

# Automated Symbolic Reachability Analysis; with Application to Delta-Notch Signaling Automata\*

Ronojoy Ghosh<sup>1</sup>, Ashish Tiwari<sup>2</sup>, and Claire Tomlin<sup>1</sup>

<sup>1</sup> Stanford University, Stanford, CA, USA  
{ronojoy,tomlin}@stanford.edu

<sup>2</sup> Computer Science Laboratory, SRI International, Menlo Park, CA, USA  
tiwari@csl.sri.com

**Abstract.** This paper describes the implementation of predicate abstraction techniques to automatically compute symbolic backward reachable sets of high dimensional piecewise affine hybrid automata, used to model Delta-Notch biological cell signaling networks. These automata are analyzed by creating an abstraction of the hybrid model, which is a finite state discrete transition system, and then performing the computation on the abstracted system. All the steps, from model generation to the simplification of the reachable set, have been automated using a variety of decision procedure and theorem-proving tools. The concluding example computes the reach set for a four cell network with 8 continuous and 256 discrete states. This demonstrates the feasibility of using these tools to compute on high dimensional hybrid automata, to provide deeper insight into realistic biological systems.

## 1 Introduction

Reachability analysis may be used to gain insight into the behavior of the physical system modeled by a hybrid automaton. In the context of hybrid automaton models of biological networks, the backward reachable sets from the equilibria of the automaton are of considerable interest, because they contain the initial conditions from which a particular biologically significant steady state is attainable. If the reachability analysis is performed on a model with symbolic parameters and rate constants, the computed reachable sets will not depend on numerical instantiations of those parameters. This is particularly important in biological systems, where the exact values of switching thresholds and chemical reaction rates might be unknown, but a range of possible values, usually expressed in terms of other symbolic constants, can be inferred. This in turn may be used to “reverse engineer” parts of a biological circuit model, by attaining through analysis parameters which are difficult or impossible to obtain experimentally.

The goal of this paper is to present our work in *automated symbolic backward reachable set computations for high-dimensional (in both continuous and discrete*

---

\* This research is supported by the DARPA Bio:Info:Micro program, grant MDA972-00-1-0032, and the DARPA BioSpice program under contract DE-AC03-765F00098.

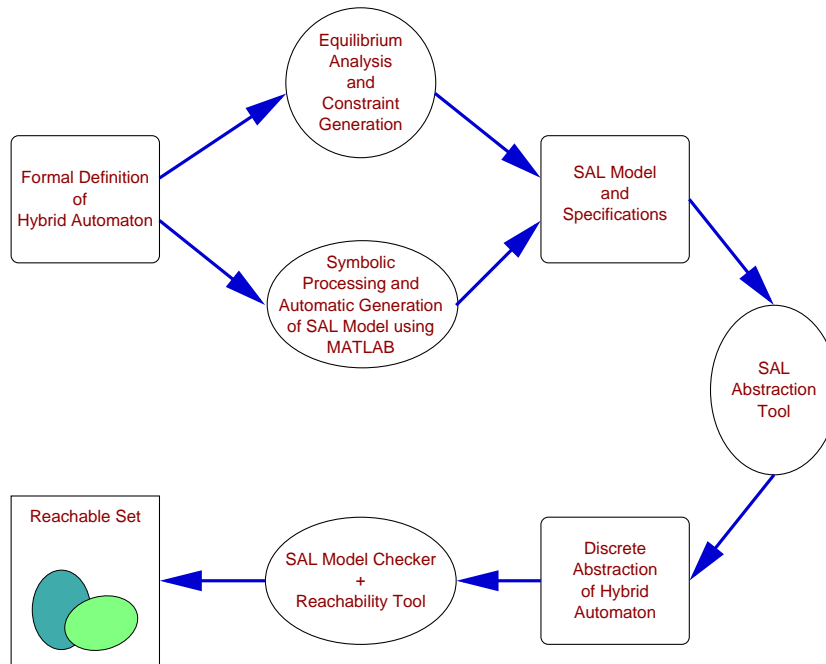
variables) *hybrid automata with multiple equilibria*. The motivating example is the Delta-Notch cell signaling system described and analyzed in [10]. The hybrid automaton is suitably high-dimensional, with  $2n$  continuous variables and  $4^n$  modes, for an  $n$  cell network. The continuous dynamics are governed by piecewise affine differential equations and mode switching occurs only through the continuous state variables crossing switching hyperplanes. Both the single ( $n = 1$ ) and the two cell ( $n = 2$ ) hybrid automata were analyzed in [10], to obtain constraints on the range of the protein kinetic parameters and switching thresholds for biologically feasible equilibria to exist. The two cell automaton was also shown to have a Zeno state with a particular Zeno execution. These results were validated through extensive simulations, but a formal backward reachable set computation of even the two cell system was lacking, due to the difficulty of computing and keeping track of the 4 dimensional reachable set across switching boundaries.

Computing reachable sets for hybrid systems is hard, due to the difficulty of representing and propagating sets in high dimensional continuous spaces. There has been a recent research focus on techniques which use approximations of various types to make the problem of computing reachable sets tractable; these include the use of linear hybrid automata [12, 20], polyhedral representations [3, 7], piecewise affine systems [4], ellipsoidal approximations [6], and projections of convergent approximations [18, 19]. Recently, qualitative simulation models [14, 15] have been proposed to abstract continuous phase portraits of hybrid automata to simpler transition graphs, on which reachability analysis can be performed. Predicate abstraction [1, 11] and quantifier elimination [24] have been proposed for computing discrete abstractions of hybrid automata. Predicate abstraction provides a means for combining theorem proving and model checking techniques by automatically mapping an infinite state system to a finite state system (called the abstract system), in which the states correspond to truth assignments to a set of user-supplied predicates. Most existing tools suffer from two disadvantages from our point of view: (a) the complexity of the computations on the hybrid automaton restricts its dimensionality, and (b) symbolic computations are not possible. The last two computation methods mentioned above circumvent these restrictions by performing the formal verification on a discrete abstraction.

In this paper, we demonstrate the application of these predicate abstraction methods to automated reachable set computation. We use quantifier elimination based decision procedures, implemented using the Symbolic Analysis Laboratory (SAL) [24], which is a framework for combining different tools for abstraction, program analysis, theorem proving, and model checking of transition systems, to abstract the hybrid automaton to a discrete finite-state automaton. The reachable set computations are then performed on the discrete abstraction. The entire process, from generating the SAL model for an arbitrarily large cell network, to obtaining the final reachable sets, is completely automated using MATLAB code written by us, the SAL tool set, and QEPCAD [13] (Quantifier Elimination by Partial Cylindrical Algebraic Decomposition), which is used to simplify

the formula defining the reachable set through quantifier elimination by partial cylindrical algebraic decomposition. However, some analysis necessary to generate the parameter constraints for equilibrium existence was done manually. Techniques to automate those analysis procedures is a current research topic. We apply this automatic computation to the reachability analysis of the Delta-Notch cell signaling system: an affine hybrid automaton with eight continuous variables and 256 discrete states. The computation time for this is about eight hours on a Pentium III 500MHz machine, and there are good prospects to go to higher dimensions.

The reachable set computation follows the procedure (Fig. 1(a)): (i) automated generation of the SAL code specifying the hybrid automaton for an arbitrarily large cell network, (ii) automated abstraction of the hybrid automaton into a discrete finite state automaton (FSA), (iii) automated computation of the backward reachable set using the reachability tool, and (iv) simplification of the reachable set definition using QEPCAD. In this paper, we start by briefly describing the Delta-Notch signaling pathway in cells, the hybrid automaton model of that process (for more details, see [10]), and the relevant analytical results that are used in the abstraction. We then describe the tool as outlined above, followed by results for large hybrid automata for two and four cell networks. Each step in the process is explained using the automaton for a single cell as an illustrative example.



**Fig. 1.** Computation process for reach set construction

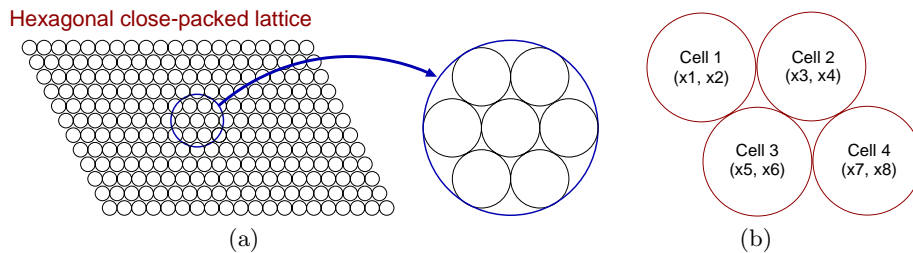
## 2 Delta-Notch Protein Signaling

### 2.1 Biological Background

Delta and Notch are both transmembrane proteins that actively signal only when cells are in direct contact, in a densely packed epidermal layer. Delta is a ligand that binds and activates its receptor Notch in neighboring cells. The activation of Notch in a cell affects the production of Notch ligands (i.e. Delta) both in itself and its neighbors, thus forming a feedback control loop. In the case of lateral inhibition, high Notch levels suppress ligand production in the cell and thus a cell producing more ligands forces its neighboring cells to produce less. The Delta-Notch signaling mechanism has been found to cause pattern formation in many different biological systems, such as the South African claw-toed frog (*Xenopus laevis*) embryonic skin [16] studied here. An example of the distinctive “salt-and-pepper” pattern formed due to lateral inhibition in the *Xenopus* epidermal layer where a regular set of ciliated cells form within a matrix of smooth epidermal cells.

### 2.2 Hybrid Automaton Model

To model the regulation of intracellular Delta and Notch protein concentrations through the feedback network, experimentally observed rules governing the biological phenomenon have to be implemented. First, cells have to be in direct contact for Delta-Notch signaling to occur. This implies that a cell is directly affected by, and directly affects in turn, only immediate neighbors. Second, Notch production is turned on by high Delta levels in the immediate neighborhood of the cell and Delta production is switched on by low Notch concentrations in the same cell. Third, at steady state, a cell with high Delta levels must have low Notch level and vice versa. Finally, both Delta and Notch protein concentrations decay exponentially. In the model, the cells are assumed to be hexagonal close packed, i. e. each cell has six neighbors in contact with it (Fig. 2(a)).



**Fig. 2.** (a) Hexagonal close-packed layout scheme for cells in two dimensional arrays. (b) Layout and variable associations of a four cell Delta-Notch signaling network

Each biological cell is modeled as a four state piecewise affine hybrid automaton. The four states capture the property that Notch and Delta protein

production can be individually switched on or off at any given time. It is assumed that there is no command-actuation delay in the mode switching. The formal definition of the hybrid automaton is given by:

$$\begin{aligned}
H_1 &= (Q_1, X_1, \Sigma_1, V_1, Init_1, f_1, Inv_1, R_1) \\
Q_1 &= \{q_1, q_2, q_3, q_4\} \\
X_1 &= (x_1, x_2)^T \in \mathfrak{R}^2 \\
\Sigma_1 &= \left\{ u_D, u_N : u_D = -x_2, u_N = \sum_{i=1}^6 x_{1,i} \right\} \\
V_1 &= \emptyset \\
Init_1 &= Q_1 \times \{X_1 \in \mathfrak{R}^2 : x_1, x_2 > 0\} \\
f_1(q, x) &= \begin{cases} [-\lambda_D x_1; -\lambda_N x_2]^T & \text{if } q = q_1 \\ [R_D - \lambda_D x_1; -\lambda_N x_2]^T & \text{if } q = q_2 \\ [-\lambda_D x_1; R_N - \lambda_N x_2]^T & \text{if } q = q_3 \\ [R_D - \lambda_D x_1; R_N - \lambda_N x_2]^T & \text{if } q = q_4 \end{cases} \\
Inv_1 &= \{q_1, \{u_D < h_D, u_N < h_N\}\} \cup \{q_2, \{u_D \geq h_D, u_N < h_N\}\} \\
&\cup \{q_3, \{u_D < h_D, u_N \geq h_N\}\} \cup \{q_4, \{u_D \geq h_D, u_N \geq h_N\}\} \\
R_1 &: \begin{bmatrix} R_1(q_1, \{u_D \geq h_D \wedge u_N < h_N\}) \in q_2 \times \mathfrak{R}^2 \\ R_1(q_1, \{u_D < h_D \wedge u_N \geq h_N\}) \in q_3 \times \mathfrak{R}^2 \\ \vdots \\ R_1(q_4, \{u_D < h_D \wedge u_N \geq h_N\}) \in q_3 \times \mathfrak{R}^2 \end{bmatrix}
\end{aligned}$$

where,  $x_1$  and  $x_2$ : Delta and Notch protein concentrations, respectively, in a cell;  $x_{1,i}$ : Delta protein concentration in  $i^{\text{th}}$  neighboring cell;  $\lambda_D$  and  $\lambda_N$ : Delta and Notch protein decay constants respectively;  $R_D$  and  $R_N$ : constant Delta and Notch protein production rates, respectively;  $h_D$  and  $h_N$ : switching thresholds for Delta and Notch protein production, respectively. experimentally-determined constants. The switching thresholds  $h_D$  and  $h_N$  are unknown and possible ranges for them are derived in [10].

In the single cell,  $x_{1,i} = 0, \forall i \in \{1, \dots, 6\}$ . The inputs  $u_D$  and  $u_N$  are the physical realization of the protein regulatory properties in the model outlined before. The two cell hybrid automaton  $H_2$  is the composition of two single cell automata, to form a model with four continuous states  $(x_1, \dots, x_4)$  and sixteen discrete modes. Here,  $u_N \neq 0$  for each of the two cells, and thus the Delta level of each cell is communicated to its neighbor to control Notch production. Modeling the full two dimensional layer of cells involves composing  $M \times N$  single cell hybrid automata, with the coupling through the input functions as described above.

### 2.3 Equilibrium Analysis

Both the single and two cell hybrid automata were analyzed in [10], to obtain constraints on the range of the protein kinetic parameters and switching thresholds for biologically feasible equilibria to exist:

$$h_D, h_N : -\frac{R_N}{\lambda_N} < h_N \leq 0 \wedge 0 < h_N \leq \frac{R_D}{\lambda_D}$$

The two cell automaton was also shown to have a Zeno state with a particular Zeno execution that is an invariant:  $x_1 = x_3 \wedge x_2 = x_4$ , where  $x_3$  and  $x_4$  are the Delta and Notch protein concentrations, respectively, in the second cell. In the discrete abstraction procedure, it is essential to include the parameter constraints and the Zeno execution as “polynomials of interest” (explained in the following section), to produce an abstraction with sufficient resolution. Hence, the analysis is an indispensable part of any formal verification done on the discrete abstraction in general.

### 3 SAL Model

The first step of the procedure is to generate the SAL language description of the hybrid model and the necessary user-defined predicates (for example, constraints derived from analysis). This task is non-trivial because the size of the model (and the number of lines of code) increase exponentially with the number of cells, since each one of the  $4^n$  distinct modes has to be encoded. Hence, we have automated the generation of the SAL specification file using MATLAB, for an arbitrarily large network. The equilibrium existence conditions and consequent parameter constraints described in the previous section are included in the SAL model.

#### 3.1 The SAL Modeling Language

Hybrid systems described in terms of interacting hybrid automata can be specified using the SAL modeling language [5, 9]. Here we shall only describe those features of the SAL modeling language that will be used in the specification of the Delta-Notch signaling mechanism.

A SAL specification consists of *module* definitions. A special kind of module, called *basemodule*, corresponds to a hybrid automaton. It consists of definitions of the *variables*, the *initial states*, and the discrete and continuous *transitions* of the hybrid automaton. The variables are classified as *input*, *output*, *local*, and *global*. Variables are typed, discrete variables are usually of type BOOLEAN and continuous variables are of type REAL. The initial states of the hybrid automaton can be specified either directly, or by specifying a formula. In the latter case, the set of initial states consists of all those valuations of variables that make the formula true.

The discrete and continuous transitions are specified using a set of *guarded commands*. A guarded command is written as:

$$guard \longrightarrow x'_1 = e_1; x'_2 = e_2; x'_3 = e_3; \dots$$

where *guard* is a formula over the variables,  $x_i$  are variables, and  $e_i$  are expressions over the variables. Semantically, the above command means that if the formula in the guard is true for some state (a state is a valuation of all the variables), then the transition can be taken. Note that the new value of a variable, say  $x_1$ , is denoted by priming it,  $x'_1$ . In case of a discrete transition,

the variables  $x_i$  are updated to the new values given by expressions  $e_i$ . In a continuous transition, the variables  $x_i$  on the left-hand sides of the assignment are special variables, whose names end with the suffix “dot”. They specify the time-derivative of the corresponding continuous variable. For example, the automatically generated SAL module specifying the single cell Delta-Notch hybrid automaton  $H_1$  is given below.

```

one_cell : CONTEXT =
BEGIN
system[u, hD, hN, RD, RN, lD, lN : REAL] : MODULE =
BEGIN
  GLOBAL x1, x2 : REAL
  GLOBAL x1dot, x2dot : REAL
  INVARIANT
    x1 > 0 AND lD*x1 < RD AND x2 > 0 AND lN*x2 < RN AND
    u < hN AND lN* hD > -RN AND hD < 0 AND hN > 0 AND
    lD*hN < RD AND RN > 0 AND RD > 0 AND lN > 0 AND lD > 0
  INITFORMULA
    u < hN AND x2 < -hD
  TRANSITION
  [
    -x2 < hD AND u < hN -->
      x1dot' = -(-lD*x1);
      x2dot' = -(-lN*x2)
    []
    -x2 < hD AND u >= hN -->
      x1dot' = -(-lD*x1);
      x2dot' = -(RN-lN*x2)
    []
    -x2 >= hD AND u < hN -->
      x1dot' = -(RD-lD*x1);
      x2dot' = -(-lN*x2)
    []
    -x2 >= hD AND u >= hN -->
      x1dot' = -(RD-lD*x1);
      x2dot' = -(RN-lN*x2)
  ]
END;
END

```

Note that the parameter constraints are explicitly included as invariant polynomials, and the initial state, defined using the `INITFORMULA` declaration, is  $q_2$ , which has an equilibrium that implies high Delta level and low Notch level and is biologically consistent. Here, the backward reachable set from the equilibrium in  $q_2$  is desired, hence the dynamics in each mode are reversed, by negating the right-hand-sides of the governing equations. In the SAL model,  $\lambda$  has been replaced by  $l$  for brevity,  $\lambda_N$  is written as  $lN$  for example. Also note that the input  $u = u_N$  for the system has been fixed to  $u_N < h_N$  for this example.

## 4 Discrete Abstraction

The second step of the reachable set computation procedure is the automatic generation of the discrete abstraction of the hybrid automaton. An abstract model is usually obtained from a given hybrid model by partitioning the state-space of the original model into a finite set and mapping the dynamics of the original model onto this finite set. In the case of a model described using  $n$  real-variables (representing, say, the concentrations of  $n$  different protein complexes),

the state-space is the  $n$ -dimensional real space  $\mathfrak{R}^n$ . We partition this space into a finite number of zones using a finite set  $P$  of polynomials over the  $n$  variables. Each zone corresponds to a subset of  $\mathfrak{R}^n$  that is sign-invariant for all polynomials in the set  $P$ . Increasing the number of polynomials in the set  $P$  results in more zones, and consequently, a finer abstraction. This basic idea, although in a much simplified form, is also at the core of qualitative reasoning techniques developed in the AI community (see, for example [21]).

The process of construction of the abstract system requires logical reasoning in the theory of reals. The first-order theory of real closed fields is known to be decidable [23] and the first practical algorithm, based on cylindrical algebraic decomposition, was given in [8], which has gone through several improvements [13, 17]. We use the first-order theory of reals to represent sets of continuous states and use reasoning over this theory for creating the abstract transition system.

#### 4.1 Abstraction Process

The abstraction technique is completely automatic and is described in [24]. The abstraction algorithm works in two phases: In the first phase, the algorithm computes a set of polynomials (over the continuous variables), which is used to partition the continuous state-space. This is achieved by starting with a finite set of polynomials which appear in the vector field of the continuous dynamics, in the conditions of discrete transitions, initialization expressions, and property to be proved. This set is saturated under the derivative operator, that is, the time derivative of a polynomial in this set is also added to this set. Note that computation of the derivative of a polynomial can be symbolically performed using the derivatives of individual variables. This saturation process may not terminate, but for our purposes can be stopped at any time. The termination condition is specified by giving the bound on the order of the derivatives that is computed.

In the second phase, the discrete and continuous transitions are abstracted. The abstract state-space is given as a cross-product of the discrete-state space and the regions of the original continuous state-space where the polynomials computed in the saturation phase are sign-invariant. This requires the use of decision procedures for the real-closed fields, intermingled with theorem-proving strategies.

For scalability of the abstraction technique, it is necessary that the formulas that arise in proof obligations while constructing an abstraction and when checking for feasibility of abstract states, do not grow in size. We have achieved this by partitioning the set of variables into classes such that variables that interact with each other (occur in the same polynomial) are put into the same class. Furthermore, when proving a proof obligation, atomic formulas that are not “relevant” to the succedent (that is, atomic formulas that do not contain any variables that occur in the polynomials in the succedent) are removed from the formula. This optimization simplifies the formulas to be proved. Furthermore, it also reduces the number of proof obligations that arise in the construction of the feasible abstract state space. A unique feature of the abstraction algorithm (and

its implementation) is that it allows to abstract parameterized systems, without instantiating the parameters. Following our one cell Delta-Notch example, the abstraction procedure maps six abstract variables,  $g_0, g_1, \dots, g_5$ , to polynomials in the original hybrid system:

```
g0 --> 1N*x2 - RN      g2 --> u - hN      g4 --> 1D*x1 - RD
g1 --> 1N*x2          g3 --> x1          g5 --> x2 + hD
```

The polynomials are either related to the continuous evolution of the model, for example  $g_0, g_1, g_3, g_4$ , or are the switching planes defining mode transitions, for example  $g_2$  and  $g_5$ . The transitions in the abstract model arise from the discrete mode change transitions of the original model, or from the continuous dynamics. For example, the continuous evolution in mode  $q_1$  is abstracted to the following SAL transition:

```
g2 = neg AND g5 = pos
-->
g5' IN IF g5 = pos
  THEN IF g1 = pos OR g1 = zero THEN {pos} ELSE {pos, zero} ENDIF
  ELSIF g5 = neg
    THEN IF g1 = neg OR g1 = zero
      THEN {neg}
      ELSE {neg, zero}
    ENDIF
  ELSE IF g1 = pos
    THEN {pos}
    ELSIF g1 = neg THEN {neg} ELSE {zero}
  ENDIF
ENDIF;
g4' = g4; g3' = g3; g2' = g2;
g1' IN IF g1 = pos
  THEN IF FALSE THEN {pos} ELSE {pos, zero} ENDIF
  ELSIF g1 = neg
    THEN IF FALSE THEN {neg} ELSE {neg, zero} ENDIF
    ELSE IF FALSE
      THEN {zero}
      ELSIF FALSE
        THEN {pos}
        ELSIF FALSE THEN {neg} ELSE {pos, zero, neg}
    ENDIF
ENDIF;
g0' = g0
```

The invariant  $-x_2 < h_D \wedge u < h_N$  has been abstracted to ( $g_2 = \text{neg}$  AND  $g_5 = \text{pos}$ ). The binary “IN” operator denotes that the (new) value of the left-hand side variable is non-deterministically chosen from the set of values specified on the right-hand side (in contrast, the “=” operator sets the value of the left-hand side variable to a deterministic value). The new value of the variable is determined by the sign of its derivative. For example, the new value of  $g_5$  depends on  $g_1$ , which is the derivative of  $g_5$  in mode  $q_1$ .

## 5 Reachability Computation

The next step is the automatic computation of the backward reachable set using the SAL model-checker coupled with the reachability tool. The result is in the form of a vector that gives the reachable set of states. This is the input to another

MATLAB script which reads the vector, and uses the polynomial invariants defining each set in that vector to construct the reachable set as a union of those invariants. This is then passed to QEPCAD which returns a compact, human-understandable form of the reachable set.

The reachability computation takes advantage of the fact that the SAL model-checker is an explicit-state model checker, i. e. it traces all possible executions of the FSA explicitly by firing all valid transitions from all reachable states, and checks each state against a linear temporal logic (LTL) formula. By giving the model checker a trivial formula to check, for example  $G(\text{TRUE})$ , we can ensure that the model-checker will search the entire reachable state space without coming up with a counter-example. The reachability tool is a LISP program which runs concurrently with the model-checker and stores each valid state that the model-checker visits during its state space exploration. The model checker is initialized from the hybrid automaton mode (which may be equivalent to several discrete states of the abstract FSA) containing the equilibrium whose backward reachable set we want to compute. Initializing the model-checker with the equilibrium-containing mode is valid for our example, because simple eigenvalue analysis of the equations show that the dynamics in that mode are exponentially stable, therefore the entire mode is backward reachable from the equilibrium point. After the model-checker terminates, it returns a vector of backward reachable sets. In our example, for the single cell automaton  $H_1$ , after simplification the backward reachable set is given by:  $x_1 > 0 \wedge x_2 > 0 \wedge \lambda_d x_1 < R_D \wedge \lambda_N x_2 < R_N \wedge u_N < h_N$ . This means that the entire state space of interest is reachable from the equilibrium provided the input condition is satisfied. For larger systems, the simplification of the reachable set vector is non-trivial because we have to compute conjunctions of a large number of inequalities defining the invariants of states. This is done using QEPCAD, which can return a reasonably compact equivalent formula.

## 6 Results

### 6.1 Two Cell System

The two cell hybrid automaton  $H_2$  is constructed by composing two single cell hybrid automata. It has four continuous state variables  $x_1, \dots, x_4$  and sixteen discrete modes  $q_1, \dots, q_{16}$ .  $x_1$  and  $x_2$  represent the Delta and Notch protein concentration in the first cell, and  $x_3$  and  $x_4$  represent the Delta and Notch protein concentration in the second cell. From previous analysis [10], it was determined that only two biologically feasible equilibria exist for this system: (i)  $x_1^* = 0, x_2^* = \frac{R_N}{\lambda_N}, x_3^* = \frac{R_D}{\lambda_D}, x_4^* = 0$ , which means the Delta protein level in the second cell is high and that in the first cell is low, and the Notch protein level is high in the first cell and low in the second, and (ii)  $x_1^* = \frac{R_D}{\lambda_D}, x_2^* = 0, x_3^* = 0, x_4^* = \frac{R_N}{\lambda_N}$ , which is the symmetric result. We are interested in finding the set of initial conditions which converge to either one or the other equilibria. Therefore the backward reachable set computation is performed for both the equilibria. The time required for the model construction

and abstraction is around 25 minutes and another 40 minutes for the reachability computation using the model checker, on a Pentium III 500MHz computer running Linux. The computed sets are given below:

Eq 1: (0,RN/1N,RD/1D,0)

```
x3 - x1 >= 0 /\ x4 - x2 <= 0 /\ [
[ x3 - x1 > 0 /\ hN - x3 > 0 ] \/
[ x3 - x1 > 0 /\ hN - x1 < 0 ] \/
[ x4 - x2 < 0 /\ hN - x3 > 0 ] \/
[ x4 - x2 < 0 /\ hN - x1 < 0 ] \/
[ x3 - x1 > 0 /\ hD + x4 > 0 ] \/
[ x3 - x1 > 0 /\ hD + x2 < 0 ] \/
[ x4 - x2 < 0 /\ hD + x4 > 0 ] \/
[ x4 - x2 < 0 /\ hD + x2 < 0 ] \/
[ hN - x3 <= 0 /\ hN - x1 > 0 /\ hD + x2 > 0 ]]
```

Eq 2: (RD/1D,0,0,RN/1N)

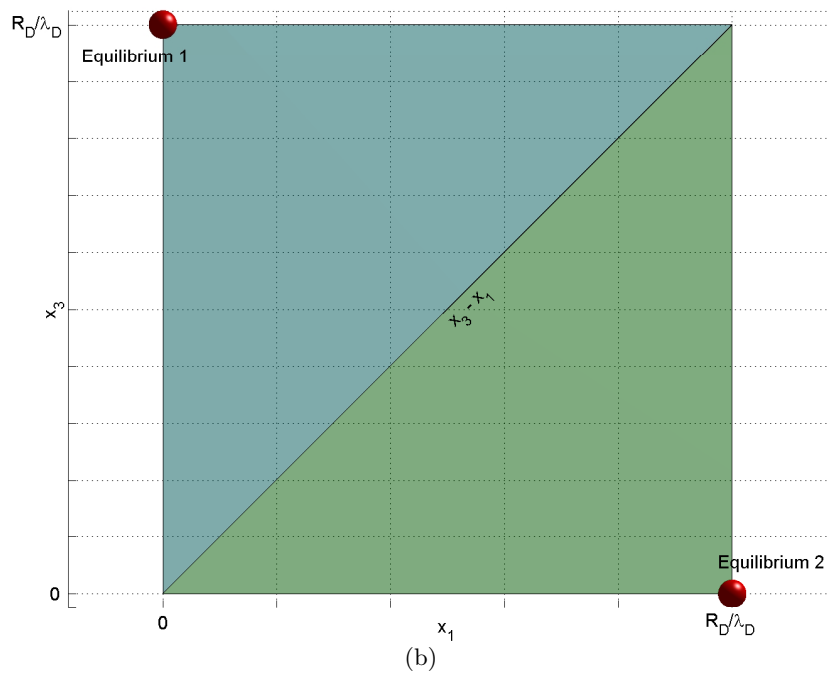
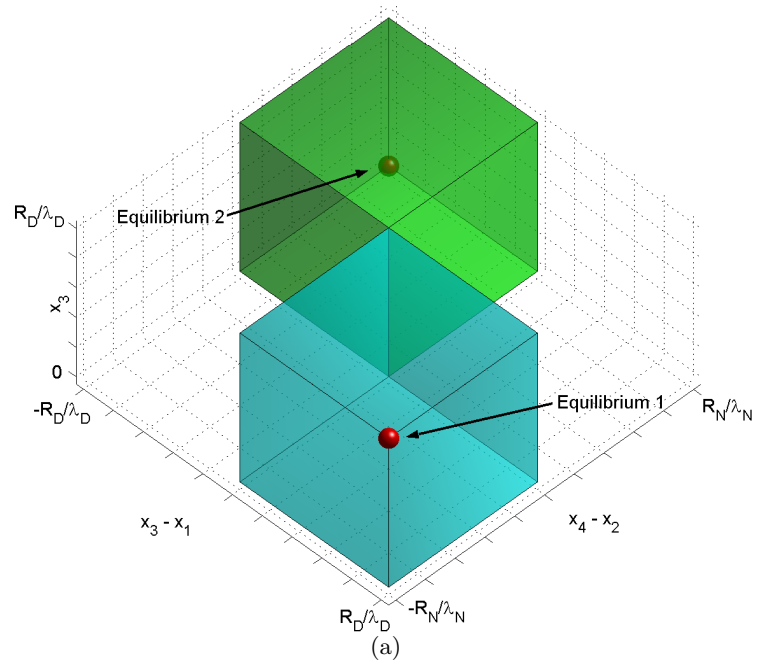
```
x3 - x1 <= 0 /\ x4 - x2 >= 0 /\ [
[ x3 - x1 < 0 /\ hN - x1 > 0 ] \/
[ x3 - x1 < 0 /\ hN - x3 < 0 ] \/
[ x4 - x2 > 0 /\ hN - x1 > 0 ] \/
[ x4 - x2 > 0 /\ hN - x3 < 0 ] \/
[ x3 - x1 < 0 /\ hD + x2 > 0 ] \/
[ x3 - x1 < 0 /\ hD + x4 < 0 ] \/
[ x4 - x2 > 0 /\ hD + x2 > 0 ] \/
[ x4 - x2 > 0 /\ hD + x4 < 0 ] \/
[ hN - x3 > 0 /\ hN - x1 <= 0 /\
hD + x4 > 0 ] ]
```

The reachable set for equilibrium 1 implies that all initial conditions that satisfy  $x_3 > x_1 \wedge x_4 < x_2$  converge to that equilibrium and the reachable set for equilibrium 2 implies that all initial conditions that satisfy  $x_3 < x_1 \wedge x_4 > x_2$  converge to that equilibrium. The four dimensional reachable sets can be better visualized by looking at their projections in lower dimensional space, examples of which are shown in Fig. 3.

In Fig. 3(a), both reachable sets are projected onto a three dimensional space with  $x_3 - x_1$ ,  $x_4 - x_2$  and  $x_3$  as the axes. In this projection, the reachable sets are disjoint cubes with the respective equilibria at the corners. Note that the edge  $x_3 = x_1 \wedge x_4 = x_2$  separates the two reach sets. Neither of the two equilibria are reachable from this edge: this separatrix defines the Zeno trajectory previously identified [10]. The two dimensional projection onto the  $x_1, x_3$  plane shown in Fig. 3(b) makes the role of the separatrix clearer. It divides the plane into two half-planes, each of which is reachable from exactly one equilibrium point. These results are significant because, first, they involve symbolic computations over relatively high dimensional hybrid automata, and, second, they predict conditions that are biologically significant but not obvious from either the model or the biological system.

## 6.2 Four Cell System

Currently, the largest network we have performed the reach set computation for, is a four cell network. The hybrid automaton for a four cell network is constructed by composing four single cell automata as shown in Fig. 2(b). The sum of the Delta protein levels of neighboring cells are used to control Notch protein production in a cell. For example, cell 1 reads the Delta protein levels from cells 2 and 3,  $x_3$  and  $x_5$  respectively, and uses the sum  $x_3 + x_5$  to regulate its Notch protein concentration,  $x_2$ . The set of parameter constraints from the two cell hybrid automata is used for abstraction. The four cell automaton has eight continuous state variables (two proteins for each of the four cells), and 256 discrete states. There are three biologically feasible equilibria for this network: (i) cells 1 and 4 have high Delta levels and low Notch levels, and cells 2 and 3 have high



**Fig. 3.** Projections of reach set for two cell hybrid automaton

Notch levels and low Delta levels, (ii) cell 2 has high Delta levels and low Notch levels at steady state, and all the other cells have low Delta levels and high Notch levels, and (iii) cell 3 has high Delta concentration and low Notch concentration at steady state and the other three cells have low Delta levels and high Notch levels. It is important to identify the backward reachable sets for each of these three equilibria and we have investigated the reach set computation for the first equilibrium mentioned above. The model generation and reach set computation time required for that is 8 hours, on a Pentium III 500MHz computer running Linux. The resultant reach set generated by the SAL model checker is rather large, it comprises the union of 1576 states of the abstracted discrete automaton defined in terms of the abstraction polynomials. Computation of a simplified formula equivalent to the union of those 1576 states is non-trivial. QEPCAD has memory and dimensionality limitations which precludes the simplification computation of all 1576 states at once. To circumvent these limitations, we analyzed the polynomials defining the abstracted states and reduced the dimensionality of the problem by appropriate projections, as explained next. The state space of the hybrid automaton is abstracted by mapping it to the following set of polynomials:

```

g0 --> -x2-hD   g3 --> -x8-hD       g6 --> x5+x3-hN       g9 --> x5+x3-RD/lD
g1 --> -x4-hD   g4 --> x7+x3+x1-hN    g7 --> x7+x5+x1-RD/lD g10 --> x7+x5+x1-2*RD/lD
g2 --> -x6-hD   g5 --> x7+x5+x1-hN    g8 --> x7+x3+x1-RD/lD g11 --> x7+x3+x1-2*RD/lD
g12 --> x7+x3+x1-3*RD/lD

```

Each state of the abstracted FSA is defined by assigning a sign to the above polynomials and computing their conjunction. For example,  $(g_0 = 0 \wedge g_1 = 0 \wedge g_2 = 0 \wedge g_3 > 0 \wedge g_4 = 0 \wedge g_5 = 0 \wedge g_6 > 0 \wedge g_7 < 0 \wedge g_8 < 0 \wedge g_9 > 0 \wedge g_{10} < 0 \wedge g_{11} < 0 \wedge g_{12} < 0)$  is a mode of the abstracted FSA. The polynomials involve twelve symbolic variables, eight variables  $x_1, \dots, x_8$  denoting the protein concentrations in the cells and four parameters associated with the dynamics. Note that the variables  $x_2, x_4, x_6, x_8$  only appear in conjunction with the parameter  $h_D$  and nowhere else. Hence, we can map those five into the following four variables:  $p_1 = -x_2-h_D$ ,  $p_2 = -x_4-h_D$ ,  $p_3 = -x_6-h_D$ ,  $p_4 = -x_8-h_D$ . This reduces the dimensionality by one. Also, note that  $x_7$  and  $x_1$  always occur together as the sum  $x_7+x_1$ , hence we can map it into a single variable  $p_5 = x_7 + x_1$  enabling us to reduce the dimensionality by one more. Similarly, the ratio  $R_D/l_D$  can be mapped to a single variable  $p_6 = RD/lD$ , which reduces the dimensionality by another order. Therefore, instead of twelve symbolic variables to simplify over, we now have only nine. This is a substantial savings as the complexity of the problem grows exponentially with the dimensionality.

The simplified reach set thus computed turns out to be the union of 1192 states, which is still too large to interpret as a whole. Instead of trying to find meaning in a large non-convex set, we then partition the reach set using the definitions of the hybrid states from the four cell hybrid automata, and compute the intersections of the invariants defining each of the 256 hybrid states, with the simplified reach set. This enables us to determine the hybrid states that are completely or partially backward reachable from the equilibrium. The natural structure of the state space is thus used to divide the reach set into smaller

parts that are easier to interpret. It is observed that out of the 256 hybrid states, only 90 states are backward reachable from the equilibrium, completely or partially. The hybrid states are defined in terms of the signs of the switching functions governing Delta and Notch production, which are:  $p_1$ ,  $p_2$ ,  $p_3$ ,  $p_4$ ,  $x_3+x_5-h_N$ ,  $p_5+x_3-h_N$  and  $p_5+x_5-h_N$ . For example, the mode in which neither Delta nor Notch is produced in any of the four cells is defined as:

$$p_1 < 0 \wedge p_2 < 0 \wedge p_3 < 0 \wedge p_4 < 0 \wedge x_3+x_5-h_N < 0 \wedge p_5+x_3-h_N < 0 \wedge p_5+x_5-h_N < 0$$

To simplify the representation of the states, we define the following expressions:

```

e1 = p1 < 0 ∧ p2 < 0 ∧ p3 < 0 ∧ p4 < 0
e2 = p1 < 0 ∧ p2 < 0 ∧ p3 < 0 ∧ p4 >= 0
e3 = p1 < 0 ∧ p2 < 0 ∧ p3 >= 0 ∧ p4 < 0
e4 = p1 < 0 ∧ p2 < 0 ∧ p3 >= 0 ∧ p4 >= 0
e5 = p1 < 0 ∧ p2 >= 0 ∧ p3 < 0 ∧ p4 < 0
e6 = p1 < 0 ∧ p2 >= 0 ∧ p3 < 0 ∧ p4 >= 0
e7 = p1 < 0 ∧ p2 >= 0 ∧ p3 >= 0 ∧ p4 < 0
e8 = p1 < 0 ∧ p2 >= 0 ∧ p3 >= 0 ∧ p4 >= 0
e9 = p1 >= 0 ∧ p2 < 0 ∧ p3 < 0 ∧ p4 < 0
e10 = p1 >= 0 ∧ p2 < 0 ∧ p3 < 0 ∧ p4 >= 0
e11 = p1 >= 0 ∧ p2 < 0 ∧ p3 >= 0 ∧ p4 < 0
e12 = p1 >= 0 ∧ p2 < 0 ∧ p3 >= 0 ∧ p4 >= 0
e13 = p1 >= 0 ∧ p2 >= 0 ∧ p3 < 0 ∧ p4 < 0
e14 = p1 >= 0 ∧ p2 >= 0 ∧ p3 < 0 ∧ p4 >= 0
e15 = p1 >= 0 ∧ p2 >= 0 ∧ p3 >= 0 ∧ p4 < 0
e16 = p1 >= 0 ∧ p2 >= 0 ∧ p3 >= 0 ∧ p4 >= 0
f1 = x3+x5-hN < 0 ∧ p5+x3-hN < 0 ∧ p5+x5-hN < 0
f2 = x3+x5-hN < 0 ∧ p5+x3-hN < 0 ∧ p5+x5-hN >= 0
f3 = x3+x5-hN < 0 ∧ p5+x3-hN >= 0 ∧ p5+x5-hN < 0
f4 = x3+x5-hN < 0 ∧ p5+x3-hN >= 0 ∧ p5+x5-hN >= 0
f5 = x3+x5-hN >= 0 ∧ p5+x3-hN < 0 ∧ p5+x5-hN < 0
f6 = x3+x5-hN >= 0 ∧ p5+x3-hN < 0 ∧ p5+x5-hN >= 0
f7 = x3+x5-hN >= 0 ∧ p5+x3-hN >= 0 ∧ p5+x5-hN < 0
f8 = x3+x5-hN >= 0 ∧ p5+x3-hN >= 0 ∧ p5+x5-hN >= 0

```

Using the above expressions, the reachable hybrid states can be listed:

Completely reachable	Partially reachable
e1 ∧ [f1 ∨ f5]	e1 ∧ [f2 ∨ f6 ∨ f7 ∨ f8]
e2 ∧ [f1 ∨ f5]	e2 ∧ [f6 ∨ f7 ∨ f8]
e3 ∧ f1	e3 ∧ [f5 ∨ f6 ∨ f8]
e4 ∧ f1	e4 ∧ [f5 ∨ f6 ∨ f7]
e5 ∧ f1	e5 ∧ [f2 ∨ f6 ∨ f8]
e6 ∧ f1	e6 ∧ [f2 ∨ f5 ∨ f6 ∨ f7]
e7 ∧ f1	e7 ∧ [f2 ∨ f6 ∨ f8]
e8 ∧ f1	e8 ∧ [f2 ∨ f5 ∨ f6 ∨ f7 ∨ f8]
e9 ∧ f1	e9 ∧ [f3 ∨ f5 ∨ f6 ∨ f7]
e10 ∧ [f1 ∨ f4]	e10 ∧ [f3 ∨ f5 ∨ f6 ∨ f7 ∨ f8]
e11 ∧ f1	e11 ∧ [f3 ∨ f5 ∨ f6 ∨ f7]
e12 ∧ f1	e12 ∧ [f3 ∨ f5 ∨ f6 ∨ f7 ∨ f8]
e13 ∧ f1	e13 ∧ [f2 ∨ f3 ∨ f4 ∨ f6]
e14 ∧ f1	e14 ∧ [f2 ∨ f3 ∨ f4 ∨ f5 ∨ f6 ∨ f7 ∨ f8]
	e15 ∧ [f1 ∨ f2 ∨ f3 ∨ f4 ∨ f5 ∨ f6 ∨ f7 ∨ f8]
	e16 ∧ [f1 ∨ f2 ∨ f3 ∨ f4 ∨ f5 ∨ f6 ∨ f7 ∨ f8]

We are currently in the process of analyzing the reach set results for biologically significant information, but a few interesting properties of this equilibrium can be mentioned:

- Among the hybrid states that are completely reachable, fourteen states satisfy the constraint  $f_1$ , i.e.  $x_7 + x_3 + x_1 - h_N < 0 \wedge x_7 + x_5 + x_1 - h_N < 0$

- $0 \wedge x_5 + x_3 - h_N < 0$ . This implies that for all Notch protein levels except  $-h_D \geq x_2 \wedge -h_D \geq x_4 \wedge -h_D \geq x_6$ , if the Delta levels satisfy the constraint **f1**, then the equilibrium will be attained.
- It is observed that all hybrid states with  $-h_D < x_2 \wedge x_5 + x_3 < h_N \wedge x_7 + x_3 + x_1 \geq h_N$  are completely unreachable. This implies that regardless of the Notch protein levels of cells two, three and four (Fig. 2(b)), if the Notch protein concentration of cell one and the Delta protein levels of all four cells satisfy the above constraints, then the equilibrium will be unreachable.

## 7 Conclusion

We have described, in this paper, the automatic computation of backward reachable sets for high dimensional piecewise affine hybrid automata. The reachability procedure combines previously developed code in SRI's SAL with automation steps written by us. We have achieved results for a hybrid automaton with 8 continuous dimensions and 256 discrete modes. The concrete example that we have used, that of a Delta-Notch signaling network, is one of many physical systems that can be modeled using this class of hybrid automata. The ability to do symbolic computations make our hybrid automaton-based modeling and analysis framework ideal for a wide range of cell biological regulatory and signaling processes. One of our current research projects focuses on applying the reachability tools that we have developed to analyze planar cell polarity signaling in *Drosophila* (fruit fly) pupal wing epithelium [2].

## References

1. R. Alur, T. Dang, and F. Ivancic. Reachability analysis of hybrid systems via predicate abstraction. In C. J. Tomlin and M. Greenstreet, editors, *Hybrid Systems: Computation and Control*, LNCS 2289, pages 35–48. Springer Verlag, 2002.
2. K. Amonlirdviman, R. Ghosh, J. Axelrod and C. Tomlin. A hybrid systems approach to modeling and analyzing planar cell polarity. In *International Conference on Systems Biology*, Stockholm, 2002.
3. E. Asarin, T. Dang, and O. Maler. d/dt: A verification tool for hybrid systems. In *Proc. of the IEEE Conf. on Decision and Control*, pages 2893–2898, Orlando, 2001.
4. A. Bemporad, F. D. Torrisi, and M. Morari. Optimization-based verification and stability characterization of piecewise affine and hybrid systems. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control*, LNCS 1790, pages 45–59. Springer Verlag, 2000.
5. 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, *LFM 2000: Fifth NASA Langley Formal Methods Workshop*, pages 187–196, Hampton, VA, June 2000. NASA Langley Research Center.
6. O. Botchkarev and S. Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In B. Krogh and N. Lynch, editors,

- Hybrid Systems: Computation and Control*, LNCS 1790, pages 73–88. Springer Verlag, 2000.
7. A. Chutinan and B. H. Krogh. Verification of infinite-state dynamic systems using approximate quotient transition systems. *IEEE Trans. on Automatic Control*, 46(9):1401–1410, 2001.
  8. G. E. Collins. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In *Proc. Second GI Conf. Automata Theory and Formal Languages*, LNCS 33, pages 134–183. Springer Verlag, 1975.
  9. Computer Science Laboratory, SRI International, Menlo Park, California. SAL: Symbolic Analysis Laboratory. <http://www.csl.sri.com/projects/sal/>.
  10. R. Ghosh and C. J. Tomlin. Lateral inhibition through delta-notch signaling: a piecewise affine hybrid model. In M. D. D. Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, LNCS 2034, pages 232–246. Springer Verlag, 2001.
  11. S. Graf and H. Saïdi. Construction of abstract state graphs with PVS. In O. Grumberg, editor, *Proc. 9th International Conference on Computer Aided Verification (CAV'97)*, volume 1254, pages 72–83. Springer Verlag, 1997.
  12. T. A. Henzinger, P. H. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1:110–122, 1997.
  13. H. Hong. An improvement of the projection operator in cylindrical algebraic decomposition. In *Proc. ISAAC 90*, pages 261–264, 1990.
  14. H. de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *J. Computational Biology*, 9(1):69–105, 2002.
  15. B. Kuipers and S. Ramamoorthy. Qualitative modeling and heterogeneous control of global systems behavior. In C. J. Tomlin and M. Greenstreet, editors, *Hybrid Systems: Computation and Control*, LNCS 2289, pages 294–307. Springer Verlag, 2002.
  16. G. Marnellos, G. A. Deblandre, E. Mjolsness, and C. Kintner. Delta-notch lateral inhibitory patterning in the emergence of ciliated cells in *Xenopus*: experimental observations and a gene network model. In *Pacific Symposium on Biocomputing*, pages 326–337, 2000.
  17. S. McCallum. An improved projection operator for cylindrical algebraic decomposition of three dimensional space. *J. Symbolic Computation*, 5:141–161, 1988.
  18. I. Mitchell. *Application of level set methods to control and reachability problems in continuous and hybrid systems*. PhD thesis, Stanford University, August 2002.
  19. I. Mitchell and C. J. Tomlin. Overapproximating reachable sets by Hamilton-Jacobi projections. *J. Symbolic Computation*, 2002.
  20. J. Preußig and H. Wong-Toi. A procedure for reachability analysis of rectangular automata. In *Proc. of the American Control Conference*, pages 1674–1678, Chicago, 2000.
  21. B. Shults and B. J. Kuipers. Proving properties of continuous systems: qualitative simulation and temporal logic. *AI Journal*, 92:91–129, 1997.
  22. O. Sokolsky and H. S. Hong. Qualitative modeling of hybrid systems. In *Proc. of the Montreal Workshop*, 2001. Available from <http://www.cis.upenn.edu/~rtg/rtg-papers.htm>.
  23. A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, second edition, 1948.
  24. A. Tiwari and G. Khanna. Series of abstractions for hybrid automata. In C. J. Tomlin and M. Greenstreet, editors, *Hybrid Systems: Computation and Control*, LNCS 2289, pages 465–478. Springer Verlag, 2002.