Algorithmic Algebraic Model Checking I: Challenges from Systems Biology*

C. Piazza[1], M. Antoniotti[2], V. Mysore[2], A. Policriti[1], F. Winkler[4], and B. Mishra[2,3]

[1] Dept. of Mathematics and Computer Science, University of Udine, Udine, Italy
[2] Courant Institute of Mathematical Sciences, NYU, New York, NY, U.S.A.
[3] Dept. of cell Biology, School of Medicine, NYU, New York, NY, U.S.A.
[4] Research Inst. for Symbolic Computation, J. Kepler University, Linz, Austria
{piazza, policriti}@dimi.uniud.it, winkler@risc.uni-linz.ac.at,
{marcoxa, vm40, mishra}@nyu.edu

Abstract. In this paper, 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 describe a method to compute bounded reachability by combining Taylor polynomials and cylindric algebraic decomposition algorithms. We discuss the power and limitations of the framework we propose and we suggest several possible extensions. We briefly show an application to the study of the Delta-Notch protein signaling system in biology.

1 Prologue

Presently, there is no clear way to determine if the current body of biological facts is sufficient to explain phenomenology. In the biological community, it is not uncommon to assume certain biological problems to have achieved a cognitive finality without rigorous justification. In these particular cases, rigorous mathematical models with automated tools for reasoning, simulation, and computation can be of enormous help to uncover cognitive flaws, qualitative simplification or overly generalized assumptions. Some ideal candidates for such study would include: prion hypothesis, cell cycle machinery (DNA replication and repair, chromosome segregation, cell-cycle period control, spindle pole duplication,

* The work reported in this paper was supported by grants from NSF's Qubic program, NSF's ITR program, Defense Advanced Research Projects Agency (DARPA), Howard Hughes Medical Institute (HHMI) biomedical support research grant, the US Department of Energy (DOE), the US Air Force (AFRL), National Institutes of Health (NIH) and New York State Office of Science, Technology & Academic Research (NYSTAR). F.W. was partially supported by the Austrian Science Foundation (FWF) under the research project DET (P16357-N04). C.P. was partially supported by MIUR FIRB grant RBAU018RCZ and the MIUR PRIN'04 grant 2004013015.

K. Etessami and S.K. Rajamani (Eds.): CAV 2005, LNCS 3576, pp. 5–19, 2005.

etc.), muscle contractility, processes involved in cancer (cell cycle regulation, angiogenesis, DNA repair, apoptosis, cellular senescence, tissue space modeling enzymes, etc.), signal transduction pathways, circadian rhythms (especially the effect of small molecular concentration on its robustness), and many others.

Fortunately, similar issues had been tackled in the past by other disciplines: verification of VLSI circuits, hybrid supervisory controllers, robotics, etc. Yet, biology poses new challenges. The most interesting biology combines unimaginable diversity with an understanding of molecular events in minute detail. A single base-pair change can influence the folding of a protein, and alter the femtosecond dynamics of any of a tangle of interacting macromolecules. Of course, a system of millions of ordinary differential equations (ODEs) and their accurate simulation via numerical integration will not have much effect on uncovering the key biological insights. What sort of natural computational abstractions of biological systems can then be most effective? Can we understand biology by "simulating the biologist, and not biology"?

1.1 Biological Models

The central dogma of biology is a good starting point for understanding a mathematical formalism for biochemical processes involved in gene regulation. This principle states that biochemical information flow in cells is unidirectional—DNA molecules code information that gets transcribed into RNA, and RNA then gets translated into proteins. To model a regulatory system for genes, we must also include an important subclass of proteins (transcription activators), which also affects and modulates the transcription processes itself, thus completing the cycle. We can write down *kinetic mass-action* equations for the time variation of the concentrations of these species, in the form of a system of *ODEs* [10, 14, 23]. In particular, 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:

$$\dot{x}_M = -k_1 x_M + k_3 \frac{1 + \theta y_P^n}{1 + y_P^n} \tag{1}$$

$$\dot{x}_P = -k_2 x_P + k_4 x_M \tag{2}$$

where the superscripted dots denote the time-derivatives.

Each equation above is an algebraic differential equation consisting of two algebraic terms, a positive term representing synthesis and a negative term representing degradation. 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, when $n = 1$, the equations are called *Michaelis-Menten* equations; 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.

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

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

When the number of species becomes large, the complexity of the system of differential equations grows rapidly. The integrability of the system of equations, for example, depends on the algebraic properties of appropriate bracket operations [19, 18]. But, we can approximately describe the behavior of such a system using a *hybrid automaton* [2, 21]. The "flow", "invariant", "guard", and "reset" conditions can be approximated by algebraic systems and the decision procedures for determining various properties of these biological systems can be developed using the methods of symbolic algorithmic algebra.

1.2 Intercellular Communication

Communication between adjacent cells are used by biological systems in coordinating the roles, which can be ultimately assigned to any individual cell. For instance, in both vertebrates as well as invertebrates, lateral inhibition through the Delta-Notch signaling pathway leads to cells, starting initially in uniform distribution, to differentiate into "salt-and-pepper" regular-spaced patterns. In a communication mechanism employing lateral inhibition, two adjacent cells interact by having one cell adopt a particular fate, which in turn inhibits its immediate neighbors from doing likewise. In flies, worms and vertebrates, the transmembrane proteins Notch and Delta (or homologs) mediate the reaction, with Notch playing a receptor with its ligand being a Delta protein on a neighboring cell.

Thus, imagine that one has a description of this system in terms of a state-space, its dynamics (i.e. rules for flows and state transitions), and the subregion of its state space corresponding to a desired property (e.g., fine-grained patterning of cells in a neighborhood). The *first* interesting question would be whether the model adequately predicts that, when started in a biologically reasonable initial state with all the model parameters assuming some known values, this system actually evolves into the subregion encoding the desired properties. If it does, the *second* question to ask would be whether one can completely and succinctly characterize all possible regions ("backward-reachable region") from which the system also evolves into the desired subregion. The volume of such a region, its symmetry and other invariants may tell us quite a lot about those properties of the underlying biological system, which may have attributed to its selective advantages. Furthermore, the model is now amenable to verification by wet-lab experimentation involving the creation and analysis of mutants (in the genes/proteins of relevance), some of which may "live" inside the reachable region and others outside. To answer the first question, a good numerical simulation tool suffices. However, it is less clear how best the second problem should be solved computationally.

In a simplified continuous time model, the changes to the normalized levels of Notch $n_{P,X}$ and Delta $d_{P,X}$ activity in a cell X can be expressed as the ODEs $\dot{n}_{P,X} = \mu[f(\bar{d}_{P,X}) - n_{P,X}]$ and $\dot{d}_{P,X} = \rho[g(n_{P,X}) - d_{P,X}]$, where

$$\bar{d}_{P,X} = \frac{1}{\#\mathcal{N}(X)} \sum_{X' \in \mathcal{N}(X)} d_{P,X'}, \;\; f(x) = \frac{x^k}{a + x^k}, \;\; g(x) = \frac{b}{b + x^h},$$

with $\mathcal{N}(X)$ being the set of neighboring cells of the cell X and μ, ρ, a, $b > 0$, k, $h \geq 1$. Note that f monotonically increases from 0 to 1 and g monotonically decreases from 1 to 0 as x takes increasing value from 0 to ∞ (see Collier et al. [9] for details of the model). Collier et al. concluded that the feedback loop was adequate for generating spatial patterns from random stochastic fluctuations in a population of initially equivalent cells, *provided that feedback is strong enough*. Though they also observed that the model does not account for the longer-range patterns.

In a related computational analysis, Ghosh et al. [11] 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 [9]. Our model described below, shows that the reachable set computed by Ghosh et al. lacks a completeness in description.

2 Technical Preliminaries

2.1 Semi-algebraic Hybrid Automata: Syntax

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 DAEs (differential algebraic equations) appearing in kinetic mass-action models of regulatory, metabolic and signal transduction processes.

Definition 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:*

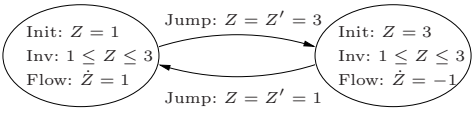
- *$Z = \{Z_1, \ldots, Z_k\}$ a finite set of variables ranging over the reals $\mathbb{R}$; $\dot{Z} = \{\dot{Z}_1, \ldots, \dot{Z}_k\}$ denotes the first derivatives with respect to the time $t \in \mathbb{R}$ during continuous change; $Z' = \{Z'_1, \ldots, Z'_k\}$ denotes the set of values at the 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: $Init(v)$, $Inv(v)$, and $Flow(v)$; the labels $Init(v)$ and $Inv(v)$ are constraints whose free variables are in Z; the label $Flow(v)$ is a constraint whose free variables are in $Z \cup \dot{Z}$;*
- *Each edge $e \in E$ is labeled by "jump" conditions: $Jump(e)$, which is a constraint whose free variables are in $Z \cup Z'$.*

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 $(\mathbb{R}, +, \times, =, <)$). We say that H is* in explicit form *if each $Flow(v)$ is of the form $\bigwedge_{i=1}^{k} \dot{Z}_i = f_i(Z_1, \ldots, Z_k)$.* □

In this paper we consider only semi-algebraic hybrid automata in explicit form. Notice that although, as defined, semi-algebraic hybrid automata in explicit form apply only to the cases where the f_i's of the flow conditions are all polynomials, the definitions can be immediately extended to deal with rational functions instead without significant changes to the basic approach.

Example 1. Consider the following semi-algebraic automaton in explicit form.



The initial mode of this hybrid automaton is shown on the left, where from the starting value of $Z = 1$, Z grows with a constant rate of 1. At time $t = 2$, when the automaton reaches a value of $Z = 3$, it jumps to the other mode on the right. In this second mode, Z wanes with a constant rate of -1 and upon reaching the value of $Z = 1$, it jumps back to the initial mode. □

2.2 Hybrid Automata: Semantics

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 $Flow(v)$. In addition, for any given $r \in \mathbb{R}^k$, we use $Init(v)(r)$ ($Inv(v)(r)$ and $Flow(v)(r)$) to denote the Boolean value obtained by pairwise substitution of r with Z in $Init(v)$ ($Inv(v)$ and $Flow(v)$, respectively). Similarly, for any given r, $s \in \mathbb{R}^k$, we use $Jump(e)(r, s)$ to denote the boolean value obtained by pairwise substitution of r with Z and s with Z' in $Jump(e)$. The semantics of hybrid automata can now be given in terms of execution traces as in the definition below.

Definition 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 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, $\rightarrow_C$, *between admissible locations is defined as follows:*

$$\langle v, r\rangle \rightarrow_C \langle v, s\rangle$$
$$\textit{iff} \quad \exists t > 0, f \in \Phi(v)\Big(f(0) = r \ \wedge \ f(t) = s \ \wedge \ \forall t' \in [0,t](\mathit{Inv}(v)(f(t')))\Big).$$

The discrete reachability transition relation, $\rightarrow_D$, *between admissible locations is defined as follows:*

$$\langle v, r\rangle \rightarrow_D \langle u, s\rangle \qquad \textit{iff} \qquad \langle v, u\rangle \in E \ \wedge \ \mathit{Jump}(\langle v, u\rangle)(r, s)$$

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

$\forall i \geq 0 \ \ \ell_i \rightarrow_C \ell_{i+1} \ \vee \ \ell_i \rightarrow_D \ell_{i+1}$. □

2.3 The Bounded Reachability Problem

Let H be a semi-algebraic k-dimensional hybrid automaton in explicit form, $S \subseteq \mathbb{R}^k$ be a set of "start states", characterized by the first order formula $\mathbb{S}(Z)$, and $B \subseteq \mathbb{R}^k$ be a set of "bad states", characterized by the first order formula $\mathbb{B}(Z)$. We wish to check that there exists no trace of H starting from a location of the form $\langle v, s\rangle$ with $s \in S$ and reaching a location of the form $\langle u, b\rangle$ with $b \in B$ within a specified time interval $[0, \mathsf{end}]$. If such traces exist we are interested in a characterization of the points of S which reach B in the time interval $[0, \mathsf{end}]$.

Note that for our applications of interest, it suffices to place an upper-bound on the time interval.

3 Our Approach

In this paper, we explore solutions to the bounded-reachability problem through symbolic computation methods, applied to the descriptions of the traces of the hybrid automaton. Because the description of the automaton is through semi-algebraic sets, the evolution of the automaton can be described even in cases where system parameters and initial conditions are unspecified. Nonetheless, semialgebraic 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" that we propose here naturally generalizes many of the basic ideas inherent to BDD-based symbolic model checking or even the more recent SAT-based approaches.

Nonetheless, our method has an inherent incompleteness: we proceed on the traces using a time step δ which implies that our answer is relative to a limited time interval. Furthermore, when the solutions of the differential equations

cannot be computed we approximate them using the first few terms of the corresponding Taylor polynomials, hence the error we accumulate depends on δ.

We start by presenting how our method applies to the case of a system of differential equations, i.e., a hybrid automaton with only just one mode and no *Init*, *Inv*, and *Jump* conditions.

3.1 The Basic Case

Consider a system of differential equations of the form $\dot{Z} = f(Z)$, where $\dot{Z}$ and Z are vectors of length k and f is a function that operates on them.

Let S, $B \subseteq \mathbb{R}^k$ be characterized by the formulæ $\mathbb{S}(Z)$ and $\mathbb{B}(Z)$, respectively. As before, let $[0, \mathsf{end}]$ be a time interval and $0 < \delta \leq \mathsf{end}$ be a time step.

We use $pj(Z0, \delta)$ to denote the Taylor polynomial of degree j relative to the solution $Z(t)$ centered in $Z0$ with a step size of δ. For instance, $p1(Z0, \delta)$ is the vector expression $Z0 + f(Z0) \cdot \delta$.

Consider the following first-order formula over the reals

$$\mathbb{F}_\delta(Z0, Z) \equiv \mathbb{S}(Z0) \ \wedge \ \exists \delta' \Big(Z = pj(Z0, \delta') \ \wedge \ 0 \leq \delta' \leq \delta \Big).$$

The points reachable from S in the time interval $[0, \delta]$ can be approximated with the set of points satisfying the formula $\exists Z0(\mathbb{F}_\delta(Z0, Z))$. Hence, the points in B and reachable from S in $[0, \delta]$ can be approximated by the formula

$$\exists Z0(\mathbb{F}_\delta(Z0, Z)) \ \wedge \ \mathbb{B}(Z).$$

Symbolic algebraic techniques can be applied in order to both simplify (e.g., eliminate quantifiers) and decide the satisfiability of this formula. If the formula is satisfiable, then the values of Z for which the formula is true represent the portion of B that can be reached in time $\delta' \leq \delta$. Similarly, the points in S which reach any point in B within the time interval $[0, \delta]$ can be characterized by the formula $\exists Z(\mathbb{F}_\delta(Z0, Z) \ \wedge \ \mathbb{B}(Z))$. If these formulæ are not satisfiable then we can proceed with a second step, getting the formula

$$\mathbb{F}_{2\delta}(Z0, Z) \equiv \mathbb{S}(Z0) \wedge \exists Z1, \delta'(Z1 = pj(Z0, \delta) \ \wedge \ Z = pj(Z1, \delta') \ \wedge \ 0 \leq \delta' \leq \delta).$$

The above reasoning can now be applied to $\mathbb{F}_{2\delta}(Z0, Z)$, i.e., use $\mathbb{F}_{2\delta}(Z0, Z)$ instead of $\mathbb{F}_\delta(Z0, Z)$, to check if S reaches B within the time interval $[0, 2\delta]$, etc. Notice that the new variable $Z1$ which occurs in $\mathbb{F}_{2\delta}(Z0, Z)$ can be eliminated by applying substitutions. If after time end all the formulæ we generate are unsatisfiable, then S cannot reach B within the time interval $[0, \mathsf{end}]$.

It is important to notice that: (1) The only approximation we have introduced is due to the use of the Taylor polynomials; (2) We have only used existential quantified formulæ; (3) The degree of the Taylor polynomial together with the degrees of the f_i's influence the complexity of the first-order formulæ we create and the number of steps needed to get a sufficient precision. As far as the approximation issues are concerned, when the derivative of order $j+1$ of f is bounded we can use the Lagrange Remainder Theorem to both under and over approximate the set of points reachable within the time interval $[0, \mathsf{end}]$ and to estimate

the error. It is easy to see that our method can be generalized to the case in which the f_i's are rational functions, i.e., ratios of polynomial functions. In fact, in this case we only have to preprocess the formulæ by computing the LCMs of the denominators and using them to get formulæ over polynomial functions.

When we terminate, we are left with deciding the satisfiability of a semialgebraic formula involving $n = 2 + k \cdot \lceil \mathsf{end}/\delta \rceil + N(\mathbb{S}) + N(\mathbb{B})$ variables in degree $d = \max[j + \deg(f), \deg(\mathbb{S}), \deg(\mathbb{B})]$, where N and deg denote the number of variables and total degree, respectively used in the semialgebraic description of $\mathbb{S}$ and $\mathbb{B}$. In addition, if we assume that the coefficients of the polynomials can be stored with at most L bits, then the total time complexity (bit-complexity) [17, 20, 24] of the decision procedure is $(L \log L \log\log L) d^{O(n)}$. We note that even with low degree polynomials, this exponential complexity in the number of variables makes it impractical to test for bounded-reachability even when the specified time interval is relatively short. Here we focus on rather simple examples where the complexity is rather manageable, and is achieved by approximating polynomial and rational functions by piecewise linear functions.

Example 2. Next, examine the following toy example. The following system of differential equations describes the dynamics $\dot{Z} = 2Z^2 + Z$, with S and B characterized by $\mathbb{S} \equiv Z > 4$ and $\mathbb{B} \equiv Z^2 < 4$. Now, consider the time interval $[0, 0.5]$ and time step 0.5. After time 0.5, using an approximation with Taylor polynomial of degree 2, we derive the formula

$$\exists Z0, \delta' \Big(Z0 > 4 \wedge Z = Z0 + (2Z0^2 + Z0) \cdot \delta' + (8Z0^3 + 6Z0^2 + Z0) \cdot (\delta')^2/2 \\ \wedge\, 0 \leq \delta' \leq 0.5 \wedge Z^2 < 4 \Big).$$

This formula is unsatisfiable, thus implying that the dynamical system reaches no bad states in the specified time interval $[0, 0.5]$. □

The formulæ involved in our method can be easily simplified, if we introduce further approximations. For instance, we may approximate reachability by first evaluating the maxima and the minima of the j-th Taylor polynomial $pj(Z, \delta')$s over S and $[0, \delta]$, and then using them as upper and lower bounds.

Example 3. Next, consider the differential equation $\dot{Z} = 2Z$, with S and B characterized by $\mathbb{S} \equiv 2 \leq Z \leq 4$ and $\mathbb{B} \equiv 3 < Z < 5$.

The Taylor polynomial of degree 1 with $\delta = 0.5$ is $Z + 2Z \cdot \delta'$, i.e., $2Z$. Note that since the maximum and the minimum in S are 8 and 4, respectively, and since the interval $[4, 8]$ intersects $(3, 5)$, S reaches B in time 0.5. □

3.2 The General Case

We are ready to deal with the general case, where we have a polynomial k-dimensional hybrid automaton H in explicit form.

Given a mode v of H, we use the notation $pj_v(Z, \delta)$ to denote the Taylor polynomial of degree j in the mode v centered in Z. The first-order formula

$$\mathbb{F}[v, \mathbb{S}](Z0, Z) \equiv \mathbb{S}(Z0) \ \wedge \ \exists \delta' \Big(Z = pj_v(Z0, \delta') \ \wedge \ 0 \leq \delta' \leq \delta \\ \wedge \forall \delta'' \big(0 \leq \delta'' \leq \delta' \rightarrow Inv(v)(pj_v(Z0, \delta'')) \big) \Big)$$

characterizes the points reached within time δ in the mode v, under the approximation implied by the use of the Taylor polynomial. Notice that, if we assume that the invariant regions are convex and we use the Taylor polynomial of degree 1, we can avoid the universal quantification. As before, the formula

$$\exists Z0 (\mathbb{F}[v, \mathbb{S}](Z0, Z)) \ \wedge \ \mathbb{B}(Z)$$

is satisfiable if and only if the set B can be reached from S without leaving mode v within the time step δ. In this case, the points of S which reach B are characterized by $\exists Z(\mathbb{F}[v, \mathbb{S}](Z0, Z) \ \wedge \ \mathbb{B}(Z))$. If the preceding formula is not satisfiable, we have to consider all possible alternative situations: that is, either we continue to evolve within the mode v or we discretely jump to another mode, $u \in V$. We define the formula $\mathbb{S}_\delta^{vu}$

$$\mathbb{S}_\delta^{vu}(Z) \equiv \begin{cases} \exists Z0 (\mathbb{F}_\delta^v(Z0, Z), & \text{if } u = v; \\ \exists Z0, Z1 (\mathbb{F}_\delta^v(Z0, Z1) \wedge Jump(\langle v, u \rangle)(Z1, Z)), & \text{otherwise.} \end{cases}$$

representing the states reached within time δ in the mode 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 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. Let $\mathbb{M}(Z) \equiv \mathbb{S}^{v,u\ldots,w}(Z)$ be one of the formulæ obtained after some number of iterations. Suppose now that we intend to jump from this mode w to the next mode z. We will then need to check whether the minimum amount of time has passed before the jump can be taken. Consider the formula:

$$\mathbb{T}(w, z, \mathbb{M})(T) \equiv \exists Z0, Z1, Z \Big(\mathbb{M}(Z0) \ \wedge \ Z1 = pj_w(Z0, T) \ \wedge \ 0 \leq T \leq \delta \\ \wedge \forall T' (0 \leq T' \leq T \rightarrow Inv(v)(pj_v(Z0, T'))) \ \wedge \ Jump(\langle w, z \rangle)(Z1, Z) \Big).$$

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

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

To avoid Zeno paths we could eliminate the paths in which the minimum is 0. Along 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 B is never reached we can be sure that B cannot be reached from S in the time interval $[0, \mathsf{end}]$. If B is reached, i.e., one of the formulæ involving $\mathbb{B}$ is satisfiable before m iterations, then we can be sure that B is reachable from S in the time interval $[0, \mathsf{end}]$. If B is reached after the first m iterations, then B is reachable from 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.

Notice that 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, we can again exploit the Lagrange Remainder Theorem to both under and over approximate the set of reachable points and to estimate errors (see [15]).

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 semialgebraic 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) [17, 20, 24] of the decision procedure is $(L \log L \log\log L) d^{2^{O(n)}}$, i.e., double-exponential in the number of variables.

3.3 Rectangular Regions

When the formulæ $\mathit{Init}(v)$s, $\mathit{Inv}(v)$s, $\mathit{Jump}(e)$s, $\mathbb{S}$, and $\mathbb{B}$ identify rectangular (closed) regions (e.g., product of intervals) we can rely on other approaches from symbolic computations, while achieving further simplifications along the way.

Given a mode v of H, the region obtained from the intersection of $\mathit{Inv}(v)$ and $\mathbb{S}$ is of the form $\mathbb{R}(v) \equiv a(v) \leq Z \leq b(v)$. We can symbolically determine the maximum $max(v)$ and the minimum $min(v)$ of $pj_v(Z, \delta')$ over $\mathbb{R}(v) \times [0, \delta]$. We can use the following formula to over-approximate the points reached within the time interval $[0, \delta]$:

$$\mathbb{O}v(Z) \equiv \min(v) \leq Z \leq max(v) \wedge \mathit{Inv}(v)(Z).$$

The formulæ $\mathbb{O}v(Z) \wedge \mathbb{B}(Z)$ and $\mathbb{O}v(Z1) \wedge \mathit{Jump}(\langle v, u\rangle)(Z1, Z)$, can be used to check if the set B is reached or if it is possible to jump to another mode. Since these formulæ identify rectangular regions, we can iterate the method.

4 A Case Study: the Delta-Notch Protein Signaling

Let us now return to the Delta-Notch protein signaling system that we had introduced earlier. The mathematical model for the Delta-Notch signaling presented in [9] can be approximated by piecewise linear functions and results in a rectangular hybrid automaton that can be analyzed symbolically.

For instance, in [11] a rather simple piecewise linear hybrid automaton model was created, and was extensively studied through the predicate abstraction method of [22]. The piecewise affine hybrid automaton of [11] is defined by: (1) A set of *global invariant conditions* which must be always true; (2) A finite number of modes; (3) Each mode is characterized by a set of *local invariant conditions* and a set of *differential equations* determining the flow of the variables.

The automaton modeling the evolution of a one-cell system has been described using the SAL language [7] in [11]. In this description, all the fluxes have been reversed in order to determine the set of initial conditions from which a particular steady-state is reached (by solving a forward reachability problem). The automata relative to the two and four cell systems have also been similarly studied. Here we consider the two-cell piecewise affine hybrid automaton and apply our method to the forward reachability problem. For a complete description of the automaton we refer the reader to [11].

The system representing the evolution of two cells presented in [11] has the following set of invariant conditions

$$\begin{aligned} & 0 \leq d_1, d_2 \leq R_D/\lambda_D \ \wedge \ 0 \leq n_1, n_2 \leq R_N/\lambda_N \\ \wedge \ & -R_N/\lambda_N \leq h_D \leq 0 \ \wedge \ 0 \leq h_N \leq R_D/\lambda_D. \end{aligned}$$

The variables d_1 and d_2 represent the concentration of the Delta protein in the first and in the second cell, respectively. The variables n_1 and n_2 represent the concentration of the Notch protein in the first and in the second cell, respectively. R_D and R_N are constants representing the Delta and Notch production rates, respectively. λ_D and λ_N are the Delta and Notch protein decay constants, respectively. h_D is an unknown switching threshold which determines the Delta protein production. h_N, similar to h_D, is an unknown switching threshold which determines the Notch protein production.

A possible equilibrium for the system is given by the point $d_1^* = 0$, $n_1^* = R_N/\lambda_N$, $d_2^* = R_D/\lambda_D$, $n_2^* = 0$, which belongs to the mode v characterized by the following invariant and flow conditions

$$\begin{gathered} 0 \leq d_1 \leq h_N \ \wedge \ -h_D \leq n_1 \leq R_N/\lambda_N \ \wedge \ h_N \leq d_2 \leq R_D/\lambda_D \ \wedge \ 0 \leq n_2 \leq -h_D, \\ \dot{d_1} = \lambda_D d_1 \wedge \dot{n_1} = -R_N + \lambda_N n_1 \wedge \dot{d_2} = -R_D + \lambda_D d_2 \wedge \dot{n_2} = \lambda_N n_2. \end{gathered}$$

We apply our method to the analysis of the admissible locations reachable from v. In particular, in this case we can apply the simplifications described in Section 3.3. Even if we limit our attention to one possible evolution with relatively few iterations, this suffices to compute a somewhat different result from what is presented in [11].

The formula $\mathbb{O}v(\langle d_1, n_1, d_2, n_2\rangle)$ representing the points reached in the time interval $[0, \delta]$ is

$$0 \leq d_1 \leq h_N + \lambda_D \cdot h_N \cdot \delta \ \wedge \ -h_D - R_N \cdot \delta - \lambda_N \cdot h_D \cdot \delta \leq n_1 \leq R_N/\lambda_N \ \wedge$$
$$h_N - R_D \cdot \delta + \lambda_D \cdot h_N \cdot \delta \leq d_2 \leq R_D/\lambda_D \ \wedge \ 0 \leq n_2 \leq -h_D - \lambda_N \cdot h_D \cdot \delta.$$

Consider a mode u characterized by the following invariant conditions

$$h_N \leq d_1 \leq R_D/\lambda_D \wedge -h_D \leq n_1 \leq R_N/\lambda_N \wedge h_N \leq d_2 \leq R_D/\lambda_D \wedge 0 \leq n_2 \leq -h_D.$$

Since the formula $\mathbb{O}(v) \ \wedge \ Inv(u)$ is satisfiable we can jump to the mode u. In particular, assuming that δ is so chosen that $h_N + \lambda_D \cdot h_N \cdot \delta \leq R_D/\lambda_D$, in the interval $[0, \delta]$, we can reach the points satisfying

$$h_N \leq d_1 \leq h_N + \lambda_D \cdot h_N \cdot \delta \ \wedge \ -h_D \leq n_1 \leq R_N/\lambda_N$$
$$\wedge \ h_N \leq d_2 \leq R_D/\lambda_D \ \wedge \ 0 \leq n_2 \leq -h_D.$$

This formula in conjunction with $d_1 < d_2$ is easily seen to be satisfiable. For instance, one can prove that with $R_N = R_D = \lambda_N = \lambda_D = 1.0$ and $-h_D = h_N = 0.5$ and starting from v with values $\langle 0.5, 0.89, 0.68, 0.42\rangle$, at time 0.5, we can reach $\langle 0.84, 0.81, 0.47, 0.04\rangle$. In [11], it was proven that all the points satisfying $d_1 < d_2 \wedge n_1 > n_2$ are reachable from the stable equilibrium state belonging to v. Our observation, which does not contradict this result of [11], nonetheless proves that our method can be combined with that of [11] to obtain better approximations of the region reachable from the equilibrium in v.

5 Related Literature, Future Work and Conclusions

To place the results described here in the context of a large existing and continually growing literature, we mention a few related results.

In [4] 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 [6]), 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 [8]), 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 [22], 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 [11], the method was applied on piecewise linear hybrid automata to study the Delta-Notch signaling process. In a sequel, we will explain connections and differences between these and our methods.

Recently in [3], 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 our 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.

The 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.

Our plan is to address these problems in a sequence of papers that will form sequels to the current paper: "Algorithmic Algebraic Model Checking (AAMC) series." An incomplete, and evolving list of topics that will be addressed are as follows: generalization to the dense time logic TCTL [1, 12]; decidability issues in this context and under various reasonable models of computation [16]; state-space discretization and predicate abstractions; "quasi-static simulation," combining flux-balanced analysis with slow dynamics; a topological characterization of bio-chemical processes, etc.

The present status of this project is as described below: There is a preliminary implementation of the algorithms in *C/C++*: part of the software system Tolque, the algebraic model checker for semi-algebraic hybrid automata. As it gets integrated with our *Lisp*-based Systems Biology tool Simpathica[5], it will allow biochemical networks to be easily represented, stored and analyzed. The resulting technology is hoped to provide a simple framework for biologists to think about biology and computer scientists to think about how biologists think about biology.

References

1. 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.

2. 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.
3. R. Alur, T. Dang, and F. Ivancic. Progress on Reachability Analysis of Hybrid Systems Using Predicate Abstraction. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control*, volume 2623 of *LNCS*, pages 4–19. Springer, 2003.
4. 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.
5. M. Antoniotti, A. Policriti, N. Ugel, and B. Mishra. Reasoning about Biochemical Processes. *Cell Biochemistry and Biophysics*, 38:271–286, 2003.
6. 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.
7. S. Bensalem, V. Ganesh, Y. Lakhnech, C. Muñoz, 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, *NASA Langley Formal Methods Workshop*, pages 187–196, 2000.
8. L.Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer-Verlag, 1997.
9. 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.
10. 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.
11. A. Cornish-Bowden. *Fundamentals of Enzyme Kinetics (3rd edn.)*. Portland Press, London, 2004.
12. 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*, volume 2623 of *LNCS*, pages 233–248. Springer, 2003.
13. 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.
14. H. Hong. Quantifier elimination in elementary algebra and geometry by partial cylindrical algebraic decomposition, version 13. *WWW site* `www.eecis.udel.edu/∼saclib`, 1995.
15. J.P. Keener and J. Sneyd. *Mathematical Physiology*. Springer-Verlag, New York, 1998.
16. 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.
17. B. Mishra. *Algorithmic Algebra*. Springer-Verlag, New York, 1993.
18. B. Mishra. *Computational Differential Algebra*, pages 111–145. World-Scientific, Singapore, 2000.

19. 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.
20. B. Mishra. *Computational Real Algebraic Geometry*, pages 740–764. CRC Press, Boca Raton, FL, 2004.
21. A. Nerode and W. Kohn. Hybrid Systems and Constraint Logic Programming. In D. S. Warren, editor, *International Conference on Logic Programming (ICLP'93)*, pages 18–24. MIT Press, 1993.
22. 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.
23. E. O. Voit. *Computational Analysis of Biochemical Systems. A Pratical Guide for Biochemists and Molecular Biologists*. Cambridge University Press, 2000.
24. F. Winkler. *Polynomial Algorithms in Computer Algebra*. Springer-Verlag, Wien, New York, 1996.