43rd IEEE Conference on Decision and Control
December 14-17, 2004
Atlantis, Paradise Island, Bahamas

An Algorithm for Reachability Computations on Hybrid Automata Models of Protein Signaling Networks

Ronojoy Ghosh and Claire Tomlin

Abstract— **Hybrid automata theory is an ideal mathematical framework for modeling biological protein signaling mechanisms. Reachability analysis of these models is essential, because the set of points backward reachable from a biologically feasible equilibrium contains all initial protein concentrations from which that steady state can be attained. This is useful for determining experimentally verifiable properties of the system under study. This paper proposes an algorithm for computing discrete abstractions of a class of hybrid automata with piecewise affine continuous dynamics, defined completely in terms of symbolic variables and parameters. These discrete abstractions are utilized to compute symbolic parametric backward reachable sets from the equilibria of the hybrid automata. The algorithm has been implemented and used to compute reachable sets for the biologically observed equilibria of multiple cell Delta-Notch protein signaling networks.**

I. Introduction

Protein signaling is an essential biological process that controls phenomena throughout the life cycle of a cell, from cell differentiation and growth, to apoptosis, or programmed cell death. Signaling involves interactions between proteins in both extracellular and cytoplasmic domains. These interactions affect the activation and concentration of the proteins involved. When one protein regulates another, it may *promote*, i.e. increase the activation or concentration of the target, or *repress*, i.e. decrease the activation or concentration of the regulated protein. Mathematically, this can be modeled using a piecewise affine hybrid automaton with different linear ordinary differential equations governing the continuous dynamics of the proteins in different modes of operation, i.e. when protein activity or production is switched on or off. The hybrid automaton can be analyzed to determine the equilibrium points of the system, and regions of state space can be computed that are backward reachable from the equilibria. This allows prediction of initial protein concentrations that lead to a biologically feasible steady state, which may be experimentally verifiable.

This paper presents a novel algorithm that iteratively partitions the state space of a piecewise affine hybrid automaton with symbolic parameters and rate constants, to produce an abstracted discrete transition system. The proposed abstraction algorithm uses a systematic way of computing transitions and exact symbolic solutions of the continuous differential equations to iteratively refine the partitions. An under-approximate backward reachable set from the equilibria of the automaton is then computed on the discrete abstraction. The most important characteristic of the analysis is that it is completely symbolic, i.e. none of the parameters such as protein production, activation, and decay coefficients or switching thresholds are numerically instantiated. Reachable sets and constraints that involve ratios of symbolic kinetic parameters are generated; for example, constraints on the relative rates of production of two different proteins so that a biologically feasible equilibrium is reached. The first section contains a summary of the properties of the Delta-Notch protein signaling network, followed by a section that gives the formal definitions of mathematical concepts used in the paper. The next section describes and explains the abstraction algorithm, and its implementation is discussed in the following section. Lastly, the paper concludes with a summary of the reachability computation results and an outline of current and future research. A more detailed presentation of the algorithm and results is given in [1].

Some portions of this paper have been published in an extended form in *Systems Biology*, volume 1, number 1, June 2004. This work was sponsored by the DARPA Biocomp program (grant number DAAD19-03-1-0373 from the Department of the Army)

R. Ghosh and C. Tomlin are with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA ronojoy@stanford.edu,tomlin@stanford.edu

II. Delta-Notch Protein Signaling Network

The motivating example for this research is the Delta-Notch protein signaling mechanism, which has been identified as a key player in several different development processes, including pattern formation due to lateral inhibition [2], and is conserved across a broad spectrum of organisms. The authors have previously developed and analyzed hybrid automata models of the lateral inhibitory function of Delta-Notch protein networks, which have been described in detail in [3]. 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.

The important properties of the hybrid automaton model of Delta-Notch protein signaling are: (a) The continuous dynamics in each discrete mode is given by a diagonal state transition matrix (corresponding to the protein constitutive decay), and a constant input vector (corresponding to the constant protein productions). The modal invariants are simple polynomial functions of the continuous state variables, and the automaton is deterministic. (b) Discrete state transitions are only triggered by the continuous flow crossing a switching hyperplane, i.e. there are no discrete transitions accompanied by a continuous state reset, and no discrete jumps out from the interior of a mode. Hence

0-7803-8682-5/04/$20.00 ©2004 IEEE

the trajectories of the system are continuous, though not necessarily smooth. (c) The number of discrete states that contain equilibria are finite and enumerable, and the equilibria are in the interior of each state. Moreover, previous analysis done by the author [3], have shown that additional constraints on the system parameters (rate constants and switching thresholds) can restrict the existence of equilibria to biologically feasible modes. (d) The system is *live*, i.e. forced transitions exist for all states that do not contain equilibria.

III. Definitions

Definition 1: A piecewise affine hybrid automaton, $H = (Q, X, \Sigma, Init, f, Inv, R)$, is defined such that

1) $Q = \{q_1, q_2, \ldots, q_m\}$ is the set of discrete states;
2) $X \subset \Re^n$ is the set of continuous state variables;
3) $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ is the set of discrete inputs;
4) $Init = Q_0 \times X_0$ is the set of initial conditions;
5) $f(q, x) = A_q x + b_q$ is the continuous vector field associated with each discrete state, where $A_q \in \Re^{n \times n}$ is a diagonal matrix, and $b_q \in \Re^n$;
6) $Inv(q) = (\bigwedge_i (p_i < 0)) \wedge (\bigwedge_j (p_j = 0)) \wedge (\bigwedge_k (p_k > 0)) \wedge (\bigwedge_l (p_l \leq 0)) \wedge (\bigwedge_m (p_m \geq 0))$, where $p_i \in P_{lt}(q), p_j \in P_{eq}(q), p_k \in P_{gt}(q), p_l \in P_{le}(q), p_m \in P_{ge}(q)$, is the invariant defining each discrete state, where $p_{()} : X \times \Sigma \rightarrow \Re$ is a polynomial;
7) $R : Q \times X \times \Sigma \rightarrow 2^{Q \times X}$ is the transition map.

For this class of hybrid automata, the state transition matrix A_q is restricted to be diagonal with real eigenvalues. However, the elements of A_q and b_q are free to be symbolic. Constraints may be imposed on these symbolic constants to restrict the behavior of the model. The polynomials defining the invariant of each hybrid state, can be separated into five classes: $P_{lt}(q)$, $P_{eq}(q)$, $P_{gt}(q)$, $P_{le}(q)$, and $P_{ge}(q)$, according to their signs in the state. For example, all polynomials $p_i \in P_{lt}(q)$ are negative in state q, and similar definitions hold for the other classes $P_{eq}(q)$, etc. $P_{lt}(q), P_{eq}(q), \ldots, P_{ge}(q)$ are mutually disjoint, and $\forall q, P_{lt}(q) \cup P_{eq}(q) \cup P_{gt}(q) \cup P_{le}(q) \cup P_{ge}(q)$ is invariant.

This implies that the polynomials defining each state are identical, their sign alone varies from state to state. These classes are used to determine adjacency, i.e. whether two states are geometrical neighbors in state space, in several different steps of the abstraction algorithm presented in Section IV. In the transition map, transitions caused by the continuous flow of the automata crossing switching boundaries defined by the state invariant are called *forced* transitions.

Definition 2: A discrete transition system, $T = (Q, \Sigma, \rightarrow, Q_0, Q_F)$, is defined such that

1) $Q = \{q_1, q_2, \ldots, q_n\}$ is a set of states;
2) $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_N\}$ is a set of events;
3) $\rightarrow \subseteq Q \times \Sigma \times Q$ is a transition relation;
4) $Q_0 \subseteq Q$ is the set of initial states;
5) $Q_F \subseteq Q$ is the set of final states.

The transition system is *finite* if the cardinality of Q is finite, and it is *deadlock free* if for every state $q \in Q$, there exists a state $q' \in Q$ and an event $sigma \in \Sigma$ such that $q \xrightarrow{\sigma} q'$. Additionally, the transition system is *live* if for each state $q \in Q$, transition $q \xrightarrow{\sigma} q'$ is eventually taken. A dual representation of a finite transition system is an adjacency matrix $A \in \{0, 1\}^{n \times n}$, where $i \in 1, 2, \ldots, n$ represents a discrete state. In the adjacency matrix, $a_{i,j} \in A : a = 1$ means that a transition $q_i \rightarrow q_j$ exists, and $a_{i,j} = 0$ means no transition exists from q_i to q_j. The final states, $q \in Q_F$, of the transition system are states that have no transitions out of them.

The hybrid automaton H can be abstracted to the discrete transition system T by removing the temporal evolution of continuous state variables within each discrete state, but preserving the transitions from one discrete state to another. The set of events Σ of the transition system T then correspond to transitions relations encoded in the transition map R of the hybrid automaton H.

Definition 3: A state $\hat{q} \in Q$, of a transition system (that may be a hybrid automaton H or a discrete transition system T), is said to be reachable from another state q if there exists a finite sequence of transitions $q \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_N} \hat{q}$, from initial state q to state $\hat{q}$. The state q is said to be backward reachable from the state $\hat{q}$.

Extending this definition to sets of states, a region $P \subseteq Q$ of the hybrid automaton H or discrete transition system T, is *backward reachable* from a state $q \in Q$, if there exist sequences of transitions leading from P to q. When the hybrid automaton H is abstracted to the discrete transition system T, its equilibria are abstracted to the states in the set of final states Q_F. For a final state $q \in Q_F$, the region P is *uniquely* backward reachable from state q if all sequences of transitions from P terminate only in the state q. The regions of attraction of the steady states of the hybrid automaton therefore correspond to the uniquely backward reachable sets for the final states of the abstracted discrete transition system. Hence, the problem of computing the regions of attraction of the hybrid automaton is transformed to the problem of computing uniquely backward reachable sets for the abstracted transition system.

The accuracy of the computed backward reachable set depends critically on the expressiveness of the abstraction. If the abstraction algorithm results in a completely deterministic discrete transition system then the computed uniquely backward reachable set is exactly equal to the region of attraction for a particular steady state. Even when such a fine partition cannot be computed, it is possible, using the proposed algorithm in this paper, to construct a *best possible* abstraction that is partially deterministic and partially nondeterministic, from which an under-approximation of the backward reachable sets can be determined.

IV. Algorithm

This section describes, in detail, the partitioning algorithm developed to iteratively refine an initial coarse parti-

tion of the state space of a hybrid automaton. This algorithm is applicable to the restricted class of hybrid automata defined in Section III. The novelty of this algorithm lies in that it uses the concept of Lie derivatives to systematically compute transitions between the refined partitions and then iteratively computes subdividing partitions that are solutions of the governing differential equations in a discrete state of the hybrid automaton.

Algorithm 1: Partitioning Algorithm

Input: A hybrid automaton:
$H = (Q, X, \Sigma, Init, f, Inv, R)$, with restrictions
Output: A discrete transition system:
$T = (Q_T, \Sigma_T, \rightarrow, Q_{T,0}, Q_{T,F})$
Step 1
foreach $q \in Q$, such that $P_{le}(q) \neq \emptyset$ **or** $P_{ge}(q) \neq \emptyset$
define q_1, q_2, such that $f(q_1, x) = f(q_2, x) = f(q, x)$, and
$Inv(q_1) : P_{lt}(q_1) = P_{lt}(q) \cup P_{le}(q), P_{eq}(q_1) = P_{eq}(q), P_{gt}(q_1) = P_{gt}(q) \cup P_{ge}(q), P_{le}(q_1) = P_{ge}(q_1) = \emptyset$
$Inv(q_2) : P_{lt}(q_2) = P_{lt}(q), P_{eq}(q_2) = P_{eq}(q) \cup P_{le}(q) \cup P_{ge}(q), P_{gt}(q_2) = P_{gt}(q), P_{le}(q_2) = P_{ge}(q_2) = \emptyset$
refine $Q = (Q \backslash \{q\}) \cup \{q_1, q_2\}$

Step 1: The initial partition of the hybrid automaton is the partition induced by the switching surfaces and modal invariants of the system. The first step of the algorithm is to separate the interiors of the partitioned states from the boundaries, thus dividing the states into two classes, those that are defined by strict inequalities (i.e., interiors) and those that are defined by at least one equality (i.e., boundaries).

The initial separation of boundary and interior states is useful because it immediately returns a list of all the states adjacent to a particular state. In terms of implementation, this is useful because it allows transition checking only between adjacent states. This is possible because, for the type of automata under study, all transitions occur through the boundaries as a result of the continuous vector flow. Therefore, ensuring that transitions occur only between adjacent states is crucial.

Step 2
foreach $q, q_1 \in Q$, such that $P_{eq}(q) = \emptyset$, $P_{eq}(q_1) \neq \emptyset$, and $(P_{lt}(q) \cap P_{gt}(q_1)) = (P_{gt}(q) \cap P_{lt}(q_1)) = \emptyset$
if $\forall p_i \in (P_{lt}(q) \cap P_{eq}(q_1)), L_q(p_i)|_{Inv(q_1)} > 0$ **and** $\forall p_j \in (P_{gt}(q) \cap P_{eq}(q_1)), L_q(p_j)|_{Inv(q_1)} < 0$, where $L_q(p_{()})|_{Inv(q_1)}$ is the Lie derivative of $p_{()}$, w.r.t. $f(q, x)$, evaluated on $Inv(q_1)$ **then** $R(q) \rightarrow q_1$
foreach $q, q_1 \in Q : q \neq q_1$, such that $P_{eq}(q) \neq \emptyset$, $(P_{eq}(q) \subset P_{eq}(q_1)) \vee (P_{eq}(q_1) \subset P_{eq}(q))$, and $(P_{lt}(q) \cap P_{gt}(q_1)) = (P_{gt}(q) \cap P_{lt}(q_1)) = \emptyset$
if $\forall p_i \in (P_{eq}(q) \cap P_{eq}(q_1)), L_q(p_i)|_{Inv(q)} = 0$ **and** $\forall p_j \in (P_{eq}(q) \cap P_{gt}(q_1)), L_q(p_j)|_{Inv(q)} > 0$ **and** $\forall p_k \in (P_{eq}(q) \cap P_{lt}(q_1)), L_q(p_k)|_{Inv(q)} < 0$ **and** $\forall p_l \in (P_{lt}(q) \cap P_{eq}(q_1)), L_q(p_l)|_{Inv(q_1)} > 0$ **and** $\forall p_m \in (P_{gt}(q) \cap P_{eq}(q_1)), L_q(p_m)|_{Inv(q_1)} < 0$ **then** $R(q) \rightarrow q_1$

Step 2: The next step is to compute transitions between the partitioned states, based on the vector field of the continuous dynamics in each partition. Adjacency is checked strictly, since the automaton is restricted to have only forced transitions. The procedure for finding transitions is different for interior and boundary states. For interior states, the Lie derivatives, under the vector field in the partition, of each boundary polynomial is computed. Next, the sign of each Lie derivative is evaluated on the boundary itself. 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 can be determined. For boundary states, the sign of the Lie derivatives of the polynomial equalities defining it have to be evaluated first; if these Lie derivatives are non-zero (i.e. positive or negative), then the trajectories have to exit the state through those surfaces. If the Lie derivatives are zero then the other boundary polynomials are checked for exit transitions, as in the case of interior states. A boundary, given by a polynomial $p = 0$, of a state appears as an invariant for that state, as $p < 0$ or $p > 0$. For an exit transition to exist through that boundary $p = 0$, the Lie derivative $L(p)$ has to be of the appropriate sign, i.e. if $p < 0$ in the state, then $L(p) > 0$, and if $p > 0$ then $L(p) < 0$ will ensure that a transition exists between the state and the adjacent boundary with $p = 0$.

An interesting property of the Delta-Notch automaton, and the class of automata it belongs to, is that sign changes of the Lie derivative along a particular boundary do not occur, which makes it easier to partition. If the boundary and the dynamics are linear, then the sign of the Lie derivative is always computable. In general, if finite-connectedness between the partitions can be assumed, then the transition generation step also always terminates.

Steps 3,4
while $\exists q \in Q$, such that $|R(q)| > 1$
if $\exists q_1 \in Q$, such that $R(q) \rightarrow q_1$ **and** $|P_{eq}(q_1)| - |P_{eq}(q)| = 2$, if several possible q_1 exist, choose one at random
compute partitioning surface $g(x)$, such that $g(x)$ satisfies $\dot{x} = f(q, x)$ and $Inv(q_1)$
define q_2, q_3, q_4, such that $f(q_2) = f(q_3) = f(q_4) = f(q)$, and
$Inv(q_2) = Inv(q) \wedge g(x) < 0$ and $R(q_2) = R(q) \backslash \{\rightarrow q_i\}$, where q_2, q_i are not adjacent
$Inv(q_3) = Inv(q) \wedge g(x) = 0$ and $R(q_3) = R(q) \backslash \{\rightarrow q_i\}$, where q_3, q_i are not adjacent
$Inv(q_4) = Inv(q) \wedge g(x) > 0$ and $R(q_4) = R(q) \backslash \{\rightarrow q_i\}$, where q_4, q_i are not adjacent
if $\exists q_j : R(q_j) \rightarrow q$ **then** $R(q_j) = (R(q_j) \backslash \{\rightarrow q\}) \bigcup_i \{\rightarrow q_i\}$, where q_i, q_j are adjacent
refine $Q = (Q \backslash \{q\}) \cup \{q_2, q_3, q_4\}$

else if $\exists q_1 \in Q$, such that $R(q) \rightarrow q_1$ **and** $\exists p_i \in P_{eq}(q_1) : p_i \notin (P_{lt}(q) \cup P_{eq}(q) \cup P_{gt}(q))$
define q_2, q_3, q_4, such that $f(q_2) = f(q_3) = f(q_4) = f(q)$, and
$Inv(q_2) = Inv(q) \wedge p_i < 0$ and $R(q_2) = R(q) \backslash \{\rightarrow q_i\}$, where q_2, q_i are not adjacent
$Inv(q_3) = Inv(q) \wedge p_i = 0$ and $R(q_3) = R(q) \backslash \{\rightarrow q_i\}$, where q_3, q_i are not adjacent
$Inv(q_4) = Inv(q) \wedge p_i > 0$ and $R(q_4) = R(q) \backslash \{\rightarrow q_i\}$, where q_4, q_i are not adjacent
if $\exists q_j : R(q_j) \rightarrow q$ **then** $R(q_j) = (R(q_j) \backslash \{\rightarrow q\}) \bigcup_i \{\rightarrow q_i\}$, where q_i, q_j are adjacent
refine $Q = (Q \backslash \{q\}) \cup \{q_2, q_3, q_4\}$
set $Q_T = Q, \Sigma_T = \emptyset, \rightarrow = R, Q_{T,0} = Q_0$
return $T = (Q_T, \Sigma_T, \rightarrow, Q_{T,0})$

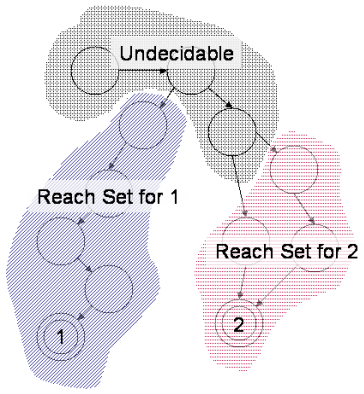


Fig. 1. Schematic state transition diagram showing approximate backward reachable sets from final states.

Step 3: Once a coarse partition with an associated transition map is computed for the hybrid automaton, a sub-partitioning step is applied to the states in the abstraction that have more than one exit transition. The sub-partitioning step divides the state into several subsets, each of which have exactly one exit transition, i.e. exactly one successor state. This step is non-trivial, as each new partitioning surface is a fully symbolic analytical solution to the continuous differential equations for the state. The sub-partitioning procedure is the most complex and is, in general, computationally intractable. Multiple transitions out of a state occur when the vector flow encounters the intersection of two or more boundaries of the state. The sub-partitioning surface is a set of trajectories that are exact solutions of the differential equations associated with that state, and that begins at the intersection of the boundaries. Computing analytic time-independent solutions to these sub-partitioning surfaces is not always possible because of the symbolic coefficients and indices. If the computation fails then this step will not yield a finer partition. However, for a large number of states, at least for hybrid automata with diagonal state transition matrices, this sub-partitioning is achievable.

Step 4 When a state is sub-partitioned as above, into several new states, its original predecessor state will now have multiple transitions and thus the partitioning algorithm will have to be iteratively applied to the predecessor states. The iteration has to be continued until all states in the partition have exactly one successor state, or none, in the case of partitions containing equilibria. This final partition, if computable, results in a discrete abstraction that is completely deterministic.

To understand the nature of the partition produced by the algorithm, its important properties are summarized here; the coarsest partition that will be produced is the one induced by the invariants of the hybrid automaton itself, without further refinement, and whose states won't have unique successor states, in general. When the refinement procedure is computable for a state, each subdivision will have an unique successor. If the iteration reaches a fixed point where all states have unique successors, then the abstraction is complete. Using the adjacency matrix of the deterministic transition system thus produced, an exact and uniquely backward reachable set of states can be computed from the equilibria of the automata.

Unfortunately, not all sub-partitions are computable. In that case, determinism for those states or their predecessor states up the sequence of transitions cannot be guaranteed. The predecessor states of the final states, if they transition only into the final state, will not have to be sub-partitioned, and so on iteratively for their predecessors. If chains of such states terminating in a final state emerge from the algorithm, a local fixed-point of the partition, i.e. a *partially* deterministic abstraction, will have been computed. Since the system is live, i.e. for all non-final states, a forced transition exists, the union of these states will give an under-approximation of the unique backwards reachable set from that equilibrium, because all points in those states are guaranteed to reach that final state and only that final state. It is an under-approximation because there might be other regions of state-space which converge to the final state of interest, however the refinement step fails to delineate them exactly, because of computability or decidability issues. Hence, in that case, the *best possible* partition, under the circumstances, will have been achieved (Fig. 1). Of course, in the worst case, there might not exist even a single predecessor state of the final state that can be sub-partitioned, and the approximate reachable set reduces to the final state itself. However, in the class of Delta-Notch automata that have been studied by the authors, large portions of the state space have been identified that are uniquely and exactly reachable from one equilibrium or the other.

V. Implementation

The model abstraction algorithm has been implemented using MATLAB, and the symbolic and string manipulations are also done in MATLAB. Each state of the current partition iteration is stored in a data structure that stores the signs of the invariant polynomials defining the state, the continuous vector flow associated with the state, and a list of the predecessor and the successor states. However, since

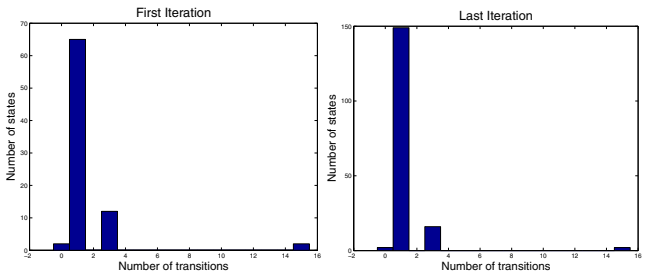
MATLAB does not have a decision procedure subroutine, the decision procedure on the polynomials while checking transitions is done using QEPCAD [4]. To reduce computational load, MATLAB and QEPCAD run on different computers and MATLAB communicates with QEPCAD through a TCP/IP socket every time a decision procedure run is required. The MATLAB program takes a canonical description of a hybrid system and parses it into states, invariants and associated continuous dynamics. For the first iteration, the program automatically computes the transition graph for the coarse initial partition. However, the iterative refinement procedure requires some manual analysis, as the MATLAB symbolic solver cannot always solve the differential equations for the new partitioning surface, which has to be done by hand. Then, the new states are added to the state vector and transitions are checked, if computable. Given below is an example of the input to the program, a portion of the definition of the hybrid automaton describing a two cell Delta-Notch system:

```
% Delta-Notch Signaling Hybrid Automaton
% Number of cells: 1x2
% Number of state variables: 4
% Number of discrete states: 16

Inv: -x2-hD < 0 /\ x3-hN < 0 /\ -x4-hD < 0 /\ x1-hN < 0
x1dot = -lD*x1
x2dot = -lN*x2
x3dot = -lD*x3
x4dot = -lN*x4,

Inv: -x2-hD < 0 /\ x3-hN < 0 /\ -x4-hD < 0 /\ x1-hN >= 0
x1dot = -lD*x1
x2dot = -lN*x2
x3dot = -lD*x3
x4dot = RN-lN*x4,
.
.
.
```

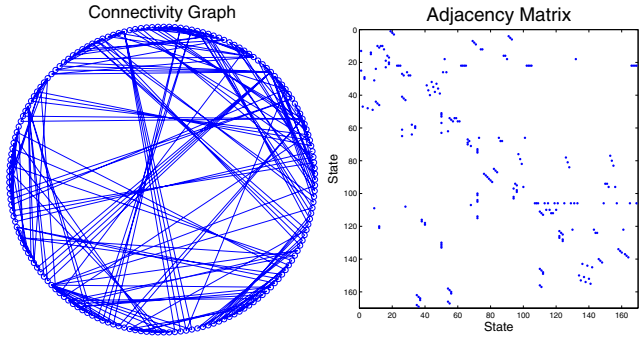
Each mode of the hybrid automaton is defined using invariant inequalities and the associated differential equations. For the two cell automaton abstraction, the algorithm starts with a coarse partition with 14 out 81 states that have more than one exit transition, and at the final iteration has 18 out 169 states that are unpartitionable. As shown in the histograms of Fig. 2, the number of states with multiple transitions rises slightly, however, as a percentage of the total number of states, it drops from around 18 per cent to around 10 per cent.



(a) (b)

Fig. 2. Histograms displaying the number of states which have a particular number of exit transitions.

The reachability computation is also done using MATLAB, on the adjacency matrix of the final abstraction using a post processing tool. The post processing tool can plot the connectivity graph of the discrete abstraction, as shown in Fig. 3(a), at any iteration stage. The adjacency matrix, displayed in Fig. 3(b), is utilized to compute the backward reachable states from an equilibrium state. Notice the sparsity of the adjacency matrix, this implies that the partition is at a stage where most states are finely refined, and have only a single exit transition. The states with multiple exit transitions are shown as rows with dots in multiple columns in Fig. 3(b), and further refinement may not be possible.



(a) (b)

Fig. 3. (a)The connectivity network graph of the discrete abstraction, each circle on the perimeter represents a state. (b) The adjacency matrix of the same discrete abstraction, each dot in position (i, j) represents a transition from state q_i to q_j.

VI. Results

The algorithm has been used to analyze the one cell, two cell and four cell Delta-Notch automata. The discrete abstraction of the one cell automaton is exact and the initial conditions converging to the equilibrium can be determined exactly. The computation for the two cell automaton is more interesting, as behavioral complexities arise from putting the two cells together in a simple network. The two cell hybrid automaton, H_{two_cell}, has four continuous states x_1, x_2, x_3, x_4, representing the Delta and Notch protein levels in cells 1 and 2 respectively and sixteen discrete states in which each protein's production is either turned on or off. Even though the vector flow in each state is relatively simple, being given by a piecewise affine differential equation with a diagonal state transition matrix, the switching between states makes the behavior of this system quite complex. After the algorithm had converged and no further partitions could be refined, the backward reachable sets from the two biologically observed equilibria were computed.

Since the state space is four dimensional, it is difficult to visualize the reachable sets. One visualization technique is to draw projections on to three dimensional space, as in Fig. 4. In all the projection diagrams, the cyan and green

sets are under-approximations of the backwards reachable sets from equilibrium 1 and 2, respectively. The two pictures in the top row of Fig. 4 are two different views of the reachable sets projected onto the $x_3 - x_1, x_4 - x_2, x_3$ space, to show their structure. The curved surface of the green reach set is actually a projection of one of the partitioning polynomials generated by the algorithm. The pictures on the bottom row are views of the projection on the $x_1, x_3, x_4 - x_2$ plane and show the spatial complexities of the refined partition.

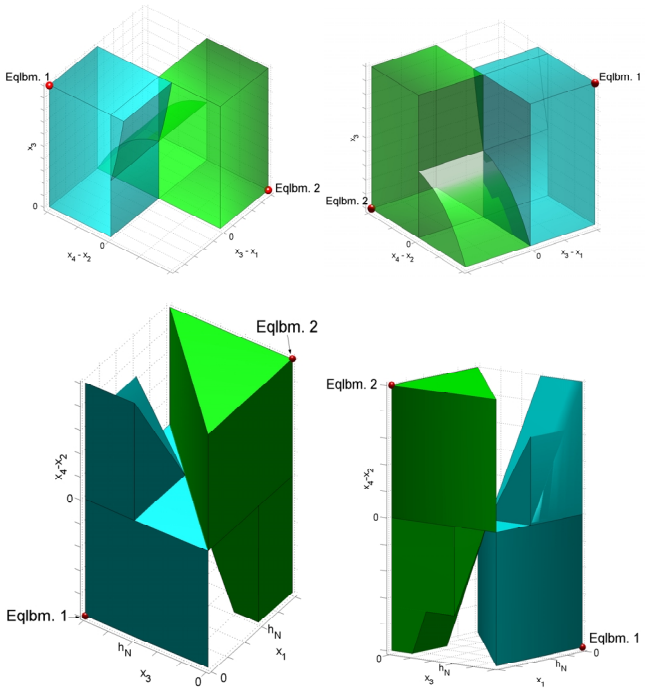


Fig. 4. Projections showing computed backward reachable set from the equilibria for the two cell Delta-Notch automaton. The cyan set represents the reachable set for equilibrium 1, and the green one for equilibrium 2.

Biological Significance

The two cell Delta-Notch system is a competitive network, and it tends to amplify differences in initial protein concentrations between the two cells. From the computed reachable sets, the following patterns of behavior of the system can be discerned: (a) If the initial protein concentrations are such that $x_3 - x_1 > 0 \wedge x_4 - x_2 < 0$, the system converges to equilibrium 1, i.e. x_2, x_3 have a high steady state concentration, and x_1, x_4 have a low steady state concentration. The network essentially amplifies the difference between the initial Delta protein concentrations of both cells and initial Notch concentrations between the two cells. As expected, for the symmetric case, if the initial concentrations are such that $x_3 - x_1 < 0 \wedge x_4 - x_2 > 0$, equilibrium 2 is attained, where x_1, x_4 have high steady state values and x_2, x_3 have low steady state values. (b) A Zeno trajectory exists when $x_1 = x_3 \wedge x_2 = x_4$ initially. This is because, if there are no initial differences, the network cannot converge to a steady state where there is a clear winner. In a nonlinear model, this would correspond to a *saddle* solution. (c) The third mode of behavior is very intriguing. The backward reachable sets indicate that there are certain sets of initial conditions, for both equilibria, where the system behaves non-intuitively: The initial difference in protein concentration between cells is not amplified. For example, with initial conditions $x_3 - x_1 < 0 \wedge x_4 - x_2 < 0$, the network can still converge to equilibrium 2, where at steady state $x_4 - x_2 > 0$. Therefore the difference in initial Notch protein concentrations is reduced and reversed, if the initial conditions satisfy some other conditions such as: $(x_3 - h_N \leq 0 \wedge -x_4 - h_D < 0 \wedge (1 - \frac{\lambda_N}{R_N}(x_4 - x_2))^{\lambda_D} \leq (\frac{x_1}{h_N})^{\lambda_N})$. The reason that such behavior is possible, is because the kinetic parameters for protein production and decay, and the switching thresholds, may be different for different proteins. Sets of initial conditions exist, where the initial concentration of some proteins are such that they can suppress, or promote, the production of other proteins long enough for the initial difference in concentrations between the two cells to be reduced or even reversed. This prediction of potential mutant behavior would be very interesting to test in a biological experiment.

VII. Conclusion

This paper describes the development and implementation of a new partitioning algorithm, using Lie derivatives and iterative refinement by exact solutions, to obtain discrete abstractions of piecewise affine autonomous hybrid automata with fully symbolic kinetic parameters. The discrete abstraction provides a substrate for computing under-approximations of backward reachable sets from the equilibria of these automata. The approximate reach set is then used, in a biological model, to determine initial conditions from which specific biologically observed steady states are reached. The authors' current research is focused on computing reachable sets for larger cell networks, and techniques to query them for biologically significant properties. Work is also in progress to apply the reachability analysis techniques developed here to the planar cell polarity (PCP) signaling network in *Drosophila* wings, which has been previously modeled as a hybrid automaton [5].

References

[1] R. Ghosh and C. Tomlin, "Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modelling: delta-notch protein signalling," *Systems Biology*, vol. 1, no. 1, pp. 170–183, June 2004.

[2] 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*, 2000, pp. 326–337.

[3] R. Ghosh and C. J. Tomlin, "Lateral inhibition through delta-notch signaling: a piecewise affine hybrid model," in *Hybrid Systems: Computation and Control*, ser. LNCS 2034, M. D. D. Benedetto and A. Sangiovanni-Vincentelli, Eds. Springer Verlag, 2001, pp. 232–246.

[4] H. Hong, "An improvement of the projection operator in cylindrical algebraic decomposition," in *Proc. ISAAC 90*, 1990, pp. 261–264.

[5] R. Ghosh, K. Amonlirdviman, and C. 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, pp. 1588–1594.