Accepted to appear in IEE Systems Biology

Symbolic Reachable Set Computation of Piecewise Affine Hybrid Automata and its Application to Biological Modelling: Delta-Notch Protein Signalling

Ronojoy Ghosh Claire Tomlin
Department of Aeronautics and Astronautics,
Stanford University, Stanford 94305, CA.
E-mail: ronojoy@stanford.edu, tomlin@stanford.edu

Abstract

Hybrid automata are an eminently suitable modelling framework for biological protein regulatory networks, as the protein concentration dynamics inside each biological cell are modelled using linear differential equations; inputs activate or deactivate these continuous dynamics through discrete switches, which themselves are controlled by protein concentrations reaching given thresholds. This paper proposes an iterative refinement algorithm for computing discrete abstractions of a class of hybrid automata with piecewise affine continuous dynamics and forced discrete transitions, defined completely in terms of symbolic variables and parameters. Furthermore, these discrete abstractions are utilized to compute symbolic parametric backward reachable sets from the equilibria of the hybrid automata, that are guaranteed to be exact or conservative under-approximations. The algorithm is then implemented using MATLAB and QEPCAD, to compute reachable sets for the biologically observed equilibria of multiple cell Delta-Notch protein signalling automaton with symbolic parameters. The results are analysed to show that novel, non-intuitive, and biologically interesting properties can be deduced from the reachability computation, thus demonstrating the utility of the algorithm.

1 Introduction

Biological cell networks exhibit complex combinations of both discrete and continuous behaviours; the dynamics that govern the spatial and temporal increase or decrease of protein concentration or activity inside a single cell are continuous differential equations, while the activation or deactivation of these continuous dynamics are triggered by switches which encode protein concentrations reaching given thresholds. Hybrid automata theory presents an ideal framework to model and analyse these processes, with the goal of generating predictions that can be experimentally verified. Formally, a hybrid automaton is a dynamical system with temporal evolution of continuous state variables governed by differential equations whose parameters change due to discrete input or event driven discrete state transitions. Their modularity and compositionality can be utilized to construct complex networked automata,

thus increasing their value as a tool for modelling and analysing the vast field of biological regulatory systems.

One specific analysis problem is the focus of this paper; the computation of sets of points or regions of the state space "backward reachable" from a particular equilibrium or steady state of the hybrid automaton, which means there are trajectories that lead from those regions to the steady state. In the context 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. In addition, 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.

Computing reachable sets for hybrid automata is difficult in general, 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 approximating the dynamics using linear hybrid automata [1, 2], and methods approximating the reachable set such as polyhedral representations [3, 4], piecewise affine systems [5], ellipsoidal approximations [6], and projections of convergent approximations [7, 8]. An interesting approach utilizing optimal control techniques has been developed by [9], which can analyse high dimensional constrained linear and piecewise affine systems. Recently, qualitative simulation models [10, 11] have been proposed to abstract continuous phase portraits of hybrid automata to simpler transition graphs, on which reachability analysis can be performed. Predicate abstraction [12, 13] and quantifier elimination [14] have been proposed for computing discrete abstractions of hybrid automata. Most of these methods suffer from one or both of two disadvantages; (a) the complexity of the computations on the hybrid automaton restricts its dimensionality, and more importantly, (b) symbolic computations are not possible. In previous work, quantifier elimination techniques have been used by the authors to compute over-approximations of the symbolic backward reachable sets for various Delta-Notch protein signalling automata [15].

In this paper, a novel algorithm is proposed 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. Important differences between the algorithm presented here and the previous reachability analysis performed [15] are: (a) The partitioning polynomials are solutions of the continuous dynamics of the hybrid automaton, and (b) the resulting reachable set is under-approximate. The biological focus of this paper is a particular mechanism of intercellular signalling known as the Delta-Notch signalling pathway, which is used in the differentiation of cell fate during embryonic development, through lateral inhibition. The objective is to present a multi-stable hybrid automaton model with symbolic parameters, that faithfully replicates the biological dynamics of lateral inhibition in a planar cell network, and then discuss the novel algorithm

to symbolically compute the regions of attraction of the steady states. The key attribute of this research effort is that both modelling and analysis is completely symbolic, i.e. none of the parameters such as protein production/activation and decay constants or switching thresholds are numerically instantiated. Rather, by doing symbolic analysis, predictions are generated that involve ratios of symbolic kinetic parameters (for example, the relative rates of production of two different proteins), resulting in a model with valid parameter ranges given in the form of constraints.
Section 2.1 describes the formal definitions of the piecewise affine hybrid automaton utilized in modelling, and the discrete transition system that it is abstracted to. In Section 2.2, the hybrid model of the Delta-Notch network is presented and its key properties summarized. In Section 3, the concepts associated with reachable set computation are defined, and in Section 4 the abstraction algorithm is described in detail. Finally, the reachable sets computed for the one, two and four cell models are presented in Section 5.

2 Model Development

2.1 Hybrid Automata and Transition Systems

This section formally defines a restricted class of hybrid automata that is used in the Delta-Notch model development, and for which the abstraction algorithm has been developed. This is followed by the definition of a discrete transition system that represents an abstraction of the hybrid automaton. A more general discussion related to abstractions of hybrid automata and their decidability can be found in [16].

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, or modes;*
2. $X \subset \boldsymbol{R}^n$ *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* $A_q \in \boldsymbol{R}^{n \times n}$ *is a diagonal matrix, and* $b_q \in \boldsymbol{R}^n$*;*
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.* $p_{()} : X \rightarrow \boldsymbol{R}$ *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 are free to be symbolic, i.e. the eigenvalues, $\lambda_1, \lambda_2, \ldots \lambda_n$, of A_q need not be numerically instantiated. The elements of vector b_q are also free to be symbolic. Constraints may be imposed on these symbolic constants to restrict

the behaviour of the model. The polynomials defining the invariant of each mode, 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, or less than (lt) zero, 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 neighbours in state space, in several different steps of the abstraction algorithm presented in Section 4. It should be noted that, as the abstraction procedure progresses, additional polynomials are added to the modal invariants to partition the modes. This may give rise to redundancies in the polynomials defining the modal invariants. The redundant constraints can be removed from the invariant using a decision procedure such as QEPCAD at every partitioning step. However, this is unnecessary because the adjacency check performed during transition computation will ensure that there are no transitions between non-adjacent modes, thus taking care of the redundant constraints in the invariant. 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. When a network of automata is built by composing several of them together, their discrete inputs, Σ are coupled to the internal state variables of other automata in the network, as will be seen in the multiple cell Delta-Notch network model in Section 2.2. In that case, the entire network behaves as an autonomous hybrid automaton, as a whole. If the assumption of zero boundary conditions, i.e. no influence from outside the network, is made then the state transitions in that automaton are all forced.

Example

To illustrate the definition of the invariant, consider the example of a hybrid automaton with two continuous state variables, x_1 and x_2, and two discrete states (or modes), q_1 and q_2. The state space of this automaton is two dimensional and the geometrical interpretation of modal invariants can be visualized directly (Figure 1).

Discrete states q_1 and q_2 are defined by assigning signs to the polynomials $x_1 + a_i x_2 + b_i, i = 1 \ldots 5$, i.e. the invariant defining q_1 is $x_1 + a_1 x_2 + b_1 < 0 \wedge x_1 + a_2 x_2 + b_2 \geq 0 \wedge x_1 + a_3 x_2 + b_3 < 0 \wedge x_1 + a_4 x_2 + b_4 < 0 \wedge x_1 + a_5 x_2 + b_5 \geq 0$. Each of the sets $P_{lt}(q_1), P_{eq}(q_1), \ldots, P_{ge}(q_1)$ are lists that can now be populated by polynomial expressions according to what sign they had in the invariant of q_1. Therefore, it can be seen that $P_{lt}(q_1) = \{x_1 + a_1 x_2 + b_1, x_1 + a_3 x_2 + b_3, x_1 + a_4 x_2 + b_4\}$ and $P_{ge}(q_1) = \{x_1 + a_2 x_2 + b_2, x_1 + a_5 x_2 + b_5\}$, and the others are empty, $P_{eq}(q_1) = P_{gt}(q_1) = P_{le}(q_1) = \emptyset$. Similarly, for state q_2, the lists of polynomials are $P_{lt}(q_2) = \{x_1 + a_1 x_2 + b_1, x_1 + a_4 x_2 + b_4\}$, $P_{ge}(q_2) = \{x_1 + a_2 x_2 + b_2, x_1 + a_3 x_2 + b_3, x_1 + a_5 x_2 + b_5\}$, and $P_{eq}(q_2) = P_{gt}(q_2) = P_{le}(q_2) = \emptyset$. As previously mentioned, these lists are used to determine whether two discrete states are geometrically next to each other in continuous state space. In the example, the polynomial expression $x_1 + a_3 x_2 + b_3$ has different signs in the two states, it is an element of $P_{lt}(q_1)$, but is also a member of $P_{ge}(q_2)$. This implies that the sign change occurs at the boundary $x_1 + a_3 x_2 + b_3 = 0$, which is part of q_2 and that the two discrete states q_1 and q_2 are geometrically contiguous or adjacent, as can be seen in Figure 1. This test can be performed automatically and efficiently for high dimensional hybrid automata to check adjacency.

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

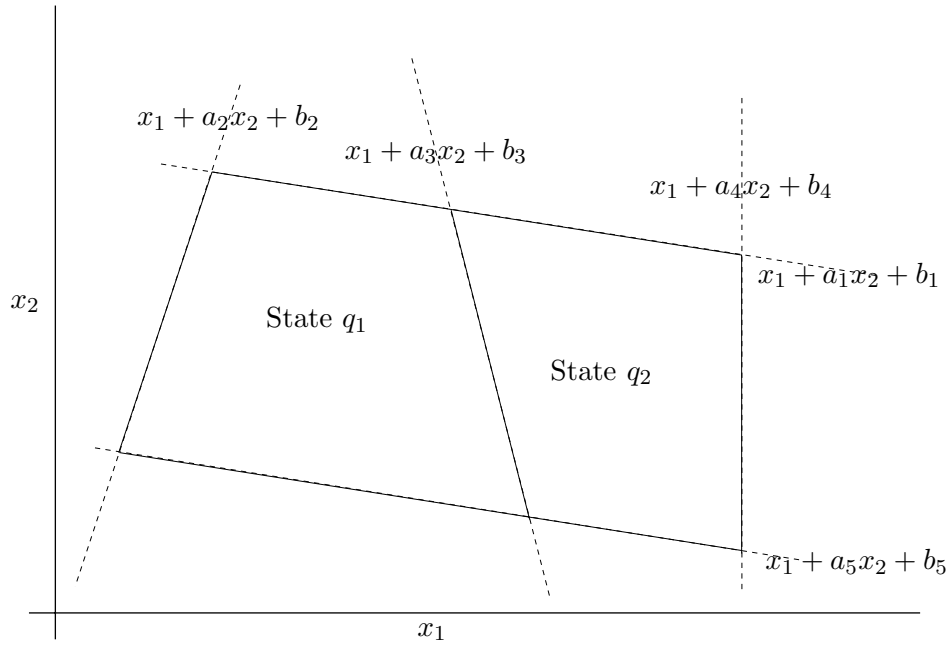


Figure 1: Continuous state space with geometrical representations of polynomial modal invariants of a two dimensional hybrid automaton.

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 T can be thought of as a graph with directed edges denoting transitions between nodes that are representations of the states $q \in Q$. 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. Note that the event σ, which triggers a transition, is not relevant here and has been dropped from the transition relation. The final states, $q \in Q_F$, of the transition system are states that have no transitions out of them, i.e. in the adjacency matrix, $\forall j, a_{final\_state,j} = 0$.
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.

Nondeterminism

The piecewise affine hybrid automaton defined in this section is deterministic, in the sense that given an initial condition, the numerical value of the constants in the modal invariants and differential equations in each mode, and the values of the discrete inputs, the exact trajectory of the continuous state variables, as well as the sequence of mode transitions, is unique. However, for computing the evolution of regions of the continuous states space, for example, finding all initial conditions that converge to a specific steady state, it is computationally more attractive to study the hybrid automaton as an abstracted discrete state transition system. The temporal evolution of the continuous state variables is abstracted away, resulting in a transition system with discrete states (that are the same as the discrete states of the hybrid automaton) connected by transitions triggered by the set of events Σ.

The abstraction of the piecewise affine hybrid automaton described above may give rise to nondeterminism in the abstracted discrete transition system. This is because the discrete states of the hybrid automaton are regions in state space, inside which trajectories starting from different points may exit the state through different boundaries. Therefore, the discrete states may have more than one transition out of them. The abstraction algorithm proposed in this paper attempts to eliminate, or at least reduce, the nondeterminism in the abstracted system by iteratively partitioning the discrete states such that each new discrete state has exactly one transition out of it.

An important quantity that will be used extensively in the abstraction algorithm is the Lie derivative of a scalar quantity along the flow of a vector field. This is formally defined as follows:

Definition 3 *The Lie derivative a smooth scalar function $V : \boldsymbol{R}^n \rightarrow \boldsymbol{R}$, with respect to a smooth vector field $f : \boldsymbol{R}^n \rightarrow \boldsymbol{R}^n$, is defined as $L_f V = \frac{\partial V}{\partial x} f(x)$.*

The sign of the Lie derivative is useful in determining the direction of flow of the vector field f, a property that is used in computing transitions in the abstraction algorithm in Section 4.

2.2 Hybrid Automaton Model for Delta-Notch Signaling

The Delta-Notch protein signaling mechanism has been identified as a key player in several different development processes, including pattern formation due to lateral inhibition [17], and is conserved across a broad spectrum of organisms. 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 neighbours. Second, Notch production is turned on by high Delta levels in the immediate neighbourhood 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 neighbours in contact with it (Figure 2(a)). An influence diagram showing the signaling network is shown in Figure 2(b).

Each biological cell is modelled as a four state piecewise affine hybrid automaton. The four states capture the property that Notch and Delta protein production can be individually

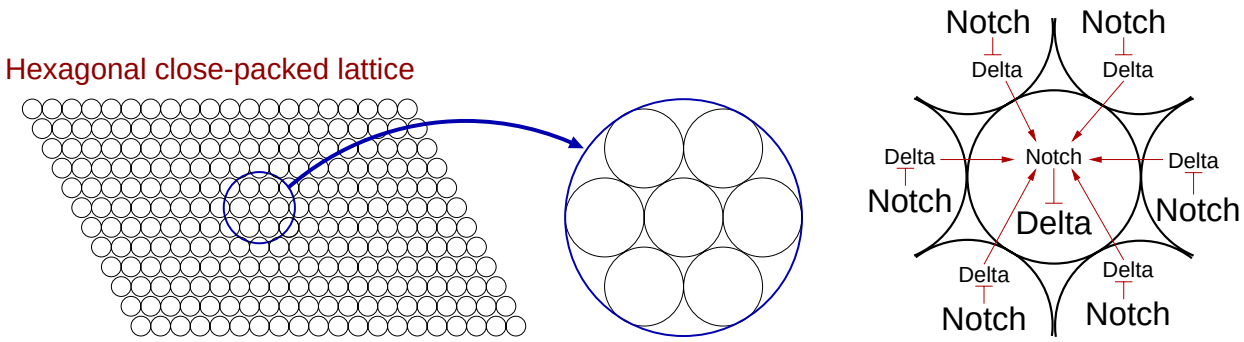


Figure 2: (a) Hexagonal close-packed layout scheme for cells in two dimensional arrays. (b) Influence diagram for Delta-Notch protein signaling network.

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:

$$
\begin{aligned}
H_{one\_cell} &= (Q, X, \Sigma, Init, f, Inv, R) \\
Q &= \{q_1, q_2, q_3, q_4\} \\
X &= (x_1, x_2)^T \in \boldsymbol{R}^2 \\
\Sigma &= \left\{ u_N = \sum_{i=1}^{6} x_{Delta,i} \right\} \\
Init &= Q \times \left\{ X \subset \boldsymbol{R}^2 : x_1, x_2 > 0 \right\} \\
f(q,x) &= \begin{cases} [-\lambda_D x_1; -\lambda_N x_2]^T & \text{if } q = q_1 \\ [R_D - \lambda_D x_1; -\lambda_N x_2]^T & \text{if } q = q_2 \\ [-\lambda_D x_1; R_N - \lambda_N x_2]^T & \text{if } q = q_3 \\ [R_D - \lambda_D x_1; R_N - \lambda_N x_2]^T & \text{if } q = q_4 \end{cases} \\
Inv &= \{q_1, \{-x_2 < h_D, u_N < h_N\}\} \cup \{q_2, \{-x_2 \geq h_D, u_N < h_N\}\} \\
&\quad \cup \{q_3, \{-x_2 < h_D, u_N \geq h_N\}\} \cup \{q_4, \{-x_2 \geq h_D, u_N \geq h_N\}\}
\end{aligned}
$$

where, x_1 and x_2: Delta and Notch protein concentrations, respectively, in a cell; $x_{Delta,i}$: Delta protein concentration in i^{th} neighbouring cell; λ_D and λ_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. The switching thresholds h_D and h_N are unknown and possible ranges for them have been derived by the authors [18] using equilibrium analysis.
In the single cell, $x_{Delta,i} = 0, \forall i \in \{1, \ldots 6\}$, as there are no neighbours whose Delta protein levels can be sensed. 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_{two_cell} is the composition of two single cell automata, to form a model with four continuous states $(x_1, \ldots, x_4)$ 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 neighbour to control Notch production. Modeling the full two dimensional layer of cells involves composing several single cell hybrid automata, with the coupling through the input functions as described above. It should be

noted that the multiple cell models developed above fall in the class of restricted hybrid automata defined in Section 2.1.

2.3 Equilibrium Analysis

The hybrid automata models described in the previous section have, in general, multiple steady states. The continuous dynamics in each discrete state, described by a system of differential equations, has a locally stable equilibrium point. However, not all the possible equilibria are compatible with biologically observed steady state lateral inhibitory patterning phenotypes, which follow the following rules:

1. No two cells with high Delta protein concentrations can lie next to each other.
2. A cell with high Notch protein concentration must have at least one neighbour with high Delta protein concentration.
3. A cell with high Delta protein concentration has low Notch protein concentration and vice versa.

This implies that the model parameters have to be systematically constrained so that a network of arbitrary size has only the biologically observed steady states and none others. This is achieved through equilibrium analysis, i.e. by examining the conditions necessary for a particular equilibrium in a discrete state to exist. Both the single and two cell hybrid automata have been analysed by the authors [18], to obtain constraints on the ranges of the protein kinetic parameters and switching thresholds for biologically feasible equilibria to exist, given by:

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

The constraints are the same for both the single and two cell automata, and similar constraints have been computed for larger cell networks. It leads to the conclusion that those constraints depend only on the number of neighbours a cell has. Since the maximum number of neighbours that it can have in a hexagonal close packed planar network is six, therefore the constraints are network size independent for networks larger than nine cells. In summary, the important properties of the hybrid automaton model of Delta-Notch protein signaling are:

1. 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.
2. 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 the trajectories of the system are continuous, though not necessarily smooth.

3. The number of discrete states that contain equilibria are finite and enumerable, and the equilibria are in the interior of each state. Moreover, equilibrium analysis [18] has shown that additional constraints on the system parameters (rate constants and switching thresholds) can restrict the existence of equilibria to biologically feasible modes.

4. The system is *live*, i.e. forced transitions exist for all states that do not contain equilibria.

3 Reachability

The fundamental goal of this work is to find initial conditions on the cellular protein concentration or activity, from which a particular biological steady state is achievable. In control theoretic terms, this is equivalent to identifying exact regions of attraction of the multiple stable equilibria of the hybrid automaton model. Since computing symbolically on that model directly is infeasible, the abstraction process proposed in this paper converts the hybrid automaton into a discrete transition system that preserves the transition structure while abstracting away the continuous dynamics. To better understand the computation of the region of attraction, the concept of *reachability* is now introduced. All notation regarding sets and variables refer to the hybrid automaton and transition system defined in Section 2.1.

Definition 4 *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. It can be shown that, for a piecewise affine hybrid automaton H with diagonal state transition matrices that conforms to the definition in Section 2.1, continuous flow in the discrete states containing stable equilibria does not cross the boundaries of the state and exit the state. This means that the equilibria-containing states have no forced transitions out of them, i.e. once the automaton has entered one of those states it will stay in that state and converge to the stable equilibrium contained within. If the hybrid automaton H is abstracted to the discrete transition system T, these equilibrium-containing states are then abstracted to the states in the set of final states Q_F, which, by definition, have no transitions out of them.
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.
It follows that the accuracy of the computed backward reachable set depends critically on the granularity of the abstraction. The finer the abstraction, the better the accuracy of

the reachable set. 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, because of decidability issues, using the proposed algorithm in this paper, it is possible to construct a "best possible" abstraction that is partially deterministic and partially nondeterministic, from which an under-approximation of the backward reachable sets from the equilibria can be determined.

4 Algorithm

This section describes, in detail, the partitioning algorithm developed to iteratively refine an initial coarse partition of the state space of a hybrid automaton. This algorithm is applicable to the restricted class of hybrid automata defined in Section 2.1, and the multiple cell Delta-Notch model developed in Section 2.2 is a perfect candidate for its application. 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. Figure 3(a) shows an example of a two dimensional hybrid automaton with the invariants defining the discrete states and appropriate continuous vector fields in each discrete state. In Figure 3(b), each discrete state is further partitioned into an interior and several boundaries.

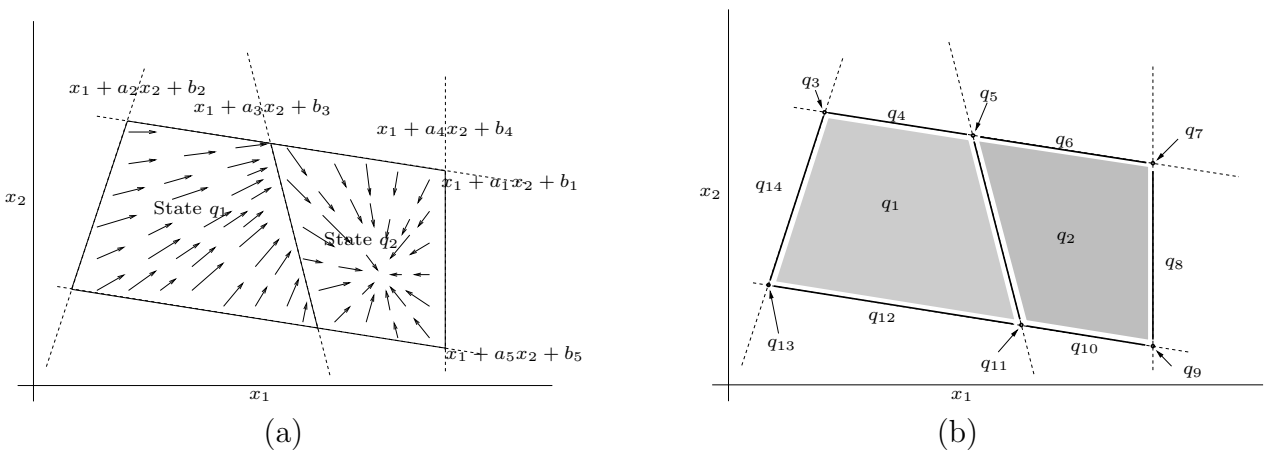


Figure 3: (a) Phase portrait of a hybrid automaton showing system dynamics and discrete state partitions. (b) States partitioned into boundary (switching surface) and interior.

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. Also, if the boundary and the dynamics are linear, then the sign of the Lie derivative is always computable using a decision procedure. In general, if finite-connectedness between the partitions can be assumed, then the transition generation step also always terminates. The vector field associated with each discrete state utilized in computing Lie derivatives is shown in Figure 4(a), and the transitions, computed from the sign of the Lie derivatives, between the states can be seen in Figure 4(b).

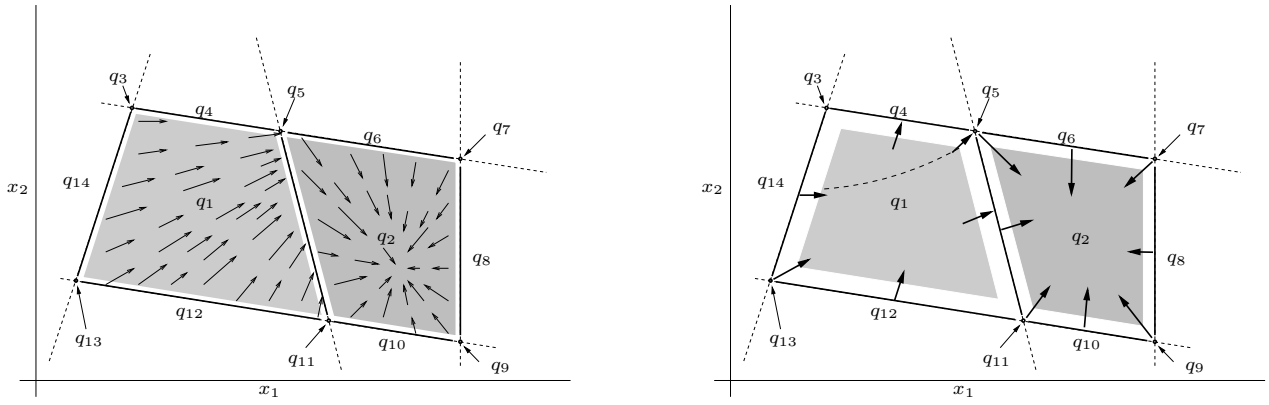


Figure 4: (a) The vector field in each discrete state is used to compute Lie derivatives that determine discrete state transitions. (b) Transitions computed between the discrete states, by abstracting the vector field. Note that there may be several transitions out of one discrete state (for example, state q_1).

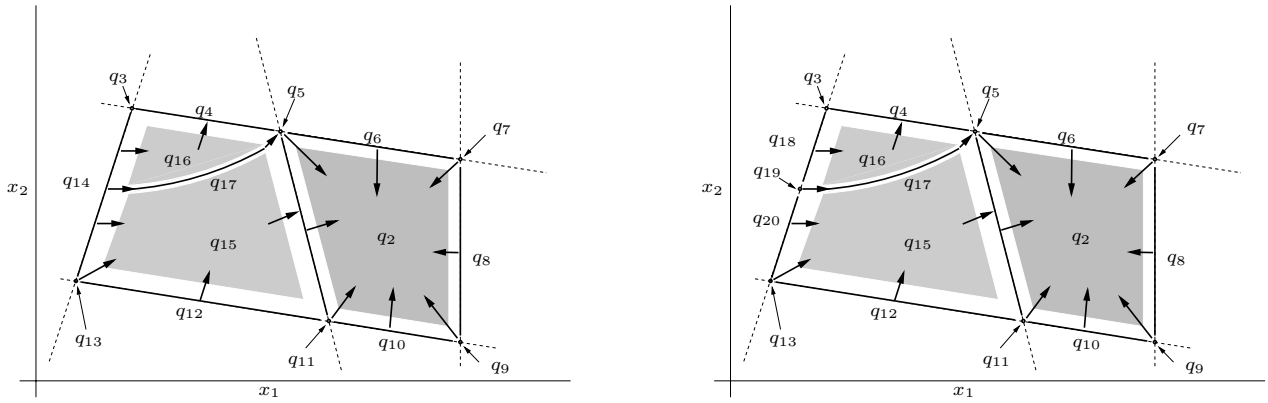
Steps 3,4
while $\exists q \in Q$, such that $|R(q)| > 1$
if $\exists q_1 \in Q$, such that $R(q) \rightarrow q_1$ **and** $|P_{eq}(q_1)| - |P_{eq}(q)| = 2$, if several possible q_1 exist, choose one at random
compute partitioning surface $g(x)$, such that $g(x)$ satisfies $\dot{x} = f(q, x)$ and $Inv(q_1)$
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 $T = (Q_T, \Sigma_T, \rightarrow, Q_{T,0})$

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 in general, the most 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 (Figure 5(a)). 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 (Figure 5(b)). The iteration has to be continued till 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.

Computation of Partitioning Surfaces The refinement procedure in step 3, of the algorithm, depends upon the ability to compute analytic time-independent solutions to the symbolic differential equations in a discrete state. The solution also has to satisfy the boundary condition generated by the intersection of two or more boundaries of the state through which the forced transitions occur. This is computationally intractable in general. However, from computing the abstraction for Delta-Notch automata using the proposed algorithm, it is the authors' experience that a majority of states either need no refinement (i.e., they have exactly one exit transition, even after their successor states have been partitioned), or have computable refining partitions. The observed properties that make the computation more tractable are:



(a) (b)

Figure 5: (a) Subdivision of state q_1 with multiple transitions, into states with one transition each. The dividing polynomial is an exact solution of the differential equations governing continuous flow in q_1 (b) Iterative application of the refinement procedure to predecessor states of q_1.

1. The continuous dynamics in each discrete state are governed by ordinary differential equations with diagonal state transition matrices A_q. This implies that the time-dependent symbolic solution to the differential equations can always be found and is given by the transcendental equation $x(t) = x(0)e^{A_q t}$. The boundary condition, i.e. intersection of boundaries of the discrete state, supplies the vector $x(0)$ and time t then has to be eliminated to give the closed form time-independent solution to the partitioning surface.

2. The polynomials that define the boundaries of the discrete states are linear. This is important because the intersection of the boundaries gives the boundary condition that the partitioning surface must satisfy. In addition, if the boundaries are also linearly independent, then the boundary condition will be simpler and it is easier to eliminate time t from the system of transcendental equations.

3. The discrete states have boundaries that are orthogonal to each other. If the boundaries are orthogonal, the computation of the intersection of the partitioning polynomial and the other boundaries is simplified. If, from property 2, the boundaries are linear and also linearly independent, computation of the intersection in this case simply reduces to the conjunction of the boundary and the new partitioning surface.

Example
An example of the partition refinement computation illustrates the properties enumerated above. To demonstrate the sub-partitioning step, consider a state from the two cell Delta-Notch automaton H_{two_cell}, defined as:

$$q_{10} : Inv(q_{10}) = -x_2 - h_D < 0 \wedge x_3 - h_N < 0 \wedge -x_4 - h_D < 0 \wedge x_1 - h_N > 0 \wedge x_4 - x_2 < 0 \wedge x_3 - x_1 < 0$$

with continuous dynamics, $f(q_{10}, x) = [-\lambda_D x_1; -\lambda_N x_2; -\lambda_D x_3; R_N - \lambda_N x_4]^T$. The polyno-

mials defining the invariant are linear and the solution to the continuous dynamics is:

$$\begin{aligned} x_1 &= x_1(0)e^{-\lambda_D t} \\ x_2 &= x_2(0)e^{-\lambda_N t} \\ x_3 &= x_3(0)e^{-\lambda_D t} \\ x_4 &= \tfrac{R_N}{\lambda_N} + \tfrac{x_4(0)\lambda_N - R_N}{\lambda_N} e^{-\lambda_N t} \end{aligned}$$

State q_{10} has three exit transitions, $R(q_{10}) \rightarrow \{q_{11}, q_{13}, q_{14}\}$, where the successor states are defined as:

$$\begin{aligned} q_{11} &: Inv(q_{11}) = -x_2 - h_D < 0 \wedge x_3 - h_N < 0 \wedge -x_4 - h_D < 0 \wedge x_1 - h_N > 0 \wedge x_4 - x_2 = 0 \wedge x_3 - x_1 < 0 \\ q_{13} &: Inv(q_{13}) = -x_2 - h_D < 0 \wedge x_3 - h_N < 0 \wedge -x_4 - h_D < 0 \wedge x_1 - h_N = 0 \wedge x_4 - x_2 < 0 \wedge x_3 - x_1 < 0 \\ q_{14} &: Inv(q_{14}) = -x_2 - h_D < 0 \wedge x_3 - h_N < 0 \wedge -x_4 - h_D < 0 \wedge x_1 - h_N = 0 \wedge x_4 - x_2 = 0 \wedge x_3 - x_1 < 0 \end{aligned}$$

Note that state q_{14} is the intersection of two boundaries $x_1 - h_N = 0 \wedge x_4 - x_2 = 0$, shown in the $x_1 - h_N, x_4 - x_2$ projection plane in Figure 6. Therefore, the partitioning polynomial for q_{10} will be the solution to $\frac{d(x_4-x_2)}{d(x_1-h_N)}\Big|_{f(q_{10},x)}$ that satisfies $x_1 - h_N = 0 \wedge x_4 - x_2 = 0$. From the solution to the continuous dynamics, it is easy to derive:

$$x_4 - x_2 = \frac{R_N}{\lambda_N} + \frac{(x_4(0) - x_2(0))\lambda_N - R_N}{\lambda_N} e^{-\lambda_N t}$$

Substituting the boundary conditions $x_4(0) - x_2(0) = 0$ and $x_1 - h_N = 0$, the time-dependent

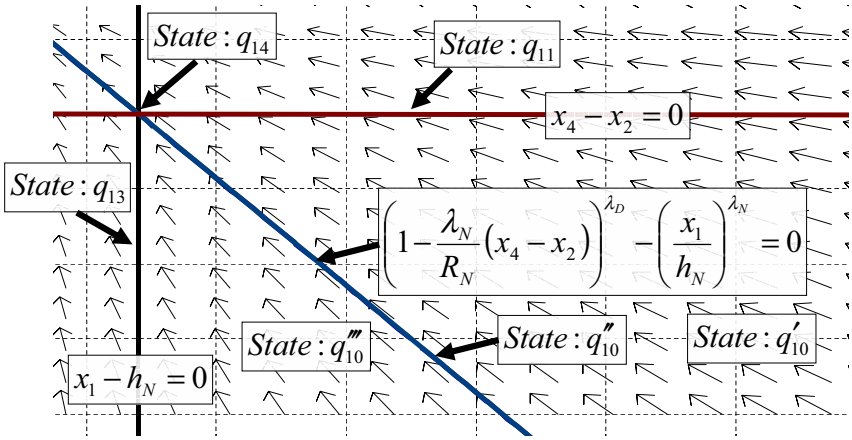


Figure 6: Generation of sub-partitioning surfaces for iterative refinement of partitions. This diagram shows the projection of the sub-partitioning surface of state q_{10} of a two cell Delta-Notch hybrid automaton.

solution for the partitioning surface is obtained:

$$\begin{aligned} x_1 &= h_N e^{-\lambda_D t} \\ x_4 - x_2 &= \tfrac{R_N}{\lambda_N}(1 - e^{-\lambda_N t}) \end{aligned}$$

By eliminating time t from the two transcendental equations, the time-independent equation of the partitioning surface p is obtained:

$$p = \left(1 - \frac{\lambda_N}{R_N}(x_4 - x_2)\right)^{\lambda_D} - \left(\frac{x_1}{h_N}\right)^{\lambda_N} = 0$$

This subdivides q_{10} into three states $q'_{10}, q''_{10}, q'''_{10}$, such that:

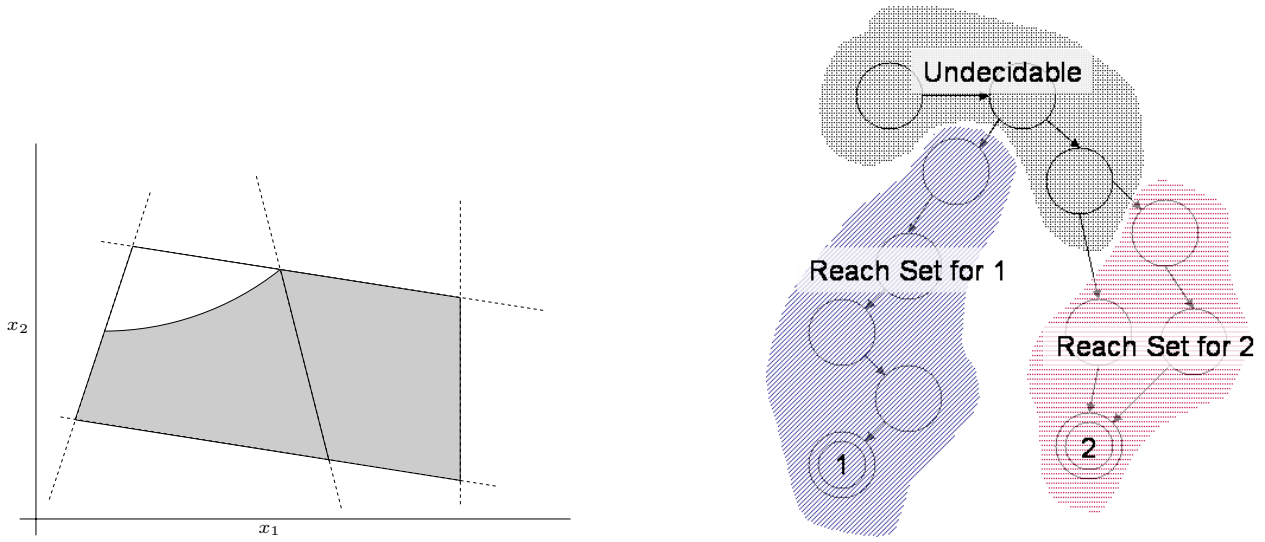
$$q'_{10} : Inv(q'_{10}) = Inv(q_{10}) \wedge \left(1 - \frac{\lambda_N}{R_N}(x_4 - x_2)\right)^{\lambda_D} - \left(\frac{x_1}{h_N}\right)^{\lambda_N} < 0 \text{ and } R(q'_{10}) \rightarrow q_{11}$$
$$q''_{10} : Inv(q''_{10}) = Inv(q_{10}) \wedge \left(1 - \frac{\lambda_N}{R_N}(x_4 - x_2)\right)^{\lambda_D} - \left(\frac{x_1}{h_N}\right)^{\lambda_N} = 0 \text{ and } R(q''_{10}) \rightarrow q_{14}$$
$$q'''_{10} : Inv(q'''_{10}) = Inv(q_{10}) \wedge \left(1 - \frac{\lambda_N}{R_N}(x_4 - x_2)\right)^{\lambda_D} - \left(\frac{x_1}{h_N}\right)^{\lambda_N} > 0 \text{ and } R(q'''_{10}) \rightarrow q_{13}$$

The intersections of the partitioning surface p with the other boundaries of state q_{10} are easy to compute because they are linearly independent from p. For example, the intersection of $p = 0$ and the boundary $x_3 - h_N = 0$ is simply the hyperplane $p = 0 \wedge x_3 - h_N = 0$. The intersection of $p = 0$ and another boundary $-x_4 - h_D = 0$ is given by substituting $x_4 = -h_D$ in $p = 0$:

$$\left(1 - \frac{\lambda_N}{R_N}(-h_D - x_2)\right)^{\lambda_D} - \left(\frac{x_1}{h_N}\right)^{\lambda_N} = 0$$

■

Therefore for certain parts of the state space, the exact partition will have been found, even though other parts may be unresolvable due to computability issues. This has profound implications for computing reachability from an equilibrium state, which is the primary motivation for the abstraction. It should also be mentioned that, if the symbolic constants in the model such as the thresholds h_D, h_N and the parameters λ_N, R_D are instantiated using integers, the computability issues may be ameliorated and the partition could be decidable. However, for the purposes of biological systems modelling, it is preferable to work in terms of symbolic constants, as it is difficult to obtain their correct numerical values.



(a) (b)

Figure 7: (a) An example of reachability computation using the abstracted transition system, the gray shaded area is completely reachable from the final state. (b) Schematic state transition diagram showing approximate backward reachable sets from final states.

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. However, if the automaton has one or more discrete states containing an equilibrium, and those have diagonal state transition matrices, it can be shown that those states will not have exit transition, i.e. they are final states of the automaton. 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 "partial" 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 uniquely backward reachable state 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 (Figure 7(b)). 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, as will be shown in Section 5.
The computational complexity of the algorithm restricts the size of the problem (i.e., the number of variables and parameters it can have). In terms of time-complexity, the most restrictive component is the symbolic decision procedure used to compute transitions across boundaries, which is doubly exponential in the number of variables and symbolic parameters, and polynomial in the number of polynomials defining each discrete state. Storage requirements rise exponentially in the number of discrete states, but that is manageable given the advances in computer memories and hard drive storage capacities. Some optimization has been performed in the transition checking portion of the algorithm in the implementation, which is, however, outside the scope of this paper.

5 Results

The algorithm has been implemented for one, two, 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 behavioural complexities arise from putting the two cells together in a simple network. Note that the number of discrete states mentioned in the introduction to each network is the initial number of states in the problem specification. The iterative

refinement procedure greatly increases the number of discrete states in the final partition.

5.1 One Cell Delta-Notch Automaton

For the one cell Delta-Notch automaton, the partitioning algorithm is decidable and returns an exact discrete abstraction. The state space is divided into three states (note that h_D is constrained to be negative):
$q_1 : Inv(q_1) = x_1 > 0 \wedge x_2 + h_D > 0$ and $q_1 \rightarrow q_2$
$q_2 : Inv(q_2) = x_1 > 0 \wedge x_2 + h_D = 0$ and $q_2 \rightarrow q_3$
$q_3 : Inv(q_2) = x_1 > 0 \wedge x_2 + h_D < 0$ and $q_3 \rightarrow \emptyset$: contains equilibrium

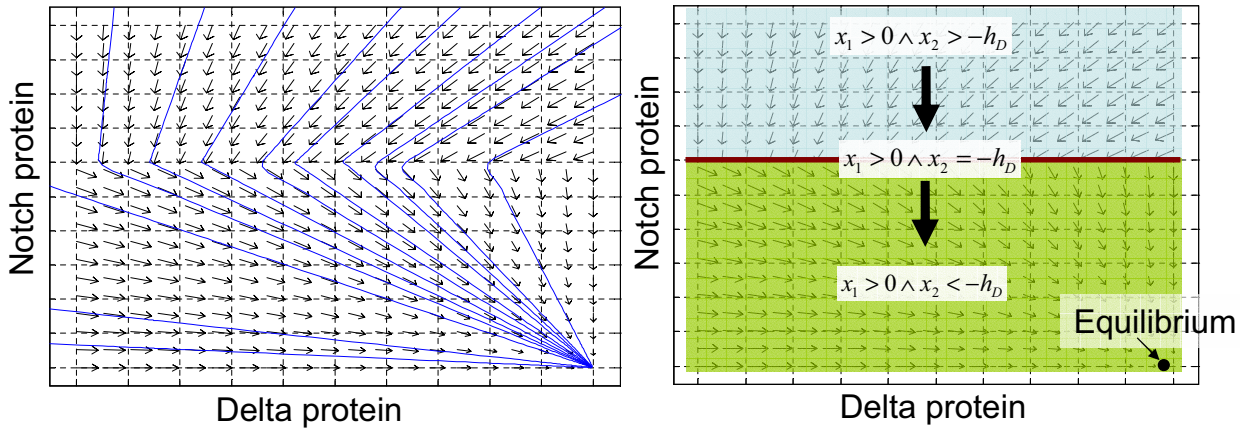


Figure 8: Exact discrete abstraction for a single cell Delta-Notch automaton.

Computing the backward reachable set from the equilibrium state q_3, it can be seen that $Reach(q_3) = q_1 \cup q_2 \cup q_3$, which is the entire state space $x_1 > 0 \wedge x_2 > 0$. The abstracted state space is shown, along with the actual phase portrait of H_{one_cell} in Figure 8. Biologically, the reach set implies that, in vacuo, each cell will converge to a steady state where it has a high level of Delta protein and a low level of Notch protein.

5.2 Two Cell Delta-Notch Automaton

The partitioning algorithm was applied to the two cell hybrid automaton H_{two_cell}, which 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 make the behaviour 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. The resulting reachable sets are given as:
Equilibrium 1: $\left(x_1 = 0, x_2 = \frac{R_N}{\lambda_N}, x_3 = \frac{R_D}{\lambda_D}, x_4 = 0\right)$
$(x_3 - x_1 \geq 0 \wedge x_4 - x_2 \leq 0 \wedge ((x_3 - x_1 > 0 \wedge h_D + x_4 \geq 0) \vee (x_3 - x_1 > 0 \wedge h_D + x_2 \leq 0) \vee$
$(x_4 - x_2 < 0 \wedge h_N - x_3 \geq 0) \vee (x_4 - x_2 < 0 \wedge h_N - x_1 \leq 0))) \vee (x_3 - x_1 < 0 \wedge x_4 - x_2 < 0 \wedge$
$((-x_2 - h_D < 0 \wedge x_1 - h_N \leq 0 \wedge -x_4 - h_D > 0 \wedge (1 - \frac{\lambda_D}{R_D}(x_3 - x_1))^{\lambda_N} \leq (\frac{x_2}{-h_D})^{\lambda_D}) \vee$

$(-x_2 - h_D \leq 0 \wedge x_3 - h_N \geq 0 \wedge -x_4 - h_D > 0 \wedge (1 - \frac{\lambda_D}{R_D}(x_3 - x_1))^{\lambda_N} \leq (\frac{R_N - \lambda_N x_4}{R_N + \lambda_N h_D})^{\lambda_D}))\vee$
$(x_3 - x_1 > 0 \wedge x_4 - x_2 > 0 \wedge ((-x_2 - h_D < 0 \wedge x_3 - h_N > 0 \wedge x_1 - h_N \leq 0 \wedge x_4 - x_2 \leq \frac{-R_N}{\lambda_N}(1 - (\frac{x_3}{h_N})^{\frac{\lambda_N}{\lambda_D}}))\vee$
$(x_3 - h_N \geq 0 \wedge -x_4 - h_D \geq 0 \wedge x_1 - h_N < 0 \wedge x_4 - x_2 \leq \frac{-R_N}{\lambda_N}(1 - (\frac{x_1}{h_N})^{\frac{\lambda_N}{\lambda_D}})))$

Equilibrium 2: $\left(x_1 = \frac{R_D}{\lambda_D}, x_2 = 0, x_3 = 0, x_4 = \frac{R_N}{\lambda_N}\right)$

$(x3 - x_1 \leq 0 \wedge x_4 - x_2 \geq 0 \wedge ((x_3 - x_1 < 0 \wedge h_D + x_2 \geq 0) \vee (x_3 - x_1 < 0 \wedge h_D + x_4 \leq 0)\vee$
$(x_4 - x_2 > 0 \wedge h_N - x_1 \geq 0) \vee (x_4 - x_2 > 0 \wedge h_N - x_3 \leq 0))) \vee (x_3 - x_1 < 0 \wedge x_4 - x_2 < 0\wedge$
$((x_3 - h_N \leq 0 \wedge -x_4 - h_D < 0 \wedge x_1 - h_N > 0 \wedge (1 - \frac{\lambda_N}{R_N}(x_4 - x_2))^{\lambda_D} \leq (\frac{x_1}{h_N})^{\lambda_N})\vee$
$(-x_2 - h_D \geq 0 \wedge x_3 - h_N < 0 \wedge x_1 - h_N \geq 0 \wedge (1 - \frac{\lambda_N}{R_N}(x_4 - x_2))^{\lambda_D} \leq (\frac{R_D - \lambda_D x_3}{R_D - \lambda_D h_N})^{\lambda_N}))\vee$
$(x_3 - x_1 > 0 \wedge x_4 - x_2 > 0 \wedge (-x_2 - h_D \geq 0 \wedge x_3 - h_N < 0 \wedge x_1 - h_N < 0 \wedge x_3 - x_1 \leq \frac{-R_D}{\lambda_D}(1 - (\frac{x_4}{-h_D})^{\frac{\lambda_D}{\lambda_N}}))\vee$
$(x_3 - h_N > 0 \wedge -x_4 - h_D \leq 0 \wedge x_1 - h_N \geq 0 \wedge x_3 - x_1 \leq \frac{-R_D}{\lambda_D}(1 - (\frac{x_2}{-h_D})^{\frac{\lambda_D}{\lambda_N}})))$

Since the state space is four dimensional, it is difficult to visualize the reachable sets. One visualization technique we use, is to draw projections on to three dimensional space, as in Figure 9. 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 Figure 9 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.

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 behaviour of the system can be discerned: (a) The network essentially amplifies the difference between the initial Delta protein concentrations of both cells and initial Notch concentrations between the two cells. (b) 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. The reason that such behaviour is possible, is because the kinetic parameters for protein production and decay, and the switching thresholds, may be different for both proteins. By not instantiating them, and allowing them to vary, sets of initial conditions are obtained 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 behaviour would be very interesting to test in a biological experiment.

5.3 Four Cell Delta-Notch Automaton

The largest network that has been analysed comprises four cells. The four cell Delta-Notch hybrid model has eight continuous state variables $x_1, x_2, \ldots, x_8$, and two hundred and fifty six discrete states. The variables x_1, x_3, x_5 and x_7 denote the Delta protein levels and x_2, x_4, x_6 and x_8 represent the Notch protein levels in cells 1, 2, 3 and 4, respectively (Figure 10(a)).

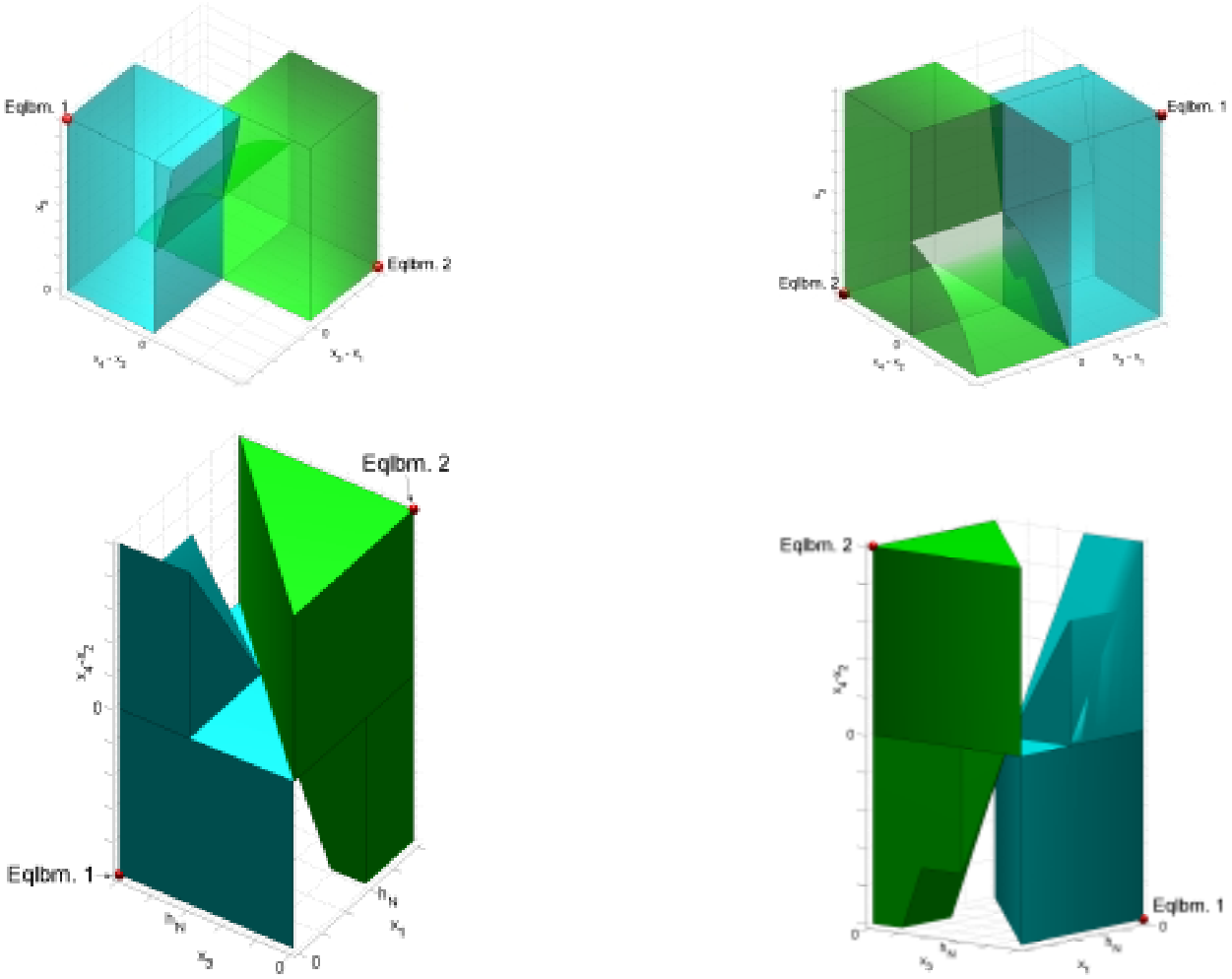


Figure 9: Projections showing computed backward reachable set from the equilibria for the two cell Delta-Notch automaton. The cyan set represents the reachable set for equilibrium 1, and the green one for equilibrium 2.

The four cell network has three possible steady states that are biologically feasible, as shown in Figure 10(b). Equilibria 1 and 2 are the cases where a single cell (cell 3 and cell 2, respectively) has a high level of Delta protein and a low level of Notch protein concentration and its three neighbours have low Delta protein and high Notch concentration. Equilibrium 3 represents a different steady state condition; two of the four cells, cells 1 and 4, have a high concentration of Delta protein and the other two cells have a low Delta and high Notch concentration.

The reachability computation for the four cell network results in the following three under-approximations of the backward reachable set from each of the three equilibria:

Equilibrium 1: $\left(x_1 = 0, x_2 = \frac{R_N}{\lambda_N}, x_3 = 0, x_4 = \frac{R_N}{\lambda_N}, x_5 = \frac{R_D}{\lambda_D}, x_6 = 0, x_7 = 0, x_8 = \frac{R_N}{\lambda_N}\right)$

$(h_D + x_6 \leq 0 \wedge h_D + x_4 \geq 0 \wedge h_D + x_2 \geq 0 \wedge h_D + x_8 \geq 0 \wedge h_N - x_7 - x_5 - x_1 \leq 0 \wedge h_N - x_5 - x_3 \leq 0 \wedge h_N - x_7 - x_3 - x_1 \geq 0 \wedge ((x_5 - x_3 > 0 \wedge x_7 - x_3 + x_1 \leq 0 \wedge h_D + x_4 \leq 0 \wedge h_N - x_7 - x_3 - x_1 > 0) \vee (x_5 - x_3 > 0 \wedge x_7 - x_3 + x_1 \leq 0 \wedge h_D + x_6 \geq 0 \wedge h_N - x_7 - x_3 - x_1 > 0) \vee (x_5 - x_3 > 0 \wedge x_7 - x_3 + x_1 \leq 0 \wedge$

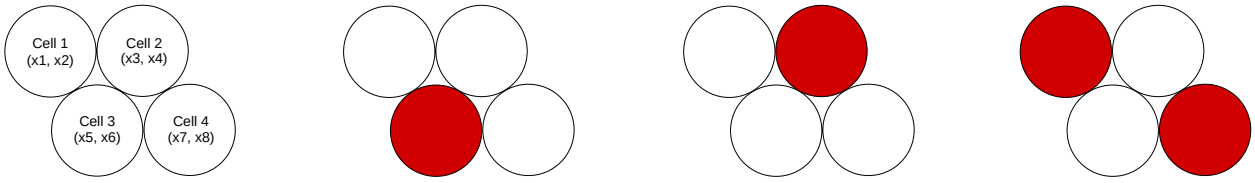


Figure 10: (a) Layout of four cell Delta-Notch network showing the variables associated with each cell. b) Biologically consistent steady states of the four cell network, a shaded cell represents a high steady state concentration of Delta protein and low level of Notch protein, and an unshaded cell has low Delta protein and high Notch protein at steady state. From left to right, equilibrium 1 has cell 3 with high Delta level and the rest with high Notch, equilibrium 2 has cell 2 with high Delta concentration and the others with high Notch, and equilibrium 3 has cells 1 and 4 with high Delta and the others with high Notch levels.

$h_N - x_7 - x_5 - x_1 \geq 0) \vee (x_5 - x_3 > 0 \wedge x_7 - x_3 + x_1 \leq 0 \wedge h_D + x_8 \leq 0 \wedge h_N - x_7 - x_3 - x_1 > 0) \vee (h_D + x_2 \leq 0 \wedge$
$h_D + x_8 > 0 \wedge h_N - x_7 - x_3 - x_1 > 0) \vee (x_7 - x_3 + x_1 \geq 0 \wedge x_7 - x_5 + x_1 < 0 \wedge h_D + x_4 \leq 0 \wedge h_N - x_7 - x_3 - x_1 > 0) \vee$
$(x_7 - x_3 + x_1 \geq 0 \wedge x_7 - x_5 + x_1 < 0 \wedge h_D + x_6 \geq 0 \wedge h_N - x_7 - x_3 - x_1 > 0) \vee (x_7 - x_3 + x_1 \geq 0 \wedge x_7 - x_5 + x_1 < 0 \wedge$
$h_N - x_5 - x_3 \geq 0) \vee (h_D + x_6 < 0 \wedge h_D + x_4 > 0 \wedge h_D + x_2 > 0 \wedge h_D + x_8 > 0 \wedge h_N - x_7 - x_3 - x_1 \leq 0) \vee$
$(x_7 - x_3 + x_1 \geq 0 \wedge x_7 - x_5 + x_1 < 0 \wedge h_D + x_8 \leq 0 \wedge h_N - x_7 - x_3 - x_1 > 0)))$

Equilibrium 2: $\left(x_1 = 0, x_2 = \frac{R_N}{\lambda_N}, x_3 = \frac{R_D}{\lambda_D}, x_4 = 0, x_5 = 0, x_6 = \frac{R_N}{\lambda_N}, x_7 = 0, x_8 = \frac{R_N}{\lambda_N}\right)$

$(h_D + x_4 \leq 0 \wedge h_D + x_6 \geq 0 \wedge h_D + x_2 \geq 0 \wedge h_D + x_8 \geq 0 \wedge h_N - x_7 - x_5 - x_1 \geq 0 \wedge h_N - x_5 - x_3 \leq 0 \wedge$
$h_N - x_7 - x_3 - x_1 \leq 0 \wedge ((x_5 - x_3 < 0 \wedge x_7 - x_5 + x_1 \leq 0 \wedge h_D + x_6 \leq 0 \wedge h_N - x_7 - x_5 - x_1 > 0) \vee (x_5 - x_3 < 0 \wedge$
$x_7 - x_5 + x_1 \leq 0 \wedge h_D + x_4 \geq 0 \wedge h_N - x_7 - x_5 - x_1 > 0) \vee (x_5 - x_3 < 0 \wedge x_7 - x_5 + x_1 \leq 0 \wedge h_N - x_7 - x_3 - x_1 \geq 0) \vee$
$(x_5 - x_3 < 0 \wedge x_7 - x_5 + x_1 \leq 0 \wedge h_D + x_8 \leq 0 \wedge h_N - x_7 - x_5 - x_1 > 0) \vee (h_D + x_4 < 0 \wedge h_D + x_2 \leq 0 \wedge h_D + x_8 > 0 \wedge$
$h_N - x_7 - x_5 - x_1 > 0) \vee (x_7 - x_5 + x_1 \geq 0 \wedge x_7 - x_3 + x_1 < 0 \wedge h_D + x_6 \leq 0 \wedge h_N - x_7 - x_5 - x_1 > 0) \vee$
$(x_7 - x_5 + x_1 \geq 0 \wedge x_7 - x_3 + x_1 < 0 \wedge h_D + x_4 \geq 0 \wedge h_N - x_7 - x_5 - x_1 > 0) \vee (x_7 - x_5 + x_1 \geq 0 \wedge x_7 - x_3 + x_1 < 0 \wedge$
$h_N - x_5 - x_3 \geq 0) \vee (h_D + x_4 < 0 \wedge h_D + x_6 > 0 \wedge h_D + x_2 > 0 \wedge h_D + x_8 > 0 \wedge h_N - x_7 - x_5 - x_1 \leq 0) \vee$
$(x_7 - x_5 + x_1 \geq 0 \wedge x_7 - x_3 + x_1 < 0 \wedge h_D + x_8 \leq 0 \wedge h_N - x_7 - x_5 - x_1 > 0)))$

Equilibrium 3: $\left(x_1 = \frac{R_D}{\lambda_D}, x_2 = 0, x_3 = 0, x_4 = \frac{R_N}{\lambda_N}, x_5 = 0, x_6 = \frac{R_N}{\lambda_N}, x_7 = \frac{R_D}{\lambda_D}, x_8 = 0\right)$

$(h_D + x_4 \geq 0 \wedge h_D + x_6 \geq 0 \wedge h_N - x_5 - x_3 \geq 0 \wedge h_N - x_7 - x_5 - x_1 \leq 0 \wedge h_N - x_7 - x_3 - x_1 \leq 0 \wedge ((h_D + x_4 > 0 \wedge$
$h_D + x_6 > 0 \wedge h_D + x_2 \geq 0 \wedge h_D + x_8 < 0) \vee (h_D + x_2 \leq 0 \wedge h_N - x_5 - x_3 > 0 \wedge h_N - x_7 - x_3 - x_1 \geq 0) \vee$
$(h_D + x_2 \leq 0 \wedge h_N - x_5 - x_3 > 0 \wedge h_N - x_7 - x_5 - x_1 \geq 0) \vee (h_D + x_4 > 0 \wedge h_D + x_6 > 0 \wedge h_D + x_2 < 0 \wedge$
$h_N - x_5 - x_3 \leq 0) \vee (x_7 - x_3 + x_1 > 0 \wedge x_7 - x_5 + x_1 > 0 \wedge h_D + x_4 \leq 0 \wedge h_D + x_8 \leq 0 \wedge h_N - x_5 - x_3 > 0) \vee$
$(x_7 - x_3 + x_1 > 0 \wedge x_7 - x_5 + x_1 > 0 \wedge h_D + x_8 = 0 \wedge h_N - x_5 - x_3 > 0) \vee (x_7 - x_3 + x_1 > 0 \wedge x_7 - x_5 + x_1 > 0 \wedge$

$h_D+x_6 \leq 0 \wedge h_D+x_8 \leq 0 \wedge h_N-x_5-x_3 > 0) \vee (h_D+x_2 \leq 0 \wedge h_D+x_8 \geq 0 \wedge h_N-x_5-x_3 > 0)))$

Biological Significance

The results from the four cell computation are more difficult to interpret because of the larger number of inequalities describing the reachable sets and the higher dimension of the state space. By studying the reachable sets, the following inferences may be drawn: (a) The backward reachable sets from equilibria 1 and 2 strongly show the amplification of initial differences property of lateral inhibition. However, initial differences alone, in Delta and Notch protein concentrations between cells 2 and 3, do not ensure convergence to those two equilibria. The Delta and Notch proteins in cells 1 and 4 must also have appropriate initial concentrations to ensure convergence. The reason behind such strict conditions is that cells 2 and 3 are each in contact with all three other cells, thus even a low Delta concentration in each of the three neighbours sum up to a strong inhibitory signal. Hence initial conditions in all the cells strongly affect the steady state Delta concentration in those two cells. (b) The third equilibrium has a larger backward reachable set, primarily because cells 1 and 4 are in contact with only two cells and therefore the external signal has a relatively milder effect on their Delta production. In some cases the initial Notch concentration in cell 1 or cell 4 do not matter at all, provided certain other conditions are met.

6 Conclusion

This paper documents an ongoing effort to utilize a mathematical modelling framework, hybrid automata theory, to model and analyse interesting biological signalling networks and deduce constraints on the symbolic parameters of the system that may be experimentally verifiable. Predictions are also made regarding the various initial conditions necessary for the model to reach one biologically observed steady state or the other. Moreover, the relation between biology and computation is symbiotic, in the sense that new mathematical analysis tools had to be developed to analyse the signalling network under study. The paper describes the development and implementation of a new partitioning algorithm, using Lie derivatives and iterative refinement by exact solutions where computable, 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 the biological model, to determine initial conditions from which specific biologically observed steady states are reached.

The clear limitation of this approach is its computational complexity in the dimension of the continuous state. In this paper, results have been shown for a four cell network (eight continuous dimensions), and it is the authors' belief that the methods in this paper will be directly extensible to problems having twenty and higher dimensions in the continuous state [9] (corresponding to ten to twelve cell networks). Such extensions will be possible, and are reasonable, if the relatively inefficient QEPCAD [19] symbolic quantifier elimination tool is replaced with a more directed algorithm, such as that in [20]. To directly analyse large cell networks (thousands of cells), though, would require combining this type of smaller network analysis with methods that would allow one to decompose a large network into smaller, interacting units. In the Delta-Notch network, it has been noted that the number of biologically feasible equilibrium patterns grows polynomially with the size of the network,

and it may be possible to reconstruct all regular equilibrium patterns in a large network of cells by composing the possible equilibrium patterns for smaller (i.e. four, nine and sixteen) cell networks. For large cell networks, interpretation of the results also becomes challenging. The authors believe that this may be addressed using a query-based technique, i.e. instead of presenting the overall reachable set, the computational result can be "queried" for truths. From this analysis, new and interesting patterns of behaviour can emerge that would have not been easy to discern for a model with instantiated parameters.

Acknowledgements
The research presented in this paper was supported by the DARPA Biocomp program (grant number DAAD19-03-1-0373 from the Department of the Army).

References

[1] HENZINGER, T. A., HO, P. H. and WONG-TOI, H.: 'Hytech: A model checker for hybrid systems', *Software Tools Tech. Transfer*, 1997, **1**, pp. 110–122

[2] PREUSSIG, J. and WONG-TOI, H.: 'A procedure for reachability analysis of rectangular automata'. Proc. Amer. Control Conf., 2000, Chicago, USA, pp. 1674–1678

[3] ASARIN, E., DANG, T. and MALER, O.: 'd/dt: A verification tool for hybrid systems'. Proc. IEEE Conf. Decision Control, 2001, Orlando, USA, pp. 2893–2898

[4] CHUTINAN, A. and KROGH, B. H.: 'Verification of infinite-state dynamic systems using approximate quotient transition systems', *IEEE Trans. Autom. Control*, 2001, **46(9)**, pp. 1401–1410

[5] BEMPORAD, A., TORRISI, F. D. and MORARI, M.: 'Optimization-based verification and stability characterization of piecewise affine and hybrid systems', in KROGH, B. and LYNCH, N. (Eds.), 'Hybrid Systems: Computation and Control' (Springer Verlag, 2000), LNCS 1790, pp. 45–59

[6] BOTCHKAREV, O. and TRIPAKIS, S.: 'Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations', in KROGH, B. and LYNCH, N. (Eds.), 'Hybrid Systems: Computation and Control' (Springer Verlag, 2000), LNCS 1790, pp. 73–88

[7] MITCHELL, I.: 'Application of level set methods to control and reachability problems in continuous and hybrid systems'. Ph. D. thesis, Stanford University, 2002

[8] MITCHELL, I. and TOMLIN, C. J.: 'Overapproximating reachable sets by Hamilton-Jacobi projections', *J. Symbolic Comput.*, 2003, **19(1-3)**, pp. 323–346

[9] KVASNICA M., GRIEDER, P., BAOTIC, M. and MORARI, M.: 'Multi-parametric toolbox (MPT)', in ALUR, R. and PAPPAS, G. J. (Eds.), 'Hybrid Systems: Computation and Control' (Springer Verlag, 2004), LNCS 2993, pp. 448–462

[10] DE JONG, H.: 'Modeling and simulation of genetic regulatory systems: A literature review', *J. Comput. Biol.*, 2002, **9(1)**, pp. 69–105

[11] KUIPERS, B. and RAMAMOORTHY, S.: 'Qualitative modelling and heterogeneous control of global systems behaviour', in TOMLIN, C. J. and GREENSTREET, M. (Eds.), 'Hybrid Systems: Computation and Control' (Springer Verlag, 2002), LNCS 2289, pp. 294–307

[12] ALUR, R., DANG, T. and IVANCIC, F.: 'Reachability analysis of hybrid systems via predicate abstraction', in TOMLIN, C. J. and GREENSTREET, M. (Eds.), 'Hybrid Systems: Computation and Control' (Springer Verlag, 2002), LNCS 2289, pp. 35–48

[13] GRAF, S. and H. SAÏDI: 'Construction of abstract state graphs with PVS'. Proc. 9th Int. Conf. Computer-Aided Verification, 1997, Haifa, Israel, LNCS 1254, pp. 72–83

[14] TIWARI, A. and KHANNA, G.: 'Series of abstractions for hybrid automata', in TOMLIN, C. J. and GREENSTREET, M. (Eds.), 'Hybrid Systems: Computation and Control' (Springer Verlag, 2002), LNCS 2289, pp. 465–478

[15] GHOSH, R., TIWARI, A. and TOMLIN, C.: 'Automated symbolic reachability analysis; with application to Delta-notch signaling automata', in MALER, O. and PNUELI, A. (Eds.), 'Hybrid Systems: Computation and Control' (Springer Verlag, 2003), LNCS 2623, pp. 233–248

[16] ALUR, R., HENZINGER, T., LAFFERRIERE, G. and PAPPAS, G. J.: 'Discrete abstractions of hybrid systems', *Proc. IEEE*, 2000, **88(7)**, pp. 971–984

[17] MARNELLOS, G., DEBLANDRE, G. A., MJOLSNESS, E. and KINTNER, C.: 'Delta-notch lateral inhibitory patterning in the emergence of ciliated cells in *Xenopus*: Experimental observations and a gene network model'. Pacific Symp. Biocomp., 2000, Oahu, Hawaii, pp. 326–337

[18] GHOSH, R. and TOMLIN, C. J.: 'Lateral inhibition through Delta-notch signaling: A piecewise affine hybrid model', in BENEDETTO, M. D. D. and SANGIOVANNI-VINCENTELLI, A. (Eds.), 'Hybrid Systems: Computation and Control' (Springer Verlag, 2001), LNCS 2034, pp. 232–246

[19] HONG, H.: 'An improvement of the projection operator in cylindrical algebraic decomposition'. Proc. Int. Symp. Symbolic Algebraic Comp., 1990, Tokyo, Japan, pp. 261–264

[20] CANNY, J.: 'Computing roadmaps of general semi-algebraic sets', *Comput. J.*, 1993, **36(5)**, pp. 504–514

A Implementation

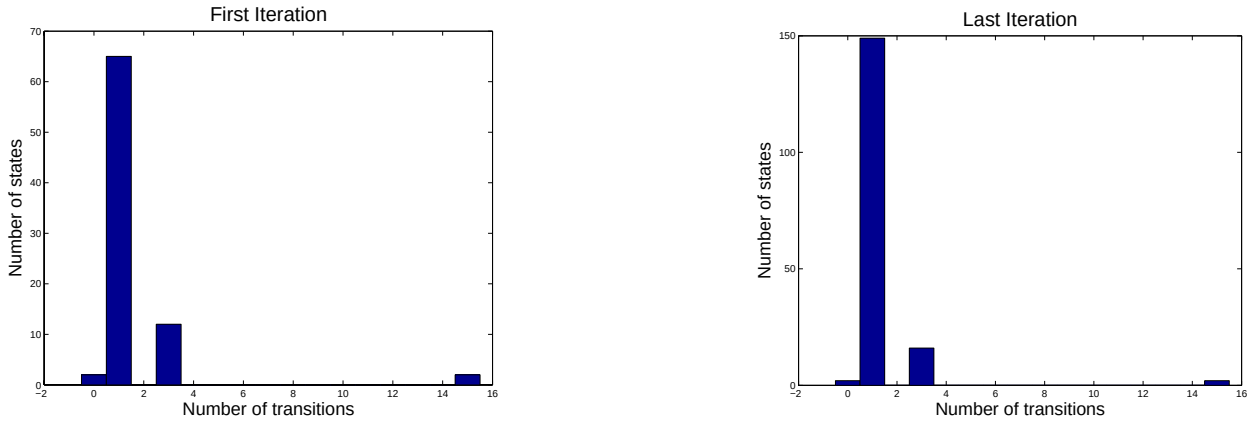
The model abstraction software has been implemented in MATLAB and the symbolic and string manipulations are also done in MATLAB. Each state of the current partition iteration is stored in a data structure that stores the signs of the invariant polynomials defining the state, the continuous vector flow 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[19]. To reduce computational load, MATLAB and QEPCAD run on separate 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. Otherwise, the state cannot be partitioned farther, and is stored as is. 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,
.
.
.
.
.

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

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

Inv: -x2-hD >= 0 /\ x3-hN >= 0 /\ -x4-hD >= 0 /\ x1-hN >= 0
x1dot = RD-lD*x1
x2dot = RN-lN*x2
x3dot = RD-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 Figure 11, the number of states with multiple transitions rises slightly, however, as a

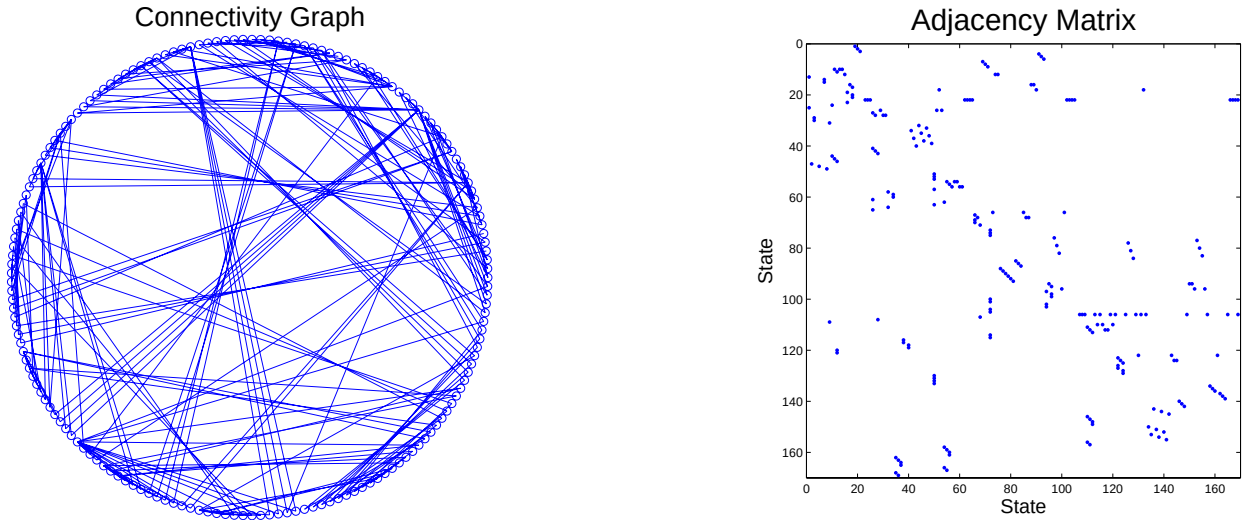
percentage of the total number of states, it drops from around 18 per cent to around 10 per cent.



(a) (b)

Figure 11: Histograms showing number of states with a certain 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 Figure 12(a), at some iteration stage. The adjacency matrix, displayed in Figure 12(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 show up as rows with dots in multiple columns in Figure 12(b), and further refinement may not be possible.



(a) (b)

Figure 12: (a)The connectivity network graph of the discrete abstraction, each circle on the circumference 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.