biological networks with memory: Ras pathway, cell cycle and immune system. *Proc. National Academy of Science U S A*, 102(18):6245–6250, 2005.

[46] Michael P. Barnett. Computer algebra in the life sciences. *SIGSAM Bull.*, 36(4):5–32, 2002.

[47] M.P. Barnett, J.F. Capitani, J. Gathen, and J. Gerhard. Symbolic calculation in chemistry: Selected examples. *International Journal of Quantum Chemistry*, 100:80–104, 2004.

[48] Clark Barrett, Sergey Berezin, Igor Shikanian, Marsha Chechik, Arie Gurfinkel, and David L. Dill. A practical approach to partial functions in CVC Lite. In *PDPAR'04 Workshop, Cork, Ireland*, July 2004.

[49] Saugata Basu. New results on quantifier elimination over real closed fields and applications to constraint databases. *Journal of the ACM*, 46(4):537–555, 1999.

[50] Saugata Basu, Richard Pollack, and Marie-Francoise Roy. On the combinatorial and algebraic complexity of quantifier elimination. In *IEEE Symposium on Foundations of Computer Science*, pages 632–641, 1994.

[51] Saugata Basu, Richard Pollack, and Marie-Franoise Roy. Computing roadmaps of semi-algebraic sets on a variety (extended abstract). In *FoCM '97: Selected*

papers of a conference on Foundations of computational mathematics, pages 1–15, New York, NY, USA, 1997. Springer-Verlag New York, Inc.

[52] G. Batt, H. de Jong, J. Geiselmann, and M. Page. Qualitative analysis of genetic regulatory networks : A model-checking approach. In *B. Bredeweg, P. Salles (eds.), Working Notes of Seventeenth International Workshop on Qualitative Reasoning, QR-03*, pages 31–38, 2003.

[53] Andreas D. Baxevanis. The molecular biology database collection: 2003 update. *Nucleic Acids Res.*, 31(1):1–12, Jan 2003.

[54] Mustafa Bayram. Computer algebra approaches to enzyme kinetics. Ph.D. Thesis, School of Mathematical Sciences, University of Bath, England, 1993.

[55] Mustafa Bayram and Ercan Celik. Simultaneous solution of polynomial equations. *Applied Mathematics and Computation*, 133:533–538, 2002.

[56] Daniel A. Beard, Shou dan Liang, and Hong Qian. Energy balance for analysis of complex metabolic networks. *Biophysical Journal*, 83:79–86, July 2002.

[57] Bernd Becker, Markus Behle, Fritz Eisenbrand, Martin Fraenzle, Marc Herbstritt, Christian Herde, Joerg Hoffmann, Daniel Kroening, Bernhard Nebel, Ilia Polian, and Ralf Wimmer. Bounded model checking and inductive verification of hybrid discrete-continuous systems. In *GI/ITG/GMM Workshop*, 2004.

[58] G. Behrmann, A. David, K. Larsen, O. Möeller, P. Petterson, and W. Yi. UPPAAL – Present and Future. In *Proceedings of the 40th IEEE Conference on Decision and Control.* IEEE, December 2001.

[59] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on UPPAAL. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods form the*

Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004, number 3185 in LNCS, pages 200–236. Springer–Verlag, September 2004.

[60] S. Bensalem, V. Ganesh, Y. Laknech, C. M unoz, S. Owre, H. Rueß, J. Rushby, V. Rusu, H. Saïdi, N. Shankar, E. Singerman, and A. Tiwari. An Overview of SAL. In C. M. Holloway, editor, *Fifth NASA Langley Formal Methods Workshop, LFM 2000*, 2000.

[61] B. Berard and C. Dufourd. Timed automata and additive clock constraints. *Information Processing Letter*, 75(1-2):1–7, 2000.

[62] H.C. Berg. Motile behavior of bacteria. *Physics Today*, 53(1):24–29, 2000.

[63] U. S. Bhalla and R. Iyengar. Emergent properties of networks of biological signaling pathways. *Science*, 283:381–387, 1999.

[64] A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. *Advances in Computers*, 58, 2003.

[65] Nikolaj Bjorner, Zohar Manna, Henny Sipma, and Toms Uribe. Deductive verification of real-time systems using step. *Theoretical Computer Science*, 253:27–60, 2001.

[66] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer-Verlag, 1997.

[67] Alexander Bockmayr and Arnaud Courtois. Using hybrid concurrent constraint programming to model dynamic biological systems. In *18th International Conference on Logic Programming, ICLP'02, Copenhagen*, volume 2401 of *LNCS*, pages 85–99. Springer, July 2002.

[68] Peer Bork. Is there biological research beyond systems biology? a comparative analysis of terms. *Molecular Systems Biology*, May 2005.

[69] Peer Bork and Luis Serrano. Towards cellular systems in 4d. *Cell*, 121(4):507–509, May 2005.

[70] O. Botchkarev and S. Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In *HSCC*, volume 1790 of *LNCS*, pages 73–88. Springer, 2000.

[71] O. Bournez, O. Maler, and A. Pnueli. Orthogonal Polyhedra: Representation and Computation. In F. Vaadrager and J. van Schuppen, editors, *Hybrid Systems: Computation and Control (HSCC 1999)*, volume 1596 of *LNCS*, pages 19–30. Springer-Verlag, 1999.

[72] M. C. Browne, E. M. Clarke, and O. Grumberg. Characterizing Finite Kripke Structures in Propositional Temporal Logic. *Theoretical Computer Science*, 59:115–131, 1988.

[73] B. Buchberger. Grobner bases: An algorithmic method in polynomial ideal theory. *Recent Trends in Multidimensional Systems Theory*, pages 184–232, 1985.

[74] A. Casagrande, C. Piazza, and B. Mishra. Semi-Algebraic Constant Reset Hybrid Automata - SACoRe. In *Proceedings of the 44rd Conference on Decision and Control and European Control Conference (CDC-ECC'05)*, pages 678–683, Seville, Spain, December 2005. IEEE Computer Society Press.

[75] Alberto Casagrande, Venkatesh Mysore, Carla Piazza, and Bud Mishra. Independent dynamics hybrid automata in system biology. In *Proceedings of the*

First International Conference on Algebraic Biology, Tokyo, (Japan), 28-30 November, 2005.

[76] M. Cascante, L.G. Boros, B. Comin-Anduix, P. de Atauri, J.J. Centelles, and P.W.-N. Lee. Metabolic control analysis in drug discovery and design. *Nature Biotechnology*, 20:243–249, 2002.

[77] E. Celik, E. Karaduman, and M. Bayram. Numerical method to solve chemical differential-algebraic equations. *International Journal of Quantum Chemistry*, 89(5):447–451, 2002.

[78] Ercan Celik and Mustafa Bayram. Application of grobner basis techniques to enzyme kinetics. *Applied Mathematics and Computation*, 153:97–109, 2004.

[79] N. Chabrier, M. Chiaverini, V. Danos, F. Fages, and V. Schächter. Modeling and querying biochemical interaction networks. *Theoretical Computer Science*, 325(1):25–44, September 2004.

[80] N. Chabrier and F. Fages. Symbolic model checking of biochemical networks. In *Proceedings of the First International Workshop on Computational Methods in Systems Biology*, pages 149 – 162, 2003.

[81] Ming Chen and Ralf Hofestdt. Quantitative petri net model of gene regulated metabolic networks in the cell. *In Silico Biology*, 3(3):347–65, 2003.

[82] A. Chutinan and B. Krogh. Verification of Polyhedral-Invariant Hybrid Automata Using Polygonal Flow Pipe Approximations. In F. W. Vaandrager and J. H. van Schuppen, editors, *Hybrid Systems: Computation and Control*, volume 1569 of *LNCS*, pages 76–90. Springer, 1999.

[83] Alongkrit Chutinan and Bruce H. Krogh. Computational techniques for hybrid system verification. *IEEE Trans. Automatic Control*, 48(1):64–75, 2003.

[84] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In *Proc. International Conference on Computer-Aided Verification (CAV 2002)*, volume 2404 of *LNCS*, Copenhagen, Denmark, July 2002. Springer.

[85] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. Nusmv: a New Symbolic Model Checker. *STTT International Journal on Software Tools for Technology Transfer*, 2:410–425, 2000.

[86] E. Clarke, A. Fehnker, Z. Han, B. Krogh, J. Ouaknine, O. Stursberg, and M. Theobald. Abstraction and counterexample-guided refinement in model checking of hybrid systems. *International Journal of Foundations of Computer Science (IJFCS) special issues on Verification and Analysis of Infinite State Systems*, 2003.

[87] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.

[88] CoCoATeam. CoCoA: a system for doing Computations in Commutative Algebra. Available at `http://cocoa.dima.unige.it`, 2005.

[89] J. R. Collier, N. A. M. Monk, P. K. Maini, and J. H. Lewis. Pattern Formation by Lateral Inhibition with Feedback: a Mathematical Model of Delta-Notch Intercellular Signalling. *Journal of Theor. Biology*, 183:429–446, 1996.

[90] G. E. Collins. Quantifier Elimination for the Elementary Theory of Real Closed

Fields by Cylindrical Algebraic Decomposition. In *Proceedings of the Second GI Conference on Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 134–183. Springer-Verlag, 1975.

[91] A. Cornish-Bowden. *Fundamentals of Enzyme Kinetics (3rd edn.).* Portland Press, London, 2004.

[92] A. Cornish-Bowden and M. L. Cardenas. Metabolic analysis in drug design. *C. R. Biologies*, 326:509–515, 2003.

[93] A. Cornish-Bowden and M.L. Cardenas. Systems biology may work when we learn to understand the parts in terms of the whole. *Biochemical Society Transactions*, 33(3), 2005.

[94] A. Cornish-Bowden and J.-H. S. Hofmeyr. Enzymes in context: Kinetic characterization of enzymes for systems biology. *The Biochemist*, 27:11–14, 2005.

[95] Athel Cornish-Bowden, Jan-Hendrik S. Hofmeyr, and Mara Luz Crdenas. Stoicheiometric analysis in studies of metabolism. *Biochem. Soc. Trans.*, 30:43–46, 2002.

[96] Patrick Cousot, Radhia Cousot, Jrme Feret, Laurent Mauborgne, Antoine Min, David Monniaux, and Xavier Rival. The astre analyser. In M. Sagiv, editor, *In ESOP 2005 – The European Symposium on Programming*, volume 3444 of *Lecture Notes in Computer Science*, pages 21–30. Springer, 2005.

[97] M. Curti, P. Degano, C. Priami, and C.T. Baldari. Modelling biochemical pathways through enhanced pi-calculus. *Theoretical Computer Science*, 2003.

[98] J.H. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, 5:29–35, 1988.

[99] E.H. Davidson, J.P. Rast, P. Oliveri, A. Ransick, C. Calestani, C.H. Yuh, T. Minokawa, G. Amore, V. Hinman, C. Arenas-Mena, O. Otim, C.T. Brown, C.B. Livi, P.Y. Lee, R. Revilla, A.G. Rust, Z. Pan, M.J. Schilstra, P.J. Clarke, M.I. Arnone, L. Rowen, R.A. Cameron, D.R. McClay, L. Hood, and H. Bolouri. A genomic regulatory network for development. *Science*, 295(5560):1669–78, Mar 2002.

[100] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In *Hybrid Systems III, Verification and Control*, volume 1066 of *LNCS*, pages 208–219. Springer-Verlag, 1996.

[101] H. de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):69 – 105, 2002.

[102] H. de Jong. Modeling and simulation of genetic regulatory networks. *Lectures Notes in Control and Information Sciences*, 294:111–118, 2003.

[103] Leonardo de Moura, Sam Owre, Harald Rue, John Rushby, N. Shankar, Maria Sorea, and Ashish Tiwari. Sal 2. In *Tool description presented at CAV, Boston MA*, volume 3114 of *LNCS*, pages 496–500. Springer Verlag, July 2004.

[104] E. Demir, O. Babur, U. Dogrusoz, A. Gursoy, A. Ayaz, G. Gulesir, G. Nisanci, R. Cetin-Atalay, and M. Ozturk. Patika: Pathway analysis tool for integration and knowledge acquisition. In *Poster Presentation in ISMB 2003, Brisbane, Australia*, 2003.

[105] Yves Deville, David Gilbert, Jacques van Helden, and Shoshana Wodak. An overview of data models for the analysis of biochemical pathways. *Briefings in Bioinformatics*, 4(3):246 – 259, 2003.

[106] P.K. Dhar, Hao Zhu, and S.K. Mishra. Computational approach to systems biology: from fraction to integration and beyond. *NanoBioscience, IEEE Transactions on*, 3(3):144– 152, Sept. 2004.

[107] Dolzmann, Gloor, and Sturm. Approaches to parallel quantifier elimination. In *ISSAC: Proceedings of the ACM SIGSAM International Symposium on Symbolic and Algebraic Computation (formerly SYMSAM, SYMSAC, EUROSAM, EUROCAL) (also sometimes in cooperation with the Symbolic and Algebraic Manipulation Groupe in Europe (SAME))*, 1998.

[108] A. Dolzmann, T. Sturm, and V. Weispfenning. Real quantifier elimination in practice. Technical Report MIP9720, FMI, Universitat Passau, D-94030 Passau, Germany, December 1997.

[109] Andreas Dolzmann and Thomas Sturm. REDLOG: Computer algebra meets computer logic. *SIGSAM Bulletin (ACM Special Interest Group on Symbolic and Algebraic Manipulation)*, 31(2):2–9, 1997.

[110] Matthew B. Dwyer, John Hatcliff, Matthew Hoosier, and Robby. Building your own software model checker using the bogor extensible model checking framework. In *In the Proceedings of 17th Conference on Computer-Aided Verification (CAV 2005)*, 2005.

[111] Jeremy S. Edwards and Bernhard O. Palsson. Metabolic flux balance analysis and the in silico analysis of escherichia coli k-12 gene deletions. *BMC Bioinformatics*, 1(1), 2000.

[112] J.S. Edwards, R.U. Ibarra, and B.O. Palsson. In silico predictions of escherichia

coli metabolic capabilities are consistent with experimental data. *Nat Biotechnol.*, 19(2):125–30, Feb 2001.

[113] S. Efroni, D. Harel, and I.R. Cohen. Toward rigorous comprehension of biological complexity: modeling, execution, and visualization of thymic t-cell maturation. *Genome Research*, 13(11):2485–97, Nov 2003.

[114] S. Eker, M. Knapp, K. Laderoute, P. Lincoln, J. Meseguer, and K. Sonmez. Pathway logic: Symbolic analysis of biological signaling. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 400–412, January 2002.

[115] M. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 409(18):391–395, Jan 2001.

[116] E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995–1072. MIT Press, 1990.

[117] Damien Eveillard, Delphine Ropers, Hidde de jong, Christiane Branlant, and Alexander Bockmayr. A multi-site constraint programming model of alternative splicing regulation. INRIA Technical Report, May 2003.

[118] J.R. Faeder, M.L. Blinov, B. Goldstein, and W.S. Hlavacek. Rule-based modeling of biochemical networks. *Complexity*, 10:22–41, 2005.

[119] D. A. Fell. *Understanding the Control of Metabolism*. Portland Press, London, 1997.

[120] J. Fisher, N. Piterman, E.J.A. Hubbard, M. Stern, and D. Harel. Computational insights into c. elegans vulval development. *Proc. Natl. Acad. Sci. USA*, 102:1951–1956, 2005.

[121] Martin Fränzle. Analysis of hybrid systems: An ounce of realism can save an infinity of states. In Jörg Flum and Mario Rodríguez-Artalejo, editors, *Computer Science Logic (CSL'99)*, volume 1683 of *LNCS*, pages 126–140. Springer, 1999.

[122] Martin Fränzle. What will be eventually true of polynomial hybrid automata? In Naoki Kobayashi and Benjamin C. Pierce, editors, *Theoretical Aspects of Computer Software, 4th International Symposium, TACS 2001, Sendai, Japan, October 29-31, 2001, Proceedings*, volume 2215 of *Lecture Notes in Computer Science*, pages 340–359. Springer, 2001.

[123] Martin Fränzle and C. Herde. Efficient proof engines for bounded model checking of hybrid systems. In *FMICS*, 2004.

[124] Goran Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. In Manfred Morari and Lothar Thiele, editors, *Hybrid Systems: Computation and Control, 8th International Workshop, HSCC 2005, Zurich, Switzerland, March 9-11, 2005, Proceedings*, volume 3414 of *Lecture Notes in Computer Science*, pages 258–273. Springer, 2005.

[125] R. Ghosh, A. Tiwari, and C. Tomlin. Automated Symbolic Reachability Analysis; with Application to Delta-Notch Signaling Automata. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control, HSCC 2003*, volume 2623 of *LNCS*, pages 233–248. Springer, 2003.

[126] R. Ghosh and C. Tomlin. Lateral Inhibition through Delta-Notch signaling: A Piecewise Affine Hybrid Model. In M. D. D. Benedetto and A. Sangiovanni-Vincentelli, editors, *Int.l Workshop on Hybrid Systems: Computation and Control*, volume 2034 of *LNCS*, pages 232–246. Springer, 2001.

[127] R. Ghosh and C. Tomlin. Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modelling: Delta-notch protein signalling. *Syst. Biol.*, 1(1), June 2004.

[128] Ronojoy Ghosh, Keith Amonlirdviman, and Claire J. Tomlin. A hybrid system model of planar cell polarity signaling in drosophila melanogaster wing epithelium. In *Proceedings of the 41st IEEE Conference on Decision and Control, Las Vegas*, December 2002.

[129] D. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 8(1):2340–2361, 1977.

[130] M.L. Giuseppin and N.A. van Riel. Metabolic modeling of saccharomyces cerevisiae using the optimal control of homeostasis: a cybernetic model definition. *Metab Eng.*, 2(1):14–33, 2000.

[131] Gene Ontology Consortium GOC. The gene ontology (go) database and informatics resource. *Nucleic Acids Research (Database issue)*, 32, 2004.

[132] I. Goryanin, T.C Hodgman, and E. Selkov. Mathematical simulation and analysis of cellular metabolism and regulation. *Bioinformatics*, 15(9):749–758, 1999.

[133] Daniel R. Grayson and Michael E. Stillman. Macaulay 2, a software system for research in algebraic geometry. Available at http://www.math.uiuc.edu/Macaulay2/.

[134] Mark R. Greenstreet and Ian Mitchell. Reachability analysis using polygonal projections. In *HSCC'99*, volume 1569 of *Lecture Notes in Computer Science*, 1999.

[135] Iva Greenwald. Lin-12/notch signaling: lessons from worms and flies. *Genes and Development*, 12(12):1751–1762, June 1998.

[136] Dima Grigoriev. Complexity of deciding tarski algebra. *Journal of Symbolic Computation*, 5:65–108, 1988.

[137] C. C. Guet, M. B. Elowitz, W. Hsing, and S. Leibler. Combinatorial synthesis of genetic networks. *Science*, 296(5572):1466–1470, 2002.

[138] Esfandiar Haghverdi, Paulo Tabuada, and George J. Pappas. Bisimulation relations for dynamical, control, and hybrid systems. *Theoretical Computer Science*, November 2003.

[139] D. Harel. Statecharts: A visual formalism for complex systems. *Sci. Comput. Programm.*, 8:231–274, 1987.

[140] E. Harel, O. Lichtenstein, and A. Pnueli. Explicit-clock temporal logic. In *Fifth Annual Symposium on Logic in Computer Science*, pages 402–413. IEEE Computer Society Press, 1990.

[141] J. Heintz, M. Roy, and P. Solerno. Sur la complexit du principe de tarski-seidenberg. *Bull. Soc. Math. France*, 118:126, 1990.

[142] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: A Model Checker for Hybrid Systems. In *Software Tools for Technology Transfer*, volume 1, pages 110–122. Springer-Verlag, 1997.

[143] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic Model Checking for Real-time Systems. In *7th Annual IEEE Symposium on Logic in Computer Science*, pages 394–406. IEEE, IEEE Computer Society Press, June 1992.

[144] T.A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's Decidable about Hybrid Automata. In *Symposium on the Theory of Computing (STOC)*, pages 373–382, 1995.

[145] Thomas A. Henzinger, Benjamin Horowitz, Rupak Majumdar, and Howard Wong-Toi. Beyond hytech: Hybrid systems analysis using interval numerical methods. In N. Lynch and B. Krogh, editors, *HSCC*, volume 1790 of *LNCS*, page 130144. Springer-Verlag Berlin Heidelberg, 2000.

[146] Thomas A. Henzinger, Joerg Preussig, and Howard Wong-Toi. Some lessons from the hytech experience. In *Proceedings of the 40th Annual Conference on Decision and Control (CDC)*, pages 2887–2892. IEEE Press, 2001.

[147] Thomas A. Henzinger and Shankar Sastry. *Hybrid Systems-Computation and Control: Proceedings of the First International Workshop, HSCC '98. Lecture Notes in Computer Science 1386*. Springer-Verlag, 1998.

[148] U.S. Department of Energy Human Genome Program HGP. Genomics and its impact on science and society: The human genome project and beyond. http://www.ornl.gov/sci/techresources/Human_Genome/publicat/primer2001, March 2003.

[149] J-HS. Hofmeyr. Metabolic control analysis in a nutshell. In *Proceedings of the Second International Conference on Systems Biology*, pages 291–300, 2001.

[150] Gerard Holzmann. *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley, September 2003.

[151] H. Hong. Quantifier elimination in elementary algebra and geometry

by partial cylindrical algebraic decomposition, version 13. *WWW site* `www.eecis.udel.edu/~saclib`, 1995.

[152] Hoon Hong, Richard Liska, and Stanly Steinberg. Testing stability by quantifier elimination. *J. Symbolic Computation*, 24:161–187, 1997.

[153] Jianghai Hu, Wei-Chung Wu, and Shankar Sastry. Modeling subtilin production in bacillus subtilis using stochastic hybrid systems. In Rajeev Alur and George J. Pappas, editors, *Hybrid Systems: Computation and Control, 7th International Workshop, HSCC 2004, Philadelphia, PA, USA, March 25-27, 2004, Proceedings*, volume 2993 of *Lecture Notes in Computer Science*, pages 417–431. Springer, 2004.

[154] M. Hucka, A. Finney, Herbert M. Sauro, H. Bolouri, J. Doyle, and H. Kitano. The erato systems biology workbench: Enabling interaction and exchange between software tools for computational biology. In *In The Proceedings of the Pacific Symposium on Biocomputing*, 2002.

[155] Inseok Hwang, Hamsa Balakrishnan, Ronojoy Ghosh, and Claire Tomlin. Reachability analysis of delta-notch lateral inhibition using predicate abstraction. *Lecture Notes in Computer Science*, 2552:715–724, Jan 2002.

[156] T. Ideker and D. Lauffenburger. Building with a scaffold: emerging strategies for high- to low-level cellular modeling. *Trends in Biotechnology*, 21(6):255–62, Jun 2003.

[157] Trey Ideker, Timothy Galitski, and Leroy Hood. A new approach to decoding life: Systems biology. *Annual Review of Genomics and Human Genetics*, 2:343–372, September 2001.

[158] B. P. Ingalls. A control theoretic interpretation of metabolic control analysis. http://www.math.uwaterloo.ca/ bingalls/Pubs/con.pdf (submitted), 2005.

[159] Nobuyoshi Ishii, Martin Robert, Yoichi Nakayama, Akio Kanai, and Masaru Tomita. Toward large-scale modeling of the microbial cell for computer simulation. *Journal of Biotechnology*, 113:281–294, 2004.

[160] S. James, P. Nilson, G. James, S. Kjellenberg, and T. Fagerstrom. Luminescence control in the marine bacterium vibrio fischeri: An analysis of the dynamics of lux regulation. *J Mol Biol*, 296(4):11271137, March 2000.

[161] Dorina Jibetean. Algebraic optimization with applications to system theory. PhD Thesis, Department of Mathematics, Vrije University, Amsterdam, 2003.

[162] Mats Jirstrand. Nonlinear control system design by quantifier elimination. *J. Symbolic Computation*, 24:137–152, 1997.

[163] N. Kam, D. Harel, H. Kugler, R. Marelly, A. Pnueli, J. A. Hubbard, and M. J. Stern. Formal modelling of c. elegans development: A scenario-based approach. *Modelling in Molecular Biology*, pages 151–173, 2004.

[164] Na'aman Kam, Irun R. Cohen, and David Harel. The immune system as a reactive system: Modeling t cell activation with statecharts. In *IEEE 2001 Symposia on Human Centric Computing Languages and Environments (HCC'01)*, 2001.

[165] M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M. Hattori. The kegg resources for deciphering the genome. *Nucleic Acids Res.*, 32:277–280, 2004.

[166] K. Kappler, R. Edwards, and L. Glass. Dynamics in high dimensional model gene networks. *Signal Processing*, 83:789–798, 2002.

[167] P.D. Karp, S. Paley, and P. Romero. The pathway tools software. *Bioinformatics*, 18(1):225–232, 2002.

[168] J.P. Keener and J. Sneyd. *Mathematical Physiology.* Springer, New York, 1998.

[169] R.D. King, K.E. Whelan, F.M. Jones, P.G.K. Reiser, C.H. Bryant, S.H. Muggleton, D.B. Kell, and S.G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427:247 – 252, 2004.

[170] Marc W. Kirschner. The meaning of systems biology. *Cell*, 121(4):503–504, May 2005.

[171] H. Kitano. Computational systems biology. *Nature*, Nov 2002.

[172] Hiroaki Kitano. International alliances for quantitative modeling in systems biology. *Molecular Systems Biology*, 1(1), 2005.

[173] Kauffman K.J., P. Prakash, and J.S. Edwards. Advances in flux balance analysis. *Curr. Opin. Biotechnol.*, 14:491–496, 2003.

[174] Steffen Klamt and Jrg Stelling. Stoichiometric analysis of metabolic networks. In *Tutorial at the 4th International Conference on Systems Biology*, 2003.

[175] Kurt W. Kohn. Molecular Interaction Map of the Mammalian Cell Cycle Control and DNA Repair Systems. *Mol. Biol. Cell*, 10(8):2703–2734, 1999.

[176] Kurt W. Kohn, Mirit I. Aladjem, John N. Weinstein, and Yves Pommier. Molecular Interaction Maps of Bioregulatory Networks: A General Rubric for Systems Biology. *Mol. Biol. Cell*, 17(1):1–13, 2006.

[177] Pascal Koiran. My favourte problems. http://perso.ens-lyon.fr/pascal.koiran/problems.html, 1999.

[178] S. A. Kripke. Semantical Considerations in Modal Logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[179] Srikanta P. Kumar and Jordan C. Feidler. Biospice: A computational infrastructure for integrative biology. *OMICS: A Journal of Integrative Biology*, 7(3):225–225, Sep 2003.

[180] A.B. Kurzhanski and P. Varaiya. On ellipsoidal techniques for reachability analysis. part i: External approximations. *Optimization Methods and Software*, 17(2):177–206(30), January 2002.

[181] A.B. Kurzhanski and P. Varaiya. On ellipsoidal techniques for reachability analysis. part ii: Internal approximations box-valued constraints. *Optimization Methods and Software*, 17(2):207–237(31), January 2002.

[182] A.B. Kurzhanski and P. Varaiya. Reachability under uncertainty. In *IEEE Conference on Decision and Control*, August 2002.

[183] A.B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for hybrid dynamics: the reachability problem. In *Proc. MTNS*, 2005.

[184] G. Lafferiere, G. J. Pappas, and S. Sastry. O-minimal Hybrid Systems. *Mathematics of Control, Signals, and Systems*, 13(1):1–21, March 2000.

[185] G. Lafferiere, G. J. Pappas, and S. Yovine. A New Class of Decidable Hybrid Systems. In *Proceedings of the 2nd International Workshop on Hybrid Systems: Computation and Control*, volume 1569 of *LNCS*, pages 137–151. Springer-Verlag, 1999.

[186] Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. Symbolic reachability

computation for families of linear vector fields. *J. Symb. Comput.*, 32(3):231–253, 2001.

[187] Galit Lahav, Nitzan Rosenfeld, Alex Sigal, Naama Geva-Zatorsky, Arnold J Levine, Michael B Elowitz, and Uri Alon. Dynamics of the p53-mdm2 feedback loop in individual cells. *Nature Genetics*, 36:147 – 150, 2004.

[188] R. Lanotte and A. Maggiolo-Schettini. Monotonic hybrid systems. *Journal of Computer and System Sciences*, 2004.

[189] R. Lanotte and S.Tini. Taylor Approximation for Hybrid Systems. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control (HSCC'05)*, volume 3414 of *LNCS*, pages 402–416. Springer, 2005.

[190] D.A. Lauffenburger and J.J. Linderman. *Receptors: Models for Binding, Trafficking, and Signalling*. Oxford University Press, 365 pgs. (1993); 2nd printing (1996), 1996.

[191] Dong-Yup Lee, Young gyun Oh, Hongseok Yoon, Sang Yup Lee, and Sunwon Park. Exploring flux distribution profiles for switching pathways using multi-objective flux balance analysis. *Genome Informatics*, 13:363–364, 2002.

[192] D.Y. Lee, H. Yun, S. Park, and S.Y. Lee. Metafluxnet: the management of metabolic reaction information and quantitative metabolic flux analysis. *Bioinformatics*, 19(16):2144–6, Nov 1 2003.

[193] E. Lee, A. Salic, R. Krüger, R. Heinrich, and M.W. Kirschner. The roles of apc and axin derived from experimental and theoretical analysis of the wnt pathway. *PLoS Biol*, 1(1), 2003.

[194] Ben Lehner, Julia Tischler, and Andrew G. Fraser. Systems biology: where it's at in 2005. *Genome Biology*, 6:338, 2005.

[195] P. Lincoln and A. Tiwari. Symbolic systems biology: Hybrid modeling and analysis of biological networks. In R. Alur and G. Pappas, editors, *Hybrid Systems: Computation and Control HSCC*, volume 2993 of *LNCS*, pages 660–672. Springer, March 2004.

[196] P. Lindskog. *Methods, Algorithms and Tools for System Indentification Based On Prior Knowledge, PhD Thesis.* Department of Electrical Engineering, Linkoping University, Sweden, 1996.

[197] Edison T. Liu. Systems biology, integrative biology, predictive biology. *Cell*, 121(4):505–506, May 2005.

[198] Catherine M. Lloyd, Matt D. B. Halstead, and Poul F. Nielsen. Cellml: its future, present and past. *Progress in Biophysics and Molecular Biology*, 85(2-3):433–450, 2004.

[199] R. Mahadevan and C.H. Schilling. The effects of alternate optimal solutions in constraint-based genome-scale metabolic models. *Metabolic Engineering*, 5:264–276, 2003.

[200] Radhakrishnan Mahadevan, Jeremy S. Edwards, and Francis J. Doyle-III. Dynamic flux balance analysis of diauxic growth in escherichia coli. *Biophysical Journal*, 83:1331–1340, September 2002.

[201] O. Maler and A. Pnueli. Reachability analysis of planar multi-linear systems. In C. Courcoubetis, editor, *Computer Aided Verification: Proc. of the 5th International Conference CAV'93*, pages 194–209, Berlin, Heidelberg, 1993. Springer.

[202] B. Mandelbrot. *The Fractal Geometry of Nature*. Freeman Co., San Francisco, 1982.

[203] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1992.

[204] Dinesh Manocha and John F. Canny. Multipolynomial resultant algorithms. *J. Symbolic Computation*, 15:99–122, 1993.

[205] G. Marnellos and E.Mjolsness. A gene network approach to modeling early neurogenesis in drosophila. In *PSB'98*, 1998.

[206] C.J. Marshall. Specificity of receptor tyrosine kinase signaling: transient versus sustained extracellular signal-regulated kinase activation. *Cell*, 80(2):179–85, 1995.

[207] Mantsika Matooane. Parallel systems in symbolic and algebraic computation. Technical Report UCAM-CL-TR-537, Computer Laboratory, University of Cambridge, June 2002.

[208] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[209] P. Mendes. Biochemistry by numbers: simulation of biochemical pathways with gepasi 3. *Trends in Biochemical Sciences*, 22:361–363, 1997.

[210] M. Minimiari and M.P. Barnett. Solving polynomial equations for chemical problems using grobner bases. *Molecular Physics*, 102(23-24):2521–2535, 2004.

[211] M.L. Minsky. Recursive unsolvability of post's problem of tag and other topics in theory of turing machines. *Ann. of Math.*, 74:437–455, 1961.

[212] B. Mishra. *Algorithmic Algebra*. Springer-Verlag, New York, 1993.

[213] B. Mishra. A Symbolic Approach to Modeling Cellular Behavior. In S. Sahni, V. K. Prasanna, and U. Shukla, editors, *High Performance Computing*, volume 2552 of *LNCS*, pages 725–732. Springer, 2002.

[214] B. Mishra. *Computational Real Algebraic Geometry*, pages 740–764. CRC Press, Boca Raton, FL, 2004.

[215] B. Mishra, M. Antoniotti, S. Paxia, and N. Ugel. Simpathica: A computational systems biology tool within the valis bioinformatics environment. *Computational Systems Biology*, 2005.

[216] B. Mishra and E. M. Clarke. Hierarchical Verification of Asynchronous Circuits Using Temporal Logic. *Theoretical Computer Science*, 38:269–291, 1985.

[217] B. Mishra, R. Daruwala, Y. Zhou, N. Ugel, A. Policriti, M. Antoniotti, S. Paxia, M. Rejali, A. Rudra, V. Cherepinsky, N. Silver, W. Casey, C. Piazza, M. Simeoni, P. Barbano, M. Spivak, J-W. Feng, V. Mysore, O. Gill, F. Cheng, B. Sun, I. Ioniata, T.S. Anantharaman, E.J.A. Hubbard, A. Pnueli, D. Harel, V. Chandru, R. Hariharan, M. Wigler, F. Park, S.-C. Lin, Y. Lazebnik, F. Winkler, C. Cantor, A. Carbone, and M. Gromov. A sense of life: Computational & experimental investigations with models of biochemical & evolutionary processes. *OMICS - A Journal of Integrative Biology (Special Issue on BioCOMP)*, 7(3):253–268, 2003.

[218] Pradyumna Mishra and George J. Pappas. Requiem: a mathematica notebook for exact symbolic reachability computation. http://www.seas.upenn.edu/ hybrid/requiem.html, 2001.

[219] Nicholas A. M. Monk. Oscillatory expression of hes1, p53, and nf-kappab driven by transcriptional time delays. *Current Biology*, 13(16):1409–1413, August 2003.

[220] John A. Morgang and David Rhodes. Mathematical modeling of plant metabolic pathways. *Metabolic Engineering*, 4:80–89, 2002.

[221] V. Mysore and B. Mishra. Algorithmic Algebraic Model Checking III: Approximate Methods. In *Infinity 2005 – The 7th International Workshop on Verification of Infinite-State Systems*, volume 149(1) of *Electronic Notes in Theoretical Computer Science*, pages 61–77, February 2006.

[222] V. Mysore, C. Piazza, and B. Mishra. Algorithmic Algebraic Model Checking II: Decidability of Semi-Algebraic Model Checking and its Applications to Systems Biology. In Doron A. Peled and Yih-Kuen Tsay, editors, *Automated Technology for Verification and Analysis: Third International Symposium, ATVA 2005, Taipei, Taiwan, October 4-7, 2005. Proceedings*, volume 3707 of *LNCS*, pages 217–233. Springer-Verlag, Oct 2005.

[223] V. Mysore and A. Pnueli. Refining the undecidability frontier of hybrid automata. In *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science: 25th International Conference, Hyderabad, India, December 15-18, 2005. Proceedings*, volume 3821 of *Lecture Notes in Computer Science*, pages 261 – 272, Dec 2005.

[224] Venkatesh Mysore, Alberto Casagrande, Carla Piazza, and Bud Mishra. Tolque – A Tool for Algorithmic Algebraic Model Checking. In *The Ninth International Workshop on Hybrid Systems Computation & Control (HSCC06) Poster Session*, March 2006.

[225] Venkatesh Mysore and Bud Mishra. Algorithmic Algebraic Model Checking IV: Metabolic Networks. *Journal of Mathematical Biology (to be submitted)*, 2006.

[226] M. Nagasaki, A. Doi, H. Matsuno, and S. Miyano. Genomic object net: a platform for modeling and simulating biopathways. *Applied Bioinformatics*, 2003.

[227] A.A. Namjoshi and R. Doraiswami. A cybernetic modeling framework for analysis of metabolic systems. *Computers & chemical engineering*, 29(3):487 – 498, 2005.

[228] J.D. Navarro, V. Niranjan, S. Peri, C.K. Jonnalagadda, and A. Pandey. From biological databases to platforms for biomedical discovery. *Trends in Biotechnology*, 21(6):263–8, June 2003.

[229] A. Noack. Ein zbdd-paket fuer effizientes model checking von petri netzen / btu cottbus. Studienarbeit, 1999.

[230] Denis Noble. Modeling the Heart–from Genes to Cells to the Whole Organ. *Science*, 295(5560):1678–1682, 2002.

[231] Maureen A. O'Malley and John Dupré. Fundamental issues in systems biology. *BioEssays*, 27(12):1270–1276, 2005.

[232] J. S. Ostroff. *Temporal Logic of Real-Time Systems*. Research Studies Press, 1990.

[233] Sam Owre and N. Shankar. Writing PVS proof strategies. In Myla Archer, Ben Di Vito, and César Muñoz, editors, *Design and Application of Strategies/Tactics in Higher Order Logics (STRATA 2003)*, NASA Conference Pub-

lication, pages 1–15, Hampton, VA, September 2003. NASA Langley Research Center.

[234] Pablo Parrilo and Sanjay Lall. Semidefinite programming relaxations and algebraic optimization in control. *European Journal of Control*, 9(2-3):307–321, 2003.

[235] Anita S.-R. Pepper, Darrell J. Killian, and E. Jane Albert Hubbard. Genetic analysis of caenorhabditis elegans glp-1 mutants suggests receptor interaction or competition. *Genetics*, 163(1):115–132, 2003.

[236] S. Petitjean. Algebraic geometry and computer vision: Polynomial systems, real and complex roots. *Journal of Mathematical Imaging and Vision*, 10:191–220, 1999.

[237] Andrew Phillips and Luca Cardelli. A graphical representation for the stochastic pi-calculus. In *BioConcur*, 2005.

[238] James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant Kalé, and Klaus Schulten. Scalable molecular dynamics with namd. *Journal of Computational Chemistry*, 26(16):1781–1802, 2005.

[239] C. Piazza, M. Antoniotti, V. Mysore, A. Policriti, F. Winkler, and B. Mishra. Algorithmic Algebraic Model Checking I: The Case of Biochemical Systems and their Reachability Analysis. In Kousha Etessami and Sriram K. Rajamani, editors, *Computer Aided Verification: 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005. Proceedings*, volume 3576 of *LNCS*, pages 5–19. Springer-Verlag, 2005.

[240] Ute Platzer and Hans-Peter Meinzer. Simulation of genetic networks in multicellular context. In *D , Kim J. und Martinez T Polani (ed.), Fifth German workshop on Artificial Life: Abstracting and Synthesizing the Principles of Living Systems*, pages 43–51. Akad. Verl.-Ges., 2002.

[241] A. Puri and P. Varaiya. Decidebility of hybrid systems with rectangular differential inclusions. *Computer Aided Verification*, pages 95–104, 1994.

[242] R. Puzone, B. Kohler, P. Seiden, and F. Celada. Immsim, a flexible model for in machina experiments on immune system responses. *Future Gener. Comput. Syst.*, 18(7):961–972, 2002.

[243] Stefan Ratschan and Zhikun She. Safety verification of hybrid systems by constraint propagation based abstraction refinement. In *HSCC*, 2005.

[244] Aviv Regev, William Silverman, and Ehud Y. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In *Pacific Symposium on Biocomputing*, pages 459–470, 2001.

[245] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals, parts i-iii. *Journal of Symbolic Computation*, 13:352, 1992.

[246] C. Robinson. *Dynamical Systems: Stability, Symbolic Dynamics, and Chaos.* CRC Press, Boca Raton, 1995.

[247] E. Rodriguez-Carbonell and A. Tiwari. Generating polynomial invariants for hybrid systems. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control, HSCC 2005*, LNCS. Springer, March 2005.

[248] L. Rubel and M. Singer. A differentially algebraic elimination theorem with

applications to analog computatbility in the calculus of variations. *American Mathematical Society*, 94(4):635–658, 1985.

[249] T. Rus and E. Van Wyk. Algebraic Implementation of Model Checking. In *Proceedings of the 3rd AMAST Workshop on Real-Time Systems*, Salt lake City, UT, U.S.A., 1996.

[250] Karen Sachs, David Gifford, Tommi Jaakkola, Peter Sorger, and Douglas A. Lauffenburger. Bayesian network approach to cell signaling pathway modeling. In *Sci. STKE*, 2002.

[251] A. Saito and K. Kaneko. Inaccessibility and undecidability in computation, geometry and dynamical systems. *Physica D*, 155:1–33, 2001.

[252] Sriram Sankaranarayanan, Henny Sipma, and Zohar Manna. Constructing invariants for hybrid systems. *Formal Methods in System Design*, July 2004.

[253] C. Sansom. Systems biology: Will it work? *Systems Biology, IEE*, 2(1):1–4, March 2005.

[254] B. D. Saunders, H. R. Lee, and S. K. Abdali. A parallel implementation of the cylindrical algebraic decomposition algorithm. In *ISSAC '89: Proceedings of the ACM-SIGSAM 1989 international symposium on Symbolic and algebraic computation*, pages 298–307, New York, NY, USA, 1989. ACM Press.

[255] Herbert M. Sauro. The computational versatility of proteomic signaling networks. *Current Proteomics*, 1:67–81, 2004.

[256] C.H. Schilling, J.S. Edwards, D. Letscher, and B.O. Palsson. Combining pathway analysis with flux balance analysis for the comprehensive study of metabolic systems. *Biotechnol Bioeng.*, 71(4):286–306, 2000-2001.

[257] C.H. Schilling, S. Schuster, B.O. Palsson, and R. Heinrich. Metabolic pathway analysis: Basic concepts and scientific applications in the post-genomic era. *Biotechnol. Prog.*, 15:296–303, 1999.

[258] Christophe H. Schilling, Markus W. Covert, Iman Famili, George M. Church, Jeremy S. Edwards, and Bernhard O. Palsson. Genome-scale metabolic model of helicobacter pylori. *Journal of Bacteriology*, 184(16), Aug. 2002.

[259] Gerardo Schneider. *Algorithmic Analysis of Polygonal Hybrid Systems. Ph.D. thesis*. VERIMAG - UJF, Grenoble, France, 2002.

[260] Daniel Segre, Dennis Vitkup, and George M. Church. Analysis of optimality in natural and perturbed metabolic networks. *PNAS*, 99(23):15112–15117, November 12 2002.

[261] Natarajan Shankar. Combining theorem proving and model checking through symbolic analysis. In C. Palamidessi, editor, *CONCUR*, volume 1877 of *LNCS*, page 116. Springer-Verlag Berlin, 2000.

[262] David E. Shaw. A fast, scalable method for the parallel evaluation of distance-limited pairwise particle interactions. *Journal of Computational Chemistry*, 26(13):1318–1328, 2005.

[263] T. Shlomi, O. Berkman, and E. Ruppin. Constraint-based modelling of perturbed organisms: A room for improvement. In *ISMB*, 2004.

[264] I. Shmulevich, I. Gluhovsky, R.F. Hashimoto, E.R. Dougherty, and W. Zhang. Steady-state analysis of genetic regulatory networks modelled by probabilistic boolean networks. *Comparative and Functional Genomics*, 4:601–608, 2003.

[265] B. I. Silva, K. Richeson, B. H. Krogh, and A. Chutinan. Modeling and Verification of Hybrid Systems Using CheckMate. In *ADPM*, 2000.

[266] B. I. Silva, O. Stursberg, B. H. Krogh, and S. Engell. An Assessment of the Current Status of Algorithmic Approaches to the Verification of Hybrid Systems. In *Proceedings of the 40th IEEE Conference on Decision and Control*. IEEE, December 2001.

[267] M. Sirava, T. Schfer, M. Eiglsperger, M. Kaufmann, O. Kohlbacher, E. Bornberg-Bauer, and H. P. Lenhof. Biominer - modeling, analyzing, and visualizing biochemical pathways and networks. *Bioinformatics*, 18(2), 2003.

[268] Boris M. Slepchenko, James C. Schaff, Ian Macara, and Leslie M. Loew. Quantitative cell biology with the virtual cell. *Trends in Cell Biology*, 13(11):570–576, November 2003.

[269] P. A. Spiro, J. S. Parkinson, and H. G. Othmer. A model of excitation and adaptation in bacterial chemotaxis. *Proc. Natl. Acad. Sci. USA*, 94:72637268, July 1997.

[270] David Van Der Spoel, Erik Lindahl, Berk Hess, Gerrit Groenhof, Alan E. Mark, and Herman J. C. Berendsen. Gromacs: Fast, flexible, and free. *Journal of Computational Chemistry*, 26(16):1701–1718, 2005.

[271] T. Sturm. Quantifier elimination-based constraint logic programming. Technical Report MIP-0202, Fakultät für Mathematik und Informatik, Universität Passau, 2002.

[272] Thomas Sturm. Real quadratic quantifier elimination in risa/asir. *Research*

Memorandum ISIS-RM-5E, ISIS, Fujitsu Laboratories Limited, 1-9-3, Nakase, Mihama-ku, Chiba-shi, Chiba 261, Japan, September 1996.

[273] Paulo Tabuada and George J. Pappas. Model checking ltl over controllable linear systems is decidable. *Hybrid Systems : Computation and Control, Lecture Notes in Computer Science*, 2623, April 2003.

[274] Paulo Tabuada and George J. Pappas. Linear temporal logic control of linear systems. *IEEE Transactions on Automatic Control*, February 2004.

[275] K. Takahashi, N. Ishikawa, Y. Sadamoto, H. Sasamoto, S. Ohta, A. Shiozawa, F. Miyoshi, Y. Nakayama, and M. Tomita. E-cell2: Multi-platform e-cell simulation system. *Bioinformatics*, 19(13):1727–1729, 2003.

[276] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, second edition, 1948.

[277] Gerald Teschl. Ordinary differential equations and dynamical systems. Lecture Notes from http://www.mat.univie.ac.at/ gerald/ftp/book-ode/index.html, 2004.

[278] A. Tiwari and G. Khanna. Series of Abstraction for Hybrid Automata. In C. J. Tomlin and M. Greenstreet, editors, *Hybrid Systems: Computation and Control*, volume 2289 of *LNCS*, pages 465–478. Springer, 2002.

[279] Ashish Tiwari. Approximate reachability for linear systems. In *In the Proceedings of Hybrid Systems: Computation and Control, HSCC*, 2003.

[280] Ashish Tiwari and Gaurav Khanna. Nonlinear systems: Approximating reach sets. In *HSCC 2004, Univ. of Pennsylvania, Philadelphia*. Springer-Verlag, March 2004.

[281] Alan Turing. On computable numbers, with an application to the entscheidungs problem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

[282] J. J. Tyson and B. Novak. Regulation of the eukaryotic cell cycle: Molecular antagonism, hysteresis, and irreversible transitions. *Journal of Theoretical Biology*, 210:249–263, 2001.

[283] P. Uetz and C.S. Vollert. Protein-protein interactions. *Encyclopedic Reference of Genomics and Proteomics in Molecular Medicine*, 2004.

[284] David W. Ussery and Lars Juhl Jensen. Systems biology: in the broadest sense of the word. *Environmental Microbiology*, 7(4):482, April 2005.

[285] Frits W. Vaandrager and Jan H. van Schuppen. *Hybrid Systems: Computation and Control, Second International Workshop, HSCC'99, Berg en Dal, The Netherlands*. Springer-Verlag, 1999.

[286] Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification. In *Proc. First IEEE Symp. on Logic in Computer Science*, pages 322–331, 1986.

[287] A. Varma and B.O. Palsson. Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type escherichia coli w3110. *Appl Environ Microbiol.*, 60(10):3724–31, Oct 1994.

[288] Diana Visser, Rene van der Heijden, Klaus Mauch, Matthias Reuss, and Sef Heijnen. Tendency modeling: A new approach to obtain simplified kinetic models of metabolism applied to saccharomyces cerevisiae. *Metabolic Engineering*, 2:252–275, 2000.

[289] E. O. Voit. *Computational Analysis of Biochemical Systems. A Pratical Guide for Biochemists and Molecular Biologists*. Cambridge University Press, 2000.

[290] E. O. Voit and M. Savageau. Equivalence between S-systems and Volterra systems. *Mathematical Biosciences*, 78:47–55, 1986.

[291] Aaron Wallack, Ioannis Z. Emiris, and Dinesh Manocha. MARS: A MAPLE/MATLAB/c resultant-based solver. In *International Symposium on Symbolic and Algebraic Computation*, pages 244–251, 1998.

[292] F. Wang and H.-C. Yen. Reachability solution characterization of parametric real-time systems. *Theoretical Computer Science*, 2003.

[293] Jrg R. Weimar. Cellular automata approaches to enzymatic reaction networks. In *ACRI*, pages 294–303, 2002.

[294] V. Weispfenning. Quantifier elimination for real algebra – the cubic case. In *International Symposium on Symbolic and Algebraic Computation*, pages 258–263, 1994.

[295] V. Weispfenning. Quantifier elimination for real algebra – the quadratic case and beyond. *AAECC*, 8:101, 1997.

[296] Volker Weispfenning. Simulation and optimization by quantifier elimination. *J. Symb. Comput.*, 24(2):189–208, 1997.

[297] H. Steven Wiley, Stanislav Y. Shvartsman, and Douglas A. Lauffenburger. Computational modeling of the egf-receptor system: a paradigm for systems biology. *Trends in Cell Biology*, 13(1):43–50, 2003.

[298] F. Winkler. *Polynomial Algorithms in Computer Algebra.* Springer-Verlag, Wien, New York, 1996.

[299] P.J. Woolf and J.J. Linderman. Untangling ligand induced activation and desensitization of g-protein-coupled receptors. *Biophysical Journal*, 84(1):3–13, Jan 2003.

[300] Hitoshi Yanami and Hirokazu Anai. Development of synrac. In *Computer Algebra Systems and Applications, CASA*, 2005.

[301] Tau-Mu Yi, Pablo Iglesias, and Brian Ingalls. Control theory in biology: From mca to chemotaxis. In *Tutorial at ICSB*, 2003.

[302] N. Yildirim. Use of symbolic and numeric computation techniques in analysis of biochemical reaction networks. *International Journal of Quantum Chemistry*, 2005.

[303] S. Yovine. Kronos: A verification tool for real-time systems. *International Journal of Software Tools for Technology Transfer*, 1(1/2):123–133, October 1997.

[304] K. Yugi, Y. Nakayama, and M. Tomita. A hybrid static/dynamic simulation algorithm: Towards large-scale pathway simulation. In *Proceedings of the Third International Conference on Systems Biology*, page 232, December 2002.

[305] Y. Zhestkov, M. Eleftheriou, A. Rayshubskiy, F. Suits, T. J. C. Ward, and B. G. Fitch. Early performance data on the blue matter molecular simulation framework by r. s. germain. *IBM Journal of Research and Development*, 49(2/3), 2005.