Symbolic Approaches for

Verification of Hybrid Systems

Ashish Tiwari

Tiwari@csl.sri.com

Computer Science Laboratory

SRI International

Menlo Park CA 94025

http://www.csl.sri.com/~tiwari

Outline

1. Hybrid Systems: Introduction and motivation
2. High level overview of our approach
3. Details
   - HybridAbstraction
     - intelligent seed forms, composition, locality, language
   - Model Checking
   - Decision Procedures
4. Examples
5. Conclusion

The World: Then...

- 20th century modelling of the physical world by physicists, chemists, biologists, engineers, etc:
  - Continuous state variable
  - Updated according to differential equations

$$m\ddot{x} = F - k\dot{x}$$

- Advent of computational machines
  - Discrete variables
  - Updated according to specific rules

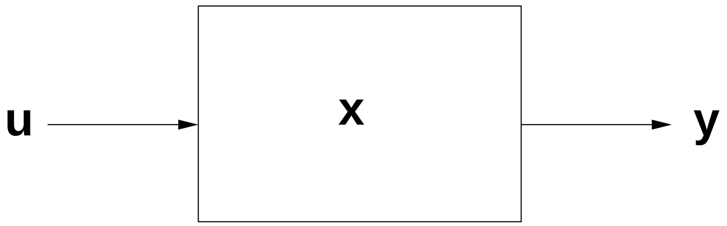
The World: Now...

- Machines interacting with the physical world
  - embedded systems
  - controllers for chemical plants, airplanes
- Or, maybe a new approach to model the same old continuous world
  - differential equations not the correct modelling language for certain phenomenon?
  - some state variables are better viewed discrete

Formal Models: Legacy...

- Continuous dynamical systems
  - The state space formulation

$$
\begin{aligned}
\dot{x}(t) &= f(x(t), u(t), t) \\
y(t) &= h(x(t), t)
\end{aligned}
$$



- Finite state automata
  - Mealy/Moore machines
  - state transition systems

Formal Models: Now...

Putting the two formal models together, Hybrid Automata:

- Embed a continuous dynamical system inside each state
- World now evolves in two different ways
  - Move from one state to another via a discrete transition
  - Remain in the state and let the continuous world evolve
- System has different modes of operation, while some discrete logic performs mode switches

Analysis

What can we say (deduce, compute) about the model?

- Reachability. Is there a way to get from point A to point B
- Safety. Does the system stay out of a bad region
  - Can the car ever collide with the car in front?
  - Can the gear box get stuck in the second gear?
- Liveness. Does something good always happen

Does the model satisfy some property.

Property is described in a logic interpreted over the formal models.

Problem

- Given a hybrid automata
- And a safety property
- Show that the property is true of the model

Some Approaches for Analysis of Hybrid Systems

- Simulation and testing
- Decidability Results: Reachability algorithms for subclasses of hybrid automata
  - Timed Automata
  - Initialized Multirate Automata
  - Initialized Rectangular Automata
  - Linear Vector Fields specified using matrix A, where A is either nilpotent, diagonalizable with rational eigenvalues, or has only imaginary eigenvalues
  - O-minimal automata
- Abstraction Techniques: Compute approximations and work on them

Abstraction: The General Idea

1. Partition the state space: each member of the partition is a state of a new system
   - defines an abstraction map from concrete states onto a smaller set of "abstract states"
   - one abstract state corresponds to a set of concrete states
2. Quotient the original system modulo the partition:
   - there is a transition from one abstract state to another if there is a possible concrete one between the corresponding sets
   - the abstract system is a conservative approximation of the original system
3. Analyze the abstract system

Abstraction: Example

- Original State Machine: Let x be of type int

$$\begin{array}{lcl} \text{Initially} & : & x = 0 \\ \text{Program} & : & x > 0 \longrightarrow x' = x + 2 \\ \text{Behavior} & : & x = 0, 2, 4, \ldots \end{array}$$

- Abstract Finite State Machine: Let $evenx$ be of type $boolean$

  Mapping $evenx \mapsto \exists y . x = 2y$

$$\begin{array}{lcl} \text{Initially} & : & evenx = True \\ \text{Program} & : & True \longrightarrow evenx' = evenx \\ \text{Behavior} & : & evenx = True, True, \ldots \end{array}$$

What is required to create an abstraction

For discrete state transition systems:

- An abstraction mapping
- Technique for abstracting transitions
  - Is there an abstract transition from state $evenx = True$ to the state $evenx = False$
  - Is there a concrete transition from some state in $\{x = 0, x = 2, x = 4, \ldots\}$ to a state in the set $\{x = 1, x = 3, x = 5, \ldots\}$
  - Is the following formula satisfiable in the theory of integers

$$\exists x.(\exists y.x = 2y) \ \wedge \ \neg(\exists z.x + 2 = 2z)$$

Hybrid Abstraction

For hybrid systems:

- What abstraction mapping?
- What theorem proving obligations, which theory?
- How to abstract continuous dynamics?

Hybrid Automata

A tuple $(Q, X, \mathbf{S}_0, F, Inv, R)$:

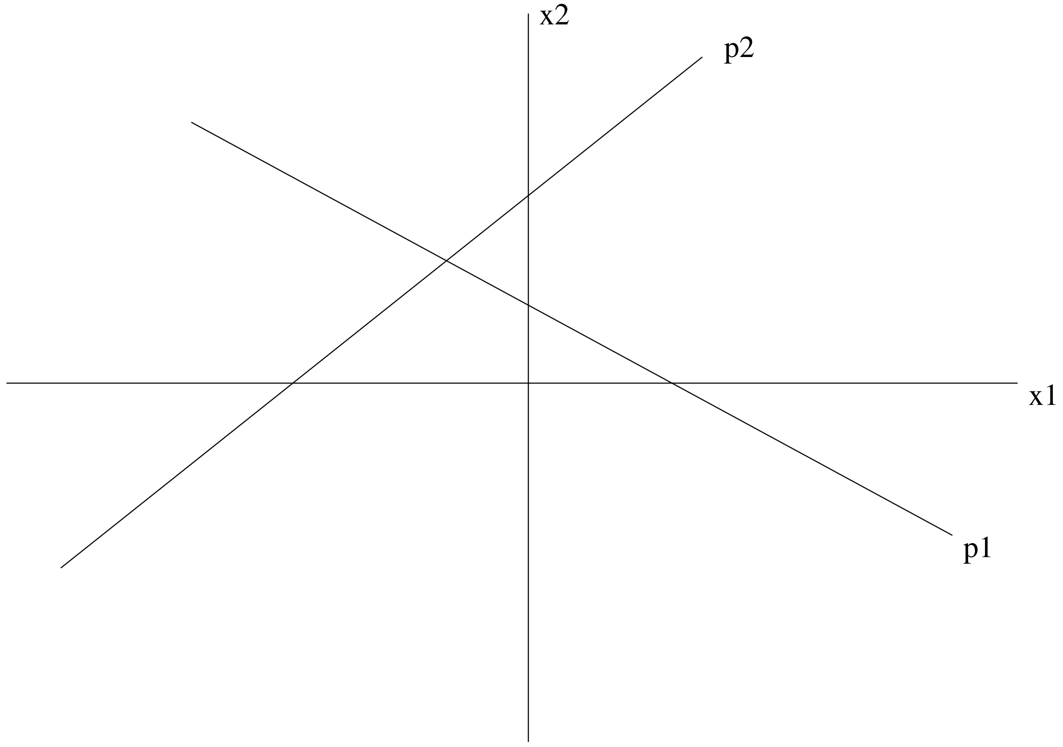
- Q: finite set of discrete variables
- X: finite set of continuous variables
- $\mathbf{X} = \mathbb{R}^{|X|}$, $\mathbf{Q}$ = set of all valuations for Q
- $\mathbf{S} = \mathbf{Q} \times \mathbf{X}$
- $\mathbf{S}_0 \subseteq \mathbf{S}$ is the set of initial states
- $F : \mathbf{Q} \mapsto (\mathbf{X} \mapsto \mathbb{R}^{|X|})$ specifies the rate of flow, $\dot{x} = F(q)(x)$
- $Inv : \mathbf{Q} \mapsto 2^{\mathbb{R}^{|X|}}$ gives the invariant set
- $R \subseteq \mathbf{Q} \times 2^{\mathbf{X}} \mapsto \mathbf{Q} \times 2^{\mathbf{X}}$ captures discontinuous state changes

Partitioning the State Space

- The discrete component, $\mathbf{Q}$, of the state space, $\mathbf{Q} \times \mathbf{X}$, is abstracted independently
  - Assume it is not abstracted
- The continuous component, $\mathbf{X}$, of the state space is partitioned based on the signs of finite number of functions over the continuous state variables
  - Let $f_1, f_2, \ldots, f_m$ be functions from $\mathbf{X}$ to $\mathbb{R}$
  - Let $q_{f1}, q_{f2}, \ldots, q_{fm}$ be m corresponding discrete variables interpreted over $\{pos, neg, zero\}$ s.t.
    - $\star$ $q_{f1} = pos$ iff $f_1 > 0$
    - $\star$ $q_{f1} = neg$ iff $f_1 < 0$
    - $\star$ $q_{f1} = zero$ iff $f_1 = 0$

Abstraction Algorithm: 1

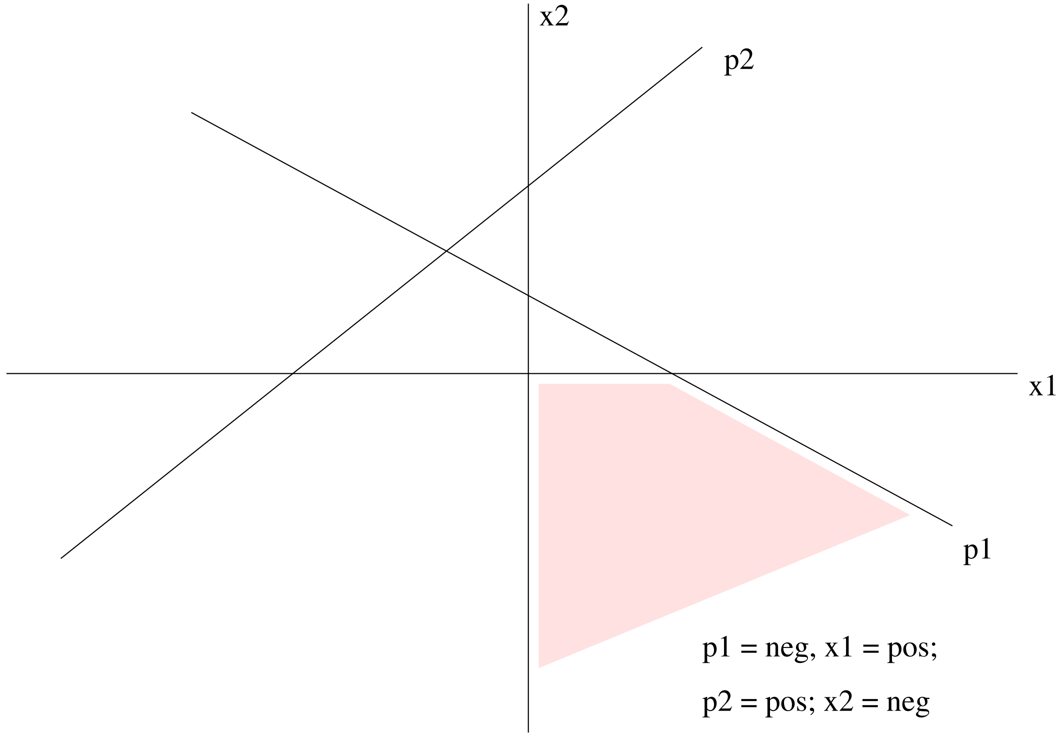
Consider a continuous dynamical system with two state variables. Concrete state space: $\mathbb{R}^2$



Partitioned w.r.t signs of four linear forms x_1, x_2, p_1, and p_2.

Abstraction Algorithm: 2

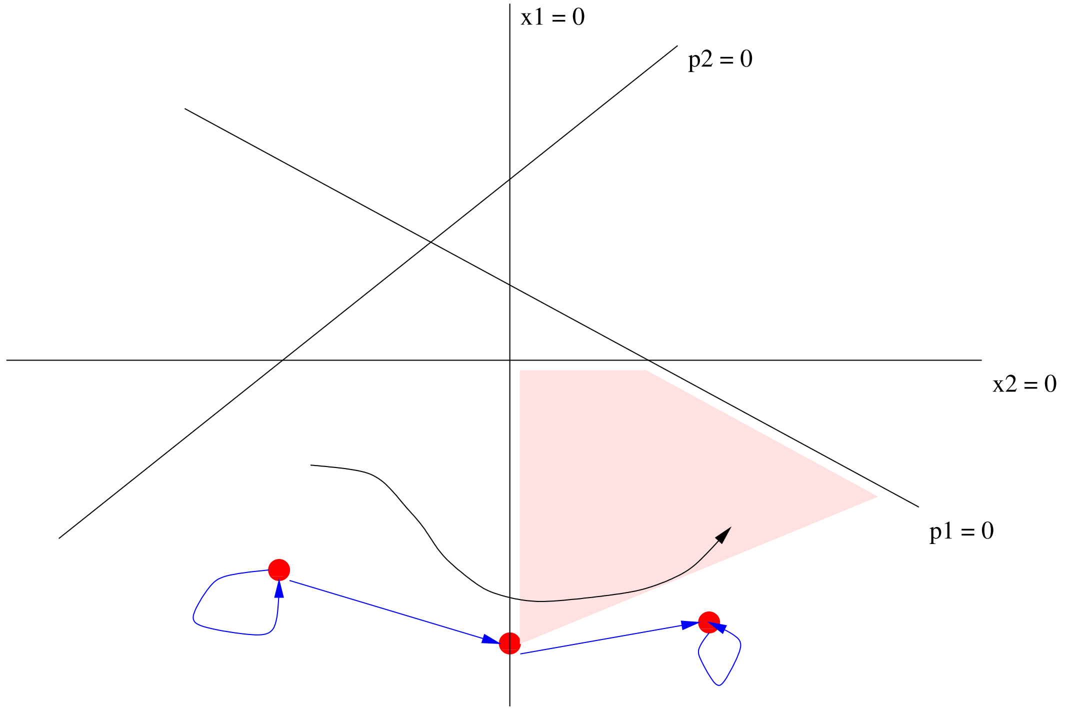
Abstract states correspond to subsets of concrete states.



Total number of abstract states = $3^4 = 81$, but feasible abstract states = $11 + 16 + 6 = 33$

Abstraction Algorithm: 3

Abstract transitions overapproximate concrete transitions.



How to abstract the continuous transitions?.

Other Approaches

Partition	Transition Computation
Polytopes	Flowpipes
Ellipsoids	Numerical techniques
Level sets	Numerical techniques
(Timed) Regions	Simple quantitative reasoning

Different representations for sets of states

But, numerical methods to map the continuous evolutions onto the represented state set

Sound for some nicely behaved subclasses

Abstracting the Continuous Dynamics

Concrete state space	:	$\mathbf{Q} \times \mathbb{R}^n$
Abstract state space	:	$\mathbf{Q} \times 3^m$

Our approach is based on qualitative reasoning

- For a given mode (continuous dynamics), for each f_i

$$\begin{array}{lcl} f_i > 0 \ \wedge \ \dot{f}_i \geq 0 & \longrightarrow & f_i' > 0 \\ f_i > 0 \ \wedge \ \dot{f}_i < 0 & \longrightarrow & f_i' > 0 \ \vee \ f_i' = 0 \end{array}$$

- Do this for all possible modes in the system to get abstractions of the continuous evolutions in all modes

But how do I know the sign of $\dot{f}_1$

Choosing Predicates for Abstraction

- If $\dot{f}_i \in \{f_1, f_2, \ldots, f_m\}$, then we know the sign of $\dot{f}_i$
  - Explicitly add derivatives of f_i's to the set
- If not, we could still deduce the sign of $\dot{f}_i$ from
  - the signs of $\{f_1, f_2, \ldots, f_m\}$
  - state invariants

  More theorem proving proof obligations!
- Add those functions to the set whose sign can be "easily" inferred
  - For linear dynamical systems, forms corresponding to eigenvectors of transpose

Choosing Partition Forms

Linear system $\dot{\vec{x}} = A\vec{x}$

Assume linear form $f = \vec{r}^T\vec{x}$

Question: For what $\vec{r}$ is this “good”?

We want $\dot{f} = kf$

i.e., $\dot{\vec{r}^T}\vec{x} = k\vec{r}^T\vec{x}$

i.e., $\vec{r}^T A\vec{x} = k\vec{r}^T\vec{x}$

i.e., $\vec{r}$ is an eigenvector of A^T corresponding to eigenvalue k

We want k to be real

For such f, qualitative abstraction of cont dynamics is easy

Choosing Partition Forms

Linear system $\dot{\vec{x}} = A\vec{x}$

Assume linear form $f = \vec{r}^T \vec{x}$

Question: For what $\vec{r}$ is this "good"?

We want $\ddot{f} = b\dot{f} + cf$

i.e., $\vec{r}^T \ddot{\vec{x}} = b\vec{r}^T \dot{\vec{x}} + c\vec{r}^T \vec{x}$

i.e., $\vec{r}^T A^2 \vec{x} = b\vec{r}^T A\vec{x} + c\vec{r}^T \vec{x}$

i.e., $\vec{r}$ is in the null space of $(A^T)^2 - bA^T - cI$

Generalize to a primary decomposition for A^T!

For such f, sign of $\ddot{f}$ can be inferred from the signs of f and $\dot{f}$, at least in some cases

Choosing Partition Forms: Primary Decomposition

Say $ch(A) = \Pi_i(a_i\lambda^2 + b_i\lambda + c_i)$, $a_i, b_i, c_i \in \mathbb{R}$

Let $\vec{r} \in Null(a_i(A^T)^2 + b_i A^T + c_i I)$

Then, $f = \vec{r}^T\vec{x}$ is a "good" linear form for abstraction

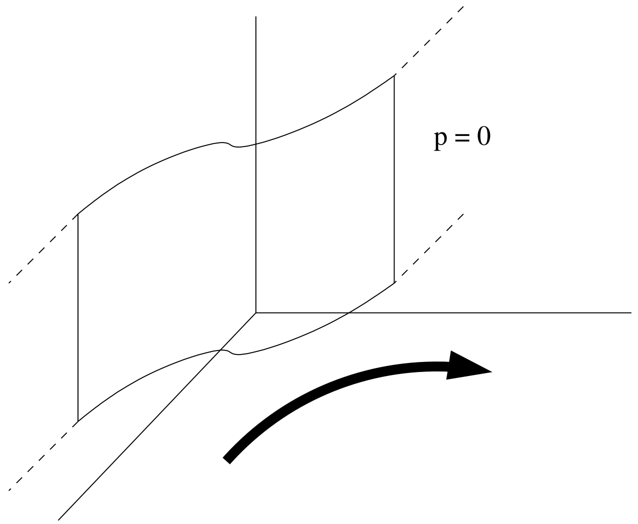
What does "goodness" translate to?

- If $a_i = 0$, then $f = f_0 e^{\frac{-c_i}{b_i}t}$
- If $b_i = 0$, then f oscillates between a min and max value
- If $b_i < 0$, then f oscillates while exponentially decaying, still bounded

Illustration

Let $[r_1, r_2, r_3]$ be an eigenvector corresponding to a negative real eigenvalue of a 3 dimensional linear system.

Let $p = r_1 x_1 + r_2 x_2 + r_3 x_3$.



- If initial states are all on one side, then the systems continues to remain on the same side, i.e., always $p \geq 0$ or always $p \leq 0$
- At, say, $x_2 = 1$ and $x_3 = 2$, the value of x_1 can be bounded

Consequences

- Can do non-trivial reachability computation for linear systems with mixed eigenvalues
  - existing decidability results can not handle this class of systems
  - we extract as much information from the system as available, and bridge the gap between the decidable and undecidable problems
  - incrementally getting to the decidability result: the more eigenvalues are negative reals, the better reachability sets you generate
  - In the limit, do we get back the decidability results?

Observations

Additional interesting consequence:

- Do not need to explicitly compute the real eigenvalues or the eigenvector $\vec{r}$
  - the eigenvalue and eigenvector are easily seen to be algebraic
  - symbolic decision procedures for real closed fields can handle the algebraic expressions representing these eigenvalues
- We assumed linear forms and linear dynamics
- Can we get useful partition forms for nonlinear dynamics using nonlinear forms?

Hybrid Abstraction Summary

Consists of three steps:

- Seed: Start with a set $\{f_1, \ldots, f_m\}$ of forms of interest
  - Expressions specifying the initial region $\mathbf{S_0}$, transition conditions (specification of $\mathbf{R}$), resets (specification of $\mathbf{R}$)
- Saturate: Close the above set under derivative computation
  - For each form f_i, and each location $l \in \mathbf{Q}$, compute $\dot{f}$, given $\dot{X} = F(l)(X)$
  - Add $\dot{f}_i$ to the seed set and recurse
  - Add "interesting" forms computed by other methods
- Add Transitions:
  discrete transitions abstracted using "usual" predicate abstraction
  continuous transitions abstracted qualitatively

Abstracting Discrete Transitions

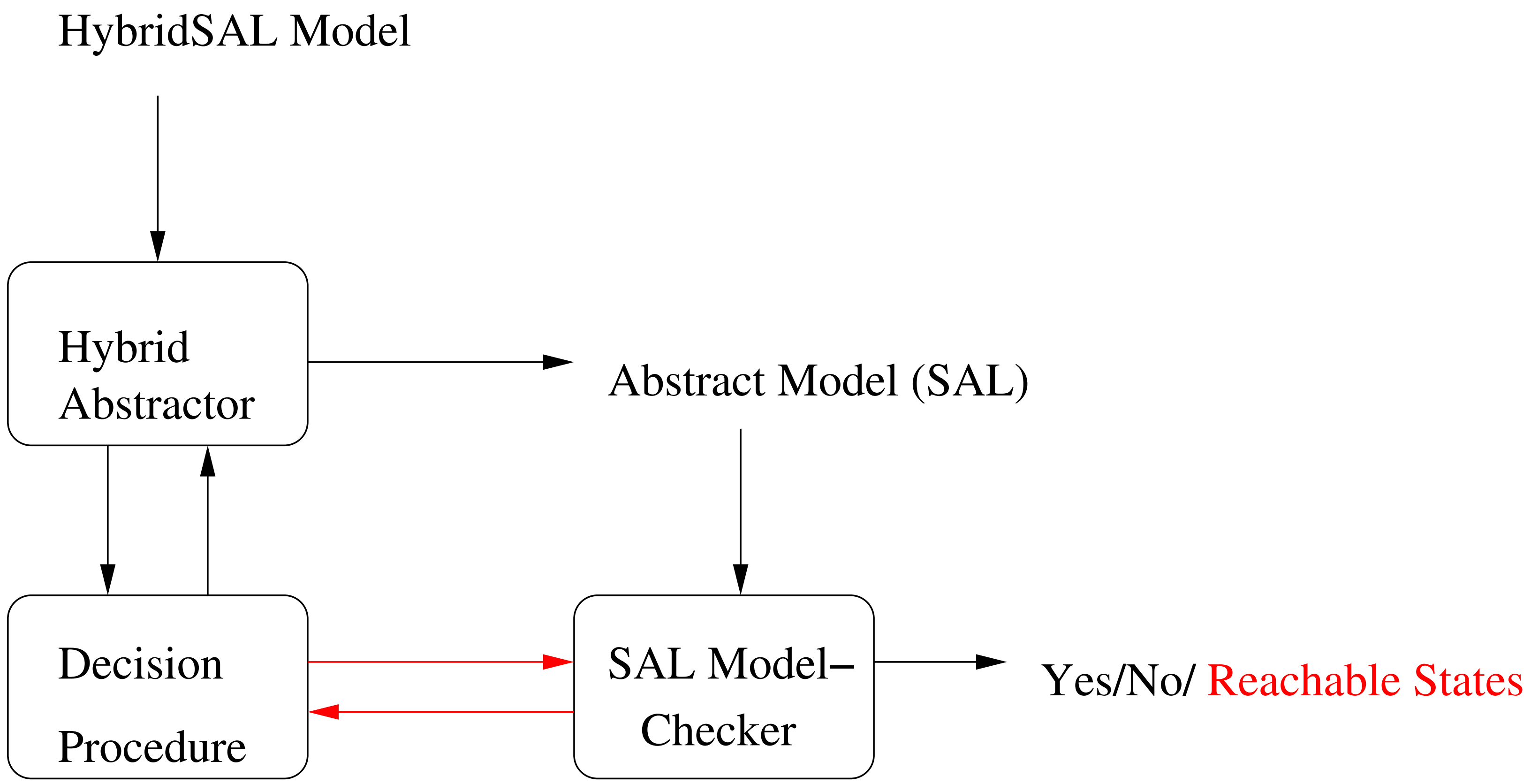
Discrete Transition: $(q, \psi(X), q', New(X))$, where

- q, q': modes,
- $\psi(X)$: enabling condition, and
- $New(X)$: assignments to continuous variables.

Abstract Discrete Transition: $((q, \phi_1), (q', \phi_2))$ if

- The formula $\phi_1 \wedge \psi$ is satisfiable and
- The formula $\exists X^o : \psi(X^o) \wedge X = New(X^o) \wedge \phi_2(X)$ is satisfiable.

Tool Architecture



Discharging Proof Obligations

- If specification uses polynomials and all forms f are polynomials, then we can use a decision procedure for the theory of real closed fields
- QEPCAD is one such decision procedure implementation
- But, abstractor generates thousands of proof obligations
- QEPCAD not robust, fast, and logical enough
- We use a home made restricted chaining calculi based decision procedure, which is sound but incomplete
- Because HybridAbstraction makes a failure-tolerant use of the decision procedure

This gives an automated abstraction algorithm for polynomial hybrid automata

Which can be extended by extending the decision procedure

Features of the Abstraction Algorithm

- *Theorem*[Correctness] The abstractions constructed by the algorithm are sound with respect to the hybrid automata semantics.
- *Theorem*[Relative Completeness] Let ϕ be a QF fmla over X (in $\Re$) that represents the set of reachable states and $P = Poly(\phi)$. Let ψ be the reachable set computed by the algorithm with seed P. If the saturation process terminates, then $\psi = \phi^c$.

  Implications to linear systems:

  - nilpotent
  - all real eigenvalues

Example: Hybrid System

Consider a simple thermostat controller with:

- Discrete modes: Two modes, $q = on$ and $q = \mathit{off}$
- Continuous variable: The temperature x
- Initial State: $q = \mathit{off}$ and $x = 75$
- Discrete Transitions:

$$\begin{array}{rcl} q = \mathit{off} \;\; and \;\; x \leq 70 & \longrightarrow & q' = on \\ q = on \;\; and \;\; x \geq 80 & \longrightarrow & q' = \mathit{off} \end{array}$$

- Continuous Flow:

$$\begin{array}{rcl} q = \mathit{off} \;\; and \;\; x > 68 & \longrightarrow & \dot{x} = -Kx \\ q = on \;\; and \;\; x < 82 & \longrightarrow & \dot{x} = K(h - x) \end{array}$$

Abstraction Algorithm

Consists of three steps:

- Seed: Start with a set P_0 of polynomials of interest

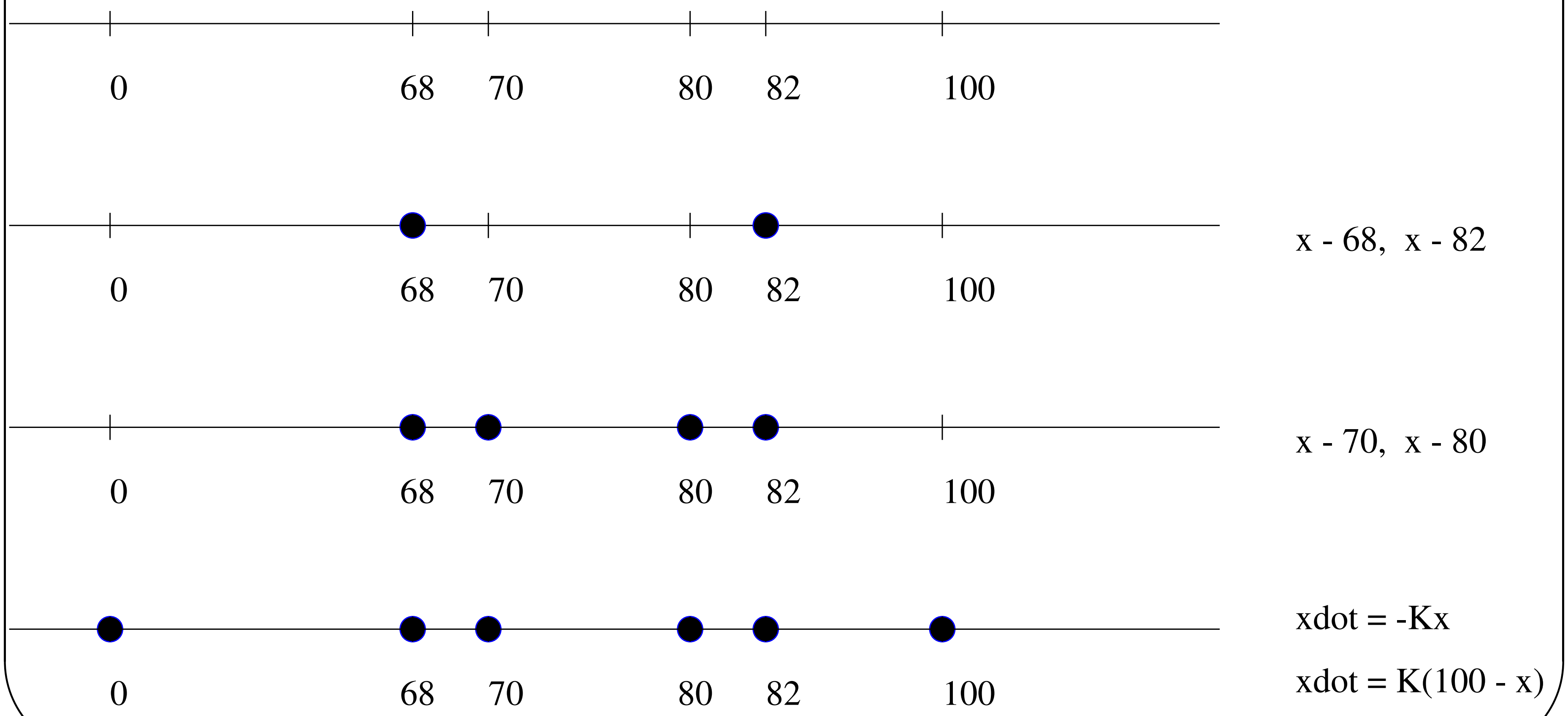
$$P_0 = \{x - 68, x - 82\}$$

- Saturate:

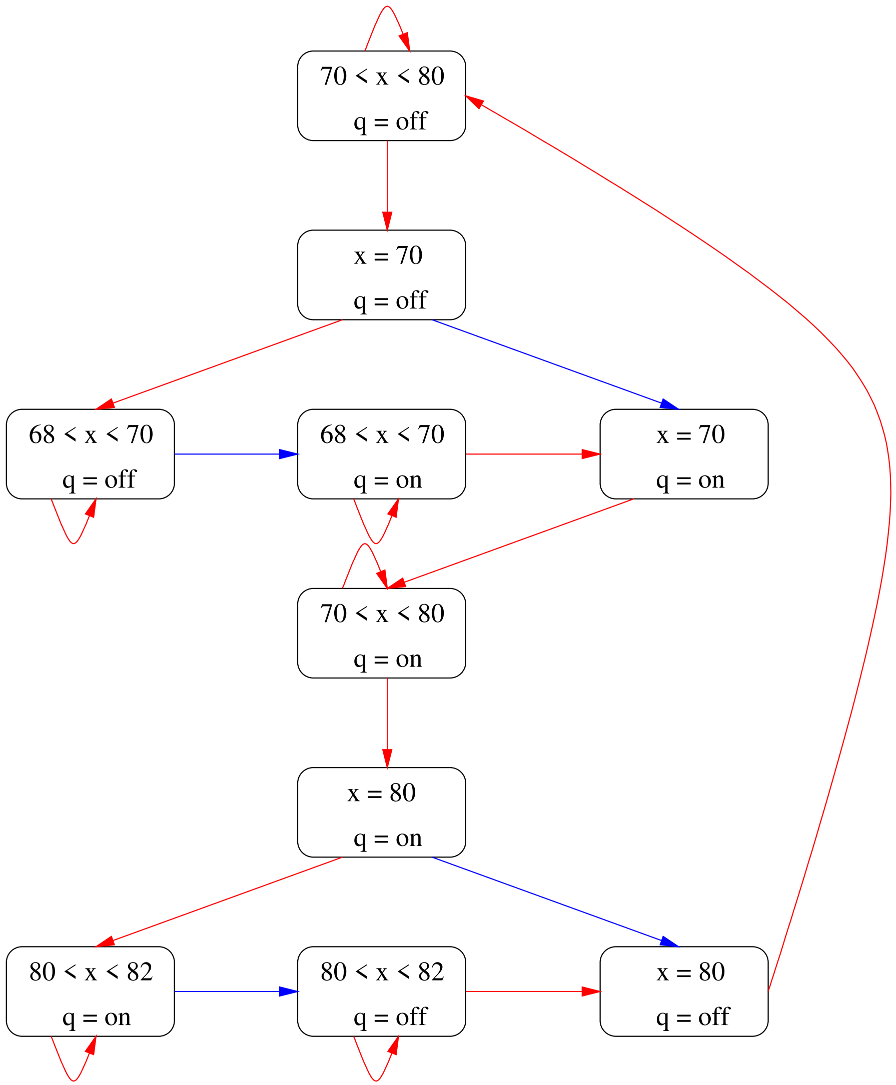
$$\begin{aligned} P_1 &= \{x - 68, x - 82, x - 70, x - 80\} \\ P_2 &= P_1 \cup \{-Kx, K(h - x)\} \\ P_3 &= P_1 \cup \{x, h - x\} \end{aligned}$$

- Abstract

Generating New Polynomials



Abstract Thermostat System



Example: Cruise Control

Example. Consider a cruise control:

$$
\begin{aligned}
\dot{v} &= a \\
\dot{a} &= -4v + 3vf - 3a + gap \\
g\dot{a}p &= -v + vf
\end{aligned}
$$

where v, a is the velocity and acceleration of this car, $v\,f$ is the velocity of car in front, and gap is the distance between the two cars.

Cruise Control: Example Contd

The transpose matrix A^T has one negative real eigenvalue λ.

If $\vec{r} = [r1, r2, r3, r4]^T$ is the eigenvector corresponding to λ, then consider the polynomial

$$p = r1 * v + r2 * vf + r3 * a + r4 * gap$$

Why is this special?

$$\begin{aligned}
\frac{dp}{dt} &= \frac{d}{dt}([v, vf, a, gap]\vec{r}) \\
&= (A[v, vf, a, gap])^T \vec{r} \\
&= [v, vf, a, gap] A^T \vec{r} \\
&= [v, vf, a, gap] \lambda \vec{r} = \lambda p
\end{aligned}$$

For negative λ, the polynomial p is monotonically exponentially decaying function over all trajectories of the dynamical system specified by A.

B. Subtilis Sporulation Network

Life Cycle of a B.Sub

1. Vegetative cell (2, 3)
2. cell division (1)
3. Stationary (4)
4. $T0$ (5)
5. Spore (1)

When faced with stress, B.Sub switches to the sporulation cycle

Two Component Systems

Most bacteria/living cells exhibit a sensor/regulator two-component system:

1. Transmembrance sensor kinase protein: an environmental change causes a conformational change in it.

2. Cytoplasmic regulator protein: The kinase activity phosphorylates the regulator, which promotes transcription of new products from a specific site on the genome.

Examples: osmoregulation, chemotaxis, sporulation.

Sporulation is more complex than most others.

Sporulation

Basic control mechanisms: provide phosphate, or strip phosphate

Example of a “component”: SinI-SinR global bistable switch

- Tetrameric SinR represses Spo0A
- SinI inactivates SinR repression by binding to it
- SinI expression is controlled by Spo0A-P, SinR, Hpr, and AbrB

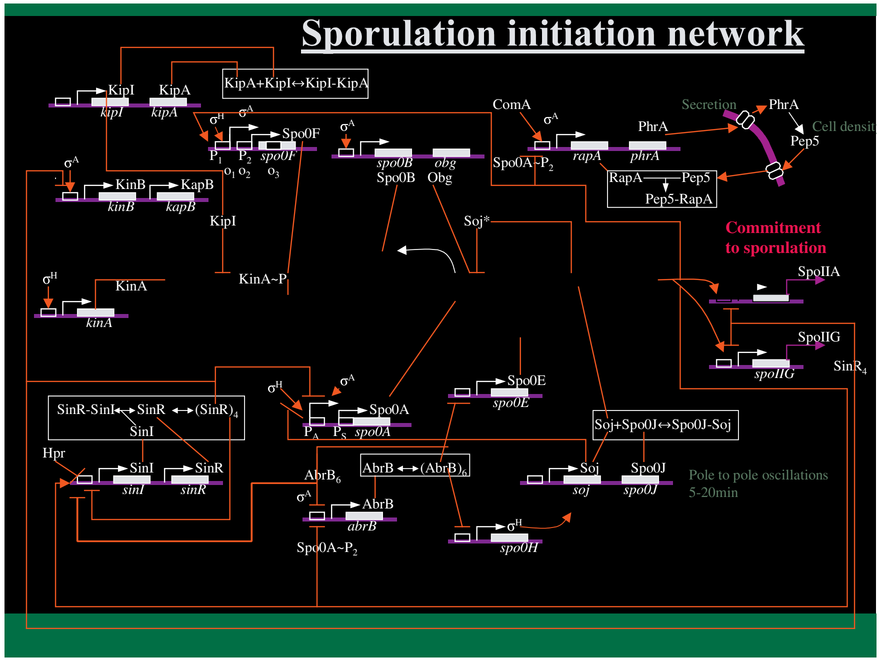
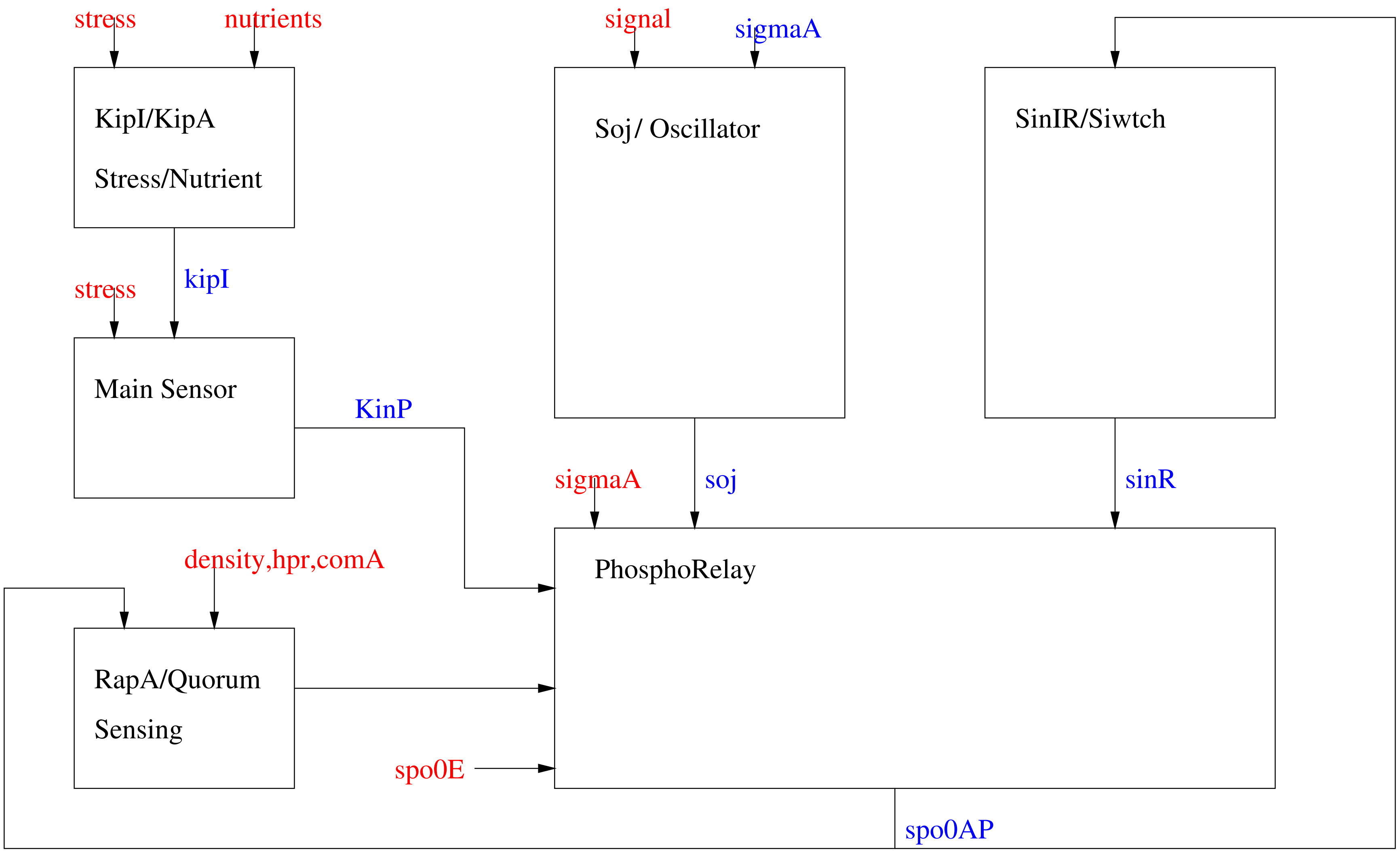


Figure 1: The sporulation inititation network in *B. subtilis*.

In The New World...



Future Work / Other topics

Composition—

- continuous state space partitioned differently in different modes
- each mode informs the other what it expects to know

Model checker: interfaces with the decision procedure for model checking

lazy abstractions

Other interesting "seed" forms—

- symbolically integrate forms
- nonlinear systems

Summary

- Completely automatic procedure for hybrid automata specified only using polynomials: uses a combination of theorem proving and decision procedure for real-closed fields.
- Abstractions can be refined by either adding new polynomials or higher-order derivatives.
- Discrete transitions are abstracted using well-known data abstraction technique.
- Continuous flows are qualitatively abstracted using arbitrary polynomials over the state variables and their higher-order derivatives.
- The method also applies to more general systems, say with trignometric and exponential functions, if the decision procedure can handle them