On Continuous, Discrete and Timed Models in Systems Biology

- Adding Time to Discrete Models of Genetic Networks
- Approximating Continuous Systems by Timed Automata

Oded Maler

CNRS - VERIMAG
Grenoble, France

2008

Joint Work with Gregory Batt, INRIA

Systems Biology

- Systems Biology: the new gold rush for many mathematical and technical disciplines
- Biophysics, Biomatics, Bioinformatics, Biostatistics...
- The generic template:
  - I do X (for my pleasure, because I studied it, that's what I know) but... X can also be useful for Biology
- So Here I am, presenting my own X, a certain species of Informatics / Computer Science
- My X is based on **automata as dynamical systems** with excursions into **hybrid** dynamics (automata plus differential equations) and **timed** dynamics (automata plus quantitative time) as an intermediate level of abstraction
- When you have a hammer, everything looks like a nail

Summary

- Informatics is more than technology, it is also a kind of mathematics/physics
- Automata as dynamical systems
- Verification illustrated
- Between the discrete and the continuous
- Timed models and their applications
  - Adding time to discrete models of genetic regulatory networks
  - From continuous to timed systems (the technical contribution of the paper in the proceedings)
- Back to the big picture

Computer Technology

- Computers, Networks, Operating Systems, Data-Bases, Web, Search Engines, Graphics,
- Embedded Systems, Sensors, Programming Languages, Word Processing, Computer Control, Robotics, Security ..
- Influence on all domains of human activity, including Biology:
- String Processing for DNA, Statistical Computations, Simulation, Animation
- Date-Bases, Micro-Arrays, Ontologies and Description Languages
- Communication and Data Sharing, Lab Management

In all those activities the computer is a useful **material tool** in the **service** of others

A More Noble Role, Perhaps

- Biology seems to be trying to go through a kind of Newtonian revolution
- The essence of such revolution is to upgrade (as much as possible) descriptive "models" by **dynamic models** with stronger predictive power and refutability
- Classical models of dynamical systems are clearly not sufficient for effective modeling of biological phenomena
- Models, insights and computer-based analysis tools developed within Informatics can help

What Is Informatics ?

- Among other things informatics is: the (pure and applied) study of **discrete-event dynamical systems** (automata, transition systems)
- A natural point of view for the "reactive systems" parts of informatics (hardware, protocols, real-time, stream processing)
- Especially for people working on modeling and verification of such systems
- Sometimes obscured (intentionally or not) by fancy formalisms: Petri nets, process algebras, rewriting systems or temporal logics..
- All honorable topics with intrinsic importance, beauty, etc.
- But sometimes should be distilled to their **essence** in order to make sense for potential users from other disciplines (rather than frighten/impress them)

Dynamical System Models in General

- State variables whose set of valuations determine the state space
- Time domain along which these values evolve
- Dynamic law which says how state variables evolve over time, possibly under the influence of external factors
- System behaviors are progressions of states in time
- Having such a model, knowing an initial state $x(0)$ one can predict, to some extent, the value of $x(t)$
-
- Remark: Variables in Biology can be of various natures and granularities (concentrations, states of individual molecules, stages in processes, etc.)

Classical Dynamical Systems

- State variables: real numbers (location, velocity, energy, voltage, concentration)
- Time domain: the real time axis $\mathbb{R}$ or a discretization of it
- Dynamic law: differential equations

$$\dot{x} = f(x, u)$$

or their discrete-time approximations

$$x(t+1) = f(x(t), u(t))$$

- Behaviors: trajectories in the continuous state space
- Achievements: Apples, Stars, Missiles, Electricity, Heat, Chemical processes
- Theorems, Papers, Simulation tools

Automata as Dynamical Systems

- Abstract discrete state space, state variables need not have a numerical meaning
- Logical time domain defined by the events (order but not metric)
- Dynamics defined by transition tables: input event **a** takes the system from state **s** to state $\mathbf{s}'$
- Behaviors are sequences of states and events
- Composition of large systems from small ones, hierarchical structuring
- Different modes of interaction: synchronous/asynchronous, state-based/event-based
- Sometime additional syntax may be required

Automata can Model many Phenomena and Devices

- Software, hardware,
- ATMs, user interfaces
- Administrative procedures
- Communication protocols
- Cooking recipes, Manufacturing instructions
- Any process that can be viewed as a sequence of steps

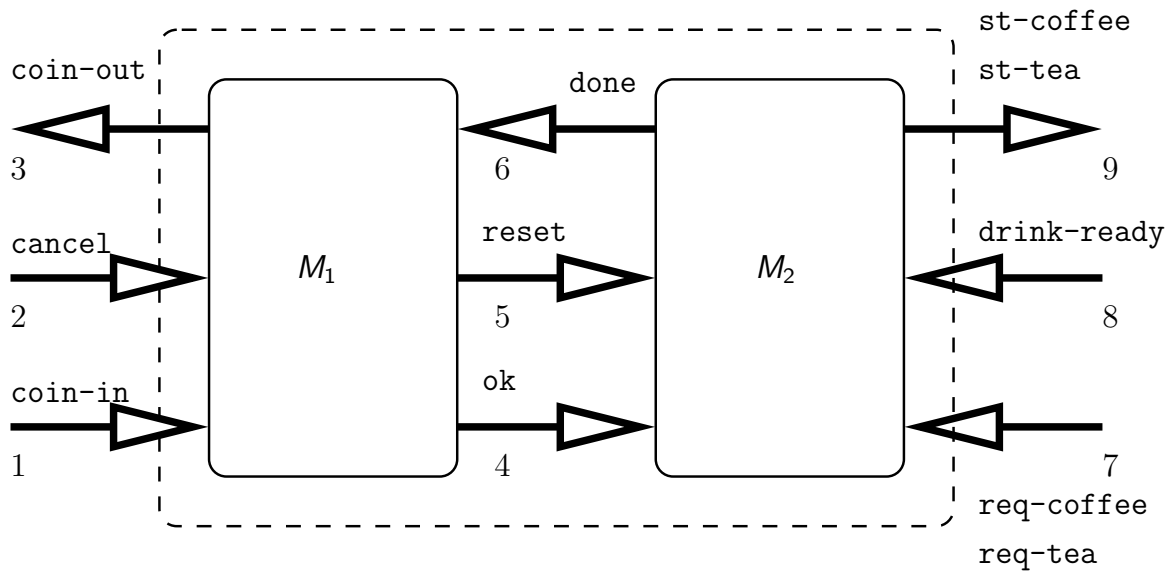
- But what can we do with these models?
- There are no analytical tools as in continuous systems
- We can simulate and sometimes do formal verification

What is Verification ?

- Given a complex discrete dynamical system with some uncontrolled inputs or unknown parameters
- Check whether ALL its behaviors satisfy some properties
- Properties:
  - Never reach some part of the state space
  - Always come eventually to some (equilibrium) state
  - Never exhibit some pattern of behavior
  - Quantitative versions of such properties..
- Existing tools can do this type of analysis for huge systems by sophisticated graph algorithms

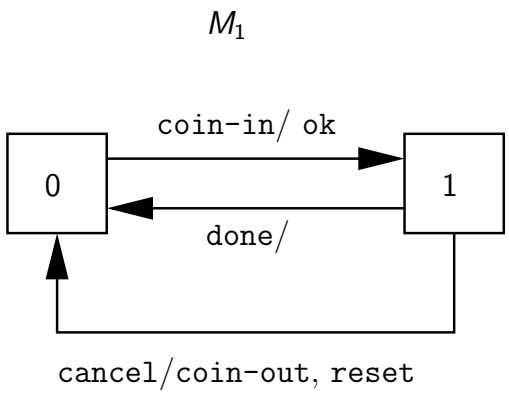
Illustration: The Coffee Machine

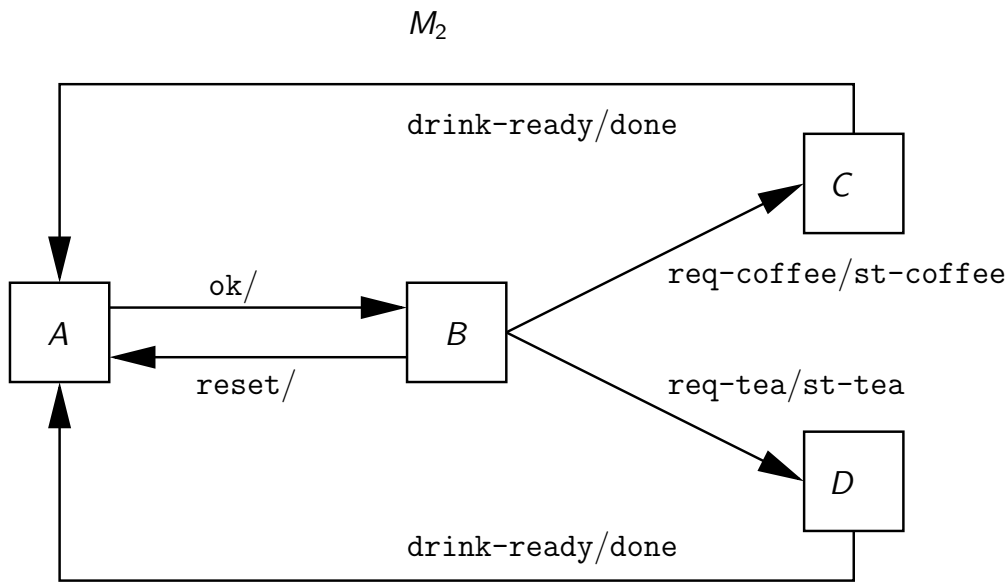
- Consider a machine that takes money and distributes drinks
- The system is built from two subsystems, one that takes care of financial matters, and one which handles choice and preparation of drinks
- They communicate by sending messages



Automaton Models

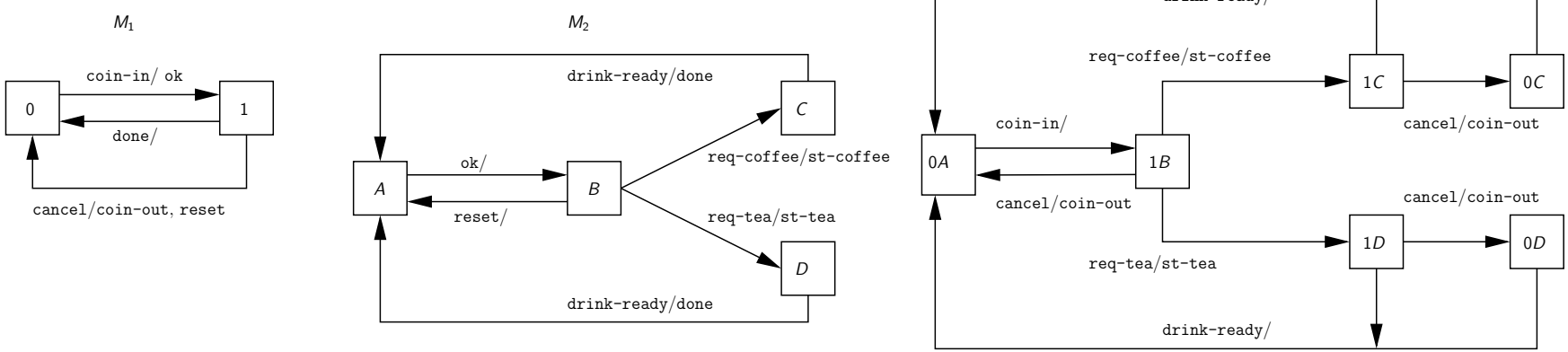
- The two systems are models as automata (state-transition systems)
- transitions are triggered by external events and events coming from the other subsystem



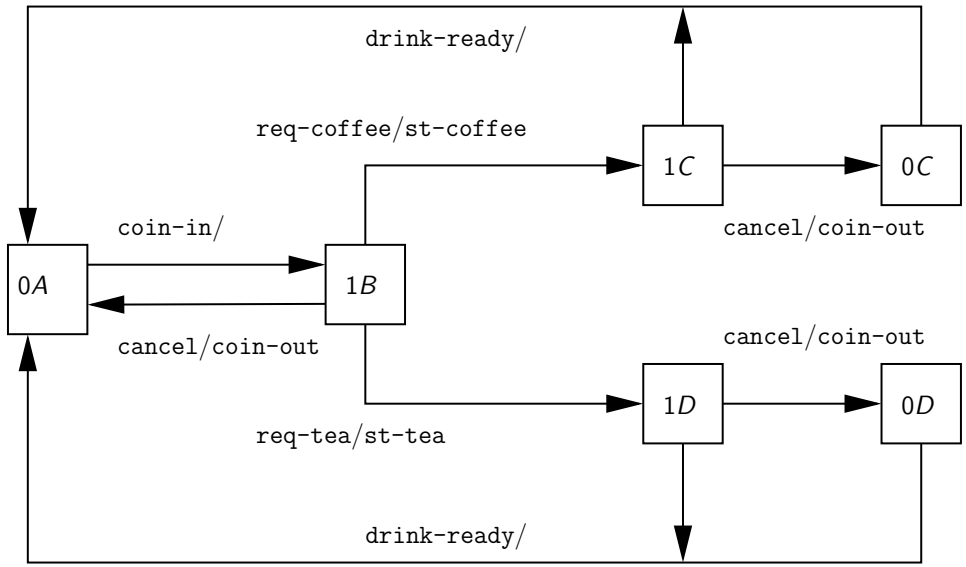


The Global Model

- The behavior of the whole system is captured by a composition (product) $M_1 \parallel M_2$ of the components
- States are elements of the Cartesian product of the respective sets of states, indicating the state of each component
- Some transitions are independent and some are synchronized, taken by the two components simultaneously
- Behaviors of the systems are paths in this transition graph



Normal Behaviors



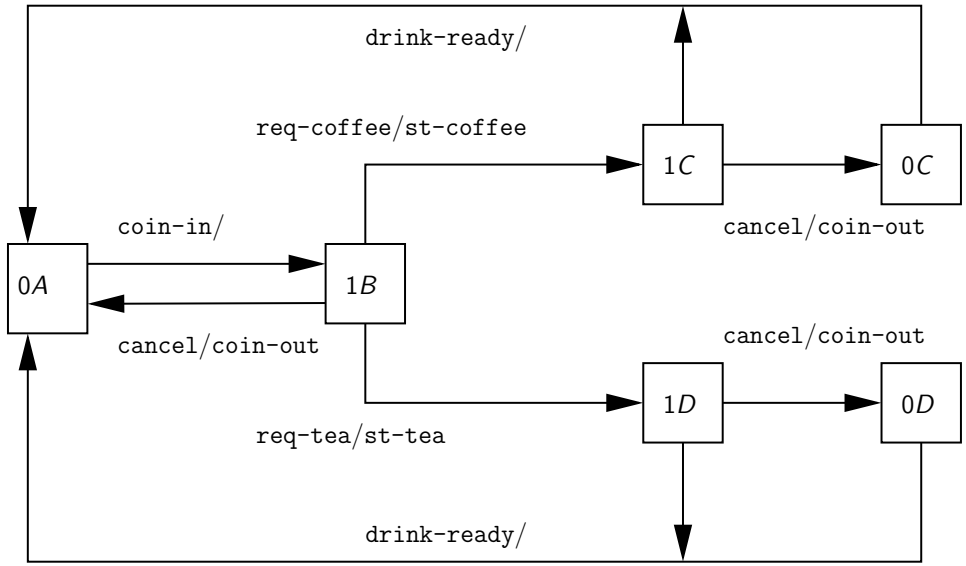
- Customer puts coin, then sees the bus arriving, cancels and gets the coin back

$$0A \texttt{ coin-in } 1B \texttt{ cancel coin-out } 0A$$

- Customer inserts coin, requests coffee, gets it and the systems returns to initial state

$$0A \texttt{ coin-in } 1B \texttt{ req-coffee st-coffee } 1C \texttt{ drink-ready } 0A$$

An Abnormal Behavior



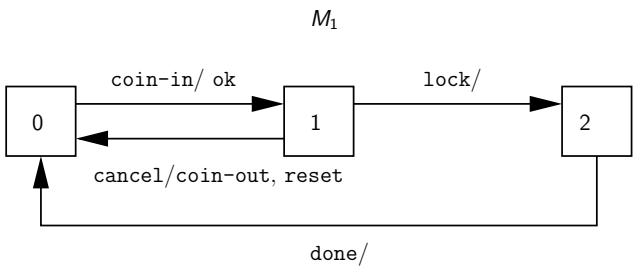
- Suppose the customer presses the cancel button *after* the coffee starts being prepared..

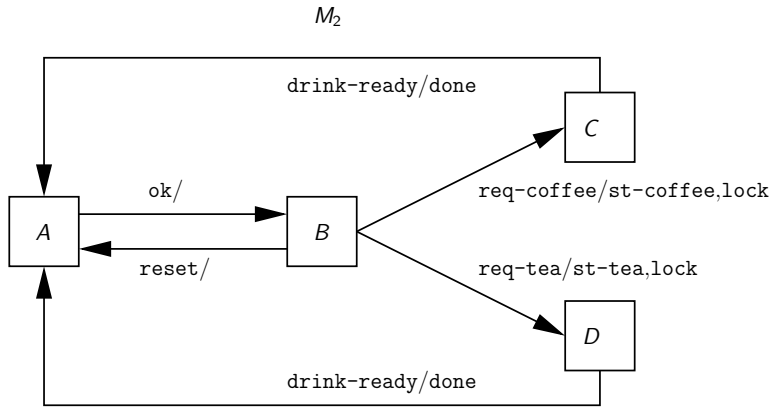
  $0A$ `coin-in` $1B$ `req-coffee` `st-coffee` $1C$ `cancel` `coin-out` $0C$ `drink-ready` $0A$

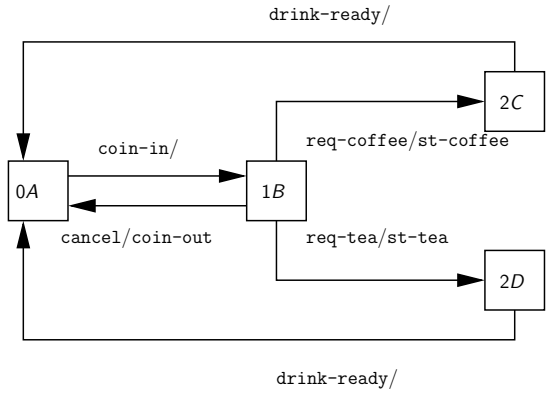
- Not so attractive for the owner of the machine

Fixing the Bug

- When M_2 starts preparing coffee it emits a `lock` signal
- When M_1 received this message it enters a new state where `cancel` is refused







The Moral of the Story I

- Many complex systems can be modeled as a composition of interacting automata
- Behaviors of the system correspond to paths in the global transition graph of the system
- The size of this graph is exponential in the number of components (state explosion, curse of dimensionality)
- These paths are labeled by input events representing influences of the outside environment
- Each input sequence may generate a different behavior
- We want to make sure that a system responds correctly to all conceivable inputs, that it behaves properly in any environment (robustness)

The Moral of the Story II

- How to ensure that a system behaves properly in the presence of all conceivable inputs and parameters?
- For every individual input sequence or parameter value we can **simulate** the reaction of the system. But we cannot do it exhaustively
- Verification is a collection of automatic and semi-automatic methods to analyze all the paths in the graph
- This is hard for humans to do and even for computers
- And this type of analysis and way of looking at phenomena is our **potential contribution** to Biology

On Levels of Abstraction

- A phenomenon can be described at different levels of abstraction and granularity
- Each level presents a trade-off in expressivity, accuracy and complexity of analysis
- When we consider processes inside the cell we encounter typically two major classes of models:
  - Evolution of protein concentrations (real numbers) following laws of mass action (continuous dynamical systems)
  - Discrete descriptions: the presence of A leads to the appearance of B which, eventually suppresses C
- I claim that not all the spectrum of possible model classes between these two has been explored

Timed Systems

- An extremely-important level of abstraction between the discrete and the continuous
- Continuous description: how the concentration of some product evolves over time
- Discrete description: the product level moves from low to high
- Timed description: the product level moves from low to high and this process takes between 3 and 5 hours to complete
- This is how we reason about our travel plans, workshop schedules and almost everything in daily life
- At this level the dynamical models are **timed automata**, automata with auxiliary clock variables

The Case for Timed Models

- Such timed discrete models will, perhaps, give a good complexity/informativeness trade-off
- This claim is illustrated (not demonstrated) using two meta-modeling case studies
  - Adding time to the purely-discrete models of genetic regulatory networks
  - Deriving timed models from continuous models (multi-affine differential equations)
- In both cases, some weaknesses of purely-discrete models are avoided
- These are proofs of concept and a lot of work remains to be done in order to improve accuracy and reduce complexity

Genetic Regulatory Networks for (and by) Dummies

- A set $G = \{g_1, \ldots, g_n\}$ of genes
- A set $P = \{p_1, \ldots, p_n\}$ of products (proteins)
- Each gene is responsible for the production of one product
- Genes activations are viewed as Boolean variables (On/Off)
- When $g_i = 1$ it will tend to increase the quantity of p_i
- When $g_i = 0$ the quantity of p_i will decrease (degradation)
- Feedback from products concentrations to genes: when the quantity of a product is below/above some threshold it may set one or more genes on or off

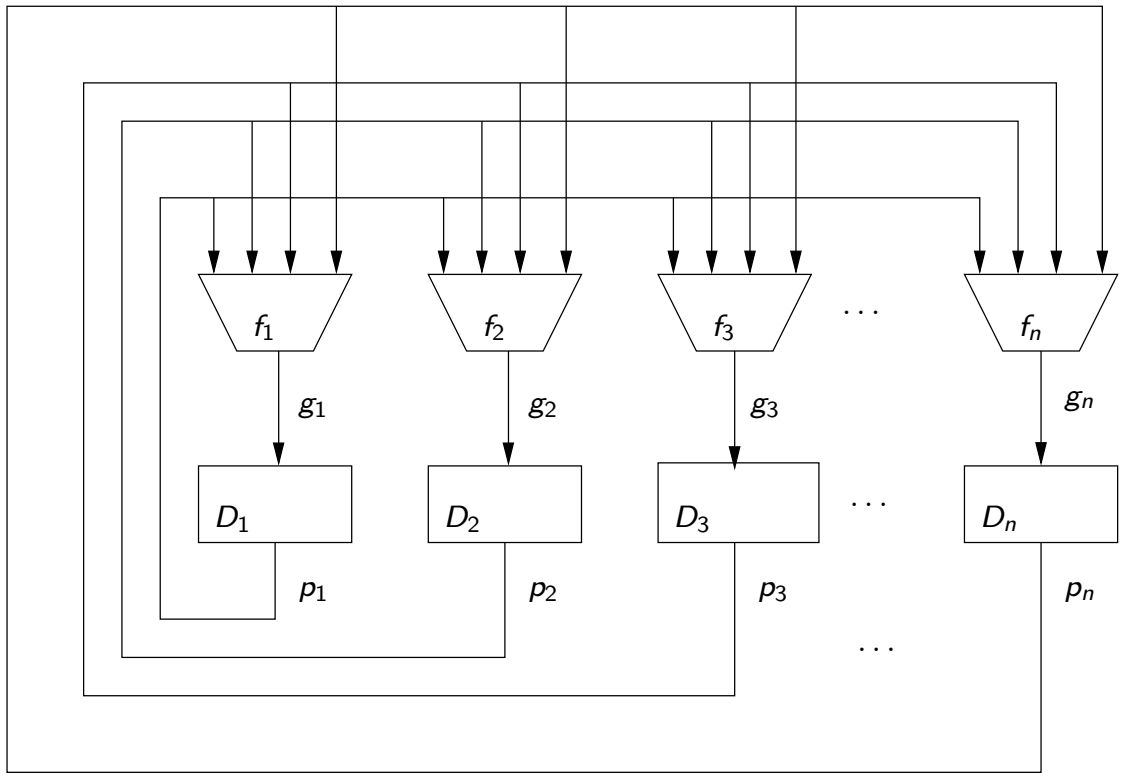
Continuous and Discrete Models of Genetic Networks

- Product quantities can be viewed as integer (quantity) or real (concentration) numbers
- The system can be viewed as a hybrid automaton with discrete states corresponding to combinations of gene activations states
- The evolution of product concentrations can be described using differential equations
- Alternatively, the domain of these concentrations can be discretized into a finite (and small) number of ranges
- The most extreme of these discretizations is to consider a Boolean domain $\{0, 1\}$ indicating **present** or **absent**

The Discrete Model of R. Thomas

- Gene activation is specified as a Boolean function over the presence/absence of products
- When a gene changes its value, its corresponding product will follow within some unspecified delay
- The resulting model is equivalent to an **asynchronous automaton**
- The relative speeds of producing different products are not modeled
- The model admits many behaviors which are not possible if these speeds are taken into account
- We add this timing information in a systematic manner, as we did in the past for asynchronous digital circuits [Maler and Pnueli 95]

Boolean Delay Networks



- A change in the activation of a gene is considered instantaneous once the value of f has changed
- This change is propagated to the product within a non-deterministic but bi-bounded delay specified by an interval

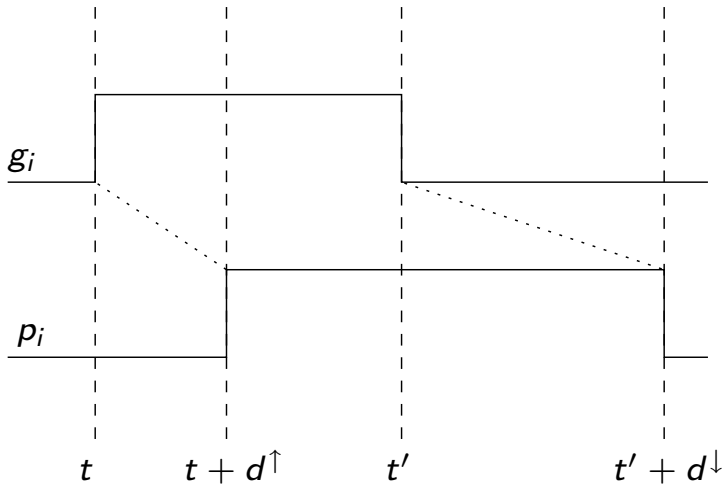
The Delay Operator

- For each i we define a delay operator D_i, a function from Boolean signals to Boolean signals characterized by 4 parameters

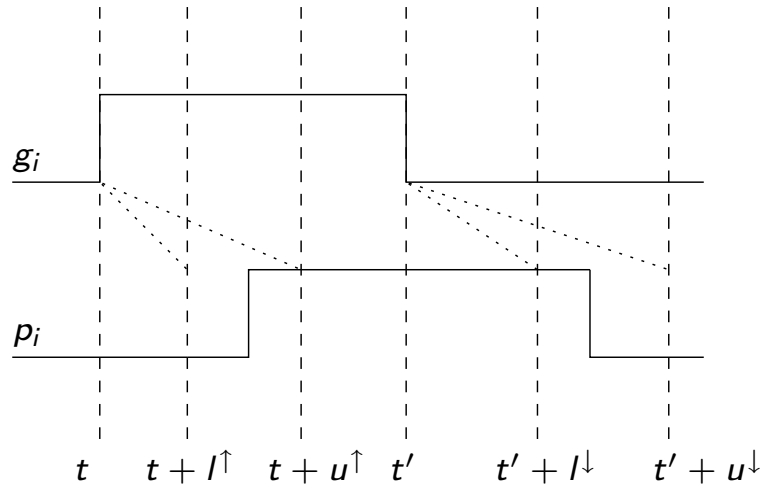
p_i	g_i	p_i'	Δ
0	0	0	–
0	1	1	$[l^{\uparrow}, u^{\uparrow}]$
1	0	0	$[l^{\downarrow}, u^{\downarrow}]$
1	1	1	–

- When $p_i \neq g_i$, p_i will catch up with g_i within $t \in [l^{\uparrow}, u^{\uparrow}]$ (rising) or $t \in [l^{\downarrow}, u^{\downarrow}]$ (falling)

The Delay Operator



Determinisitc



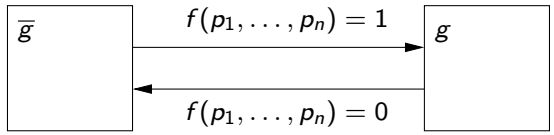
Nondeterministic

- The semantics of the network is the set of all Boolean signals satisfying the following set of signal inclusions

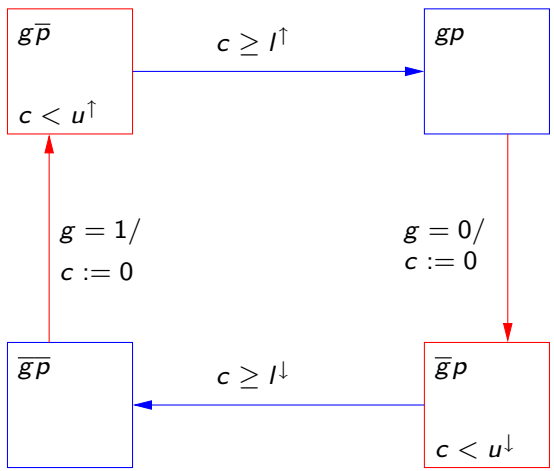
$$g_i = f_i(p_1, \ldots, p_n)$$
$$p_i \in D_i(g_i)$$

Modeling with Timed Automata

- For each equation $g_i = f_i(p_1, \ldots, p_n)$ we build the automaton



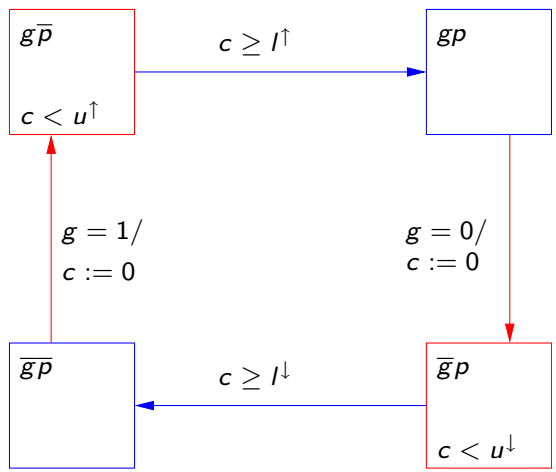
- For each delay inclusion $p_i \in D_i(g_i)$ we build the automaton



- Composing these automata we obtain a timed automaton whose semantics coincides with that of the system of signal inclusions

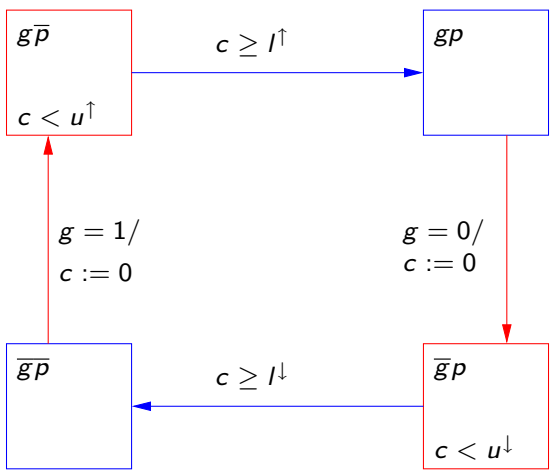
The Delay Automaton

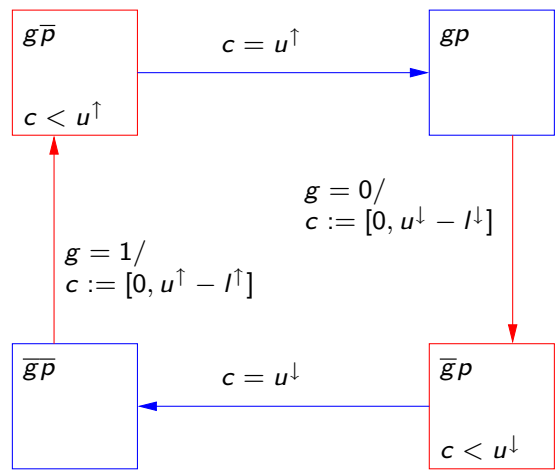
- The automaton has two stable states gp and $\overline{g}\,\overline{p}$ where the gene and the product agree
- When g changes (excitation) the automaton moves to the unstable state and resets a clock to zero
- It can stay in an unstable state as long as $c < u$ and can stabilize as soon as $c > l$.



Expressing Temporal Uncertainty

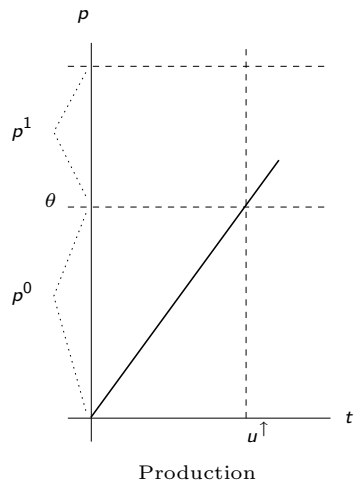
- In this automaton the uncertainty interval $[l, u]$ is expressed by the non-punctual intersection of the guard $c \geq l$ and the invariant $c < u$
- An alternative representation: making the stabilization transition deterministic and accompany the excitation transition with a non-deterministic reset



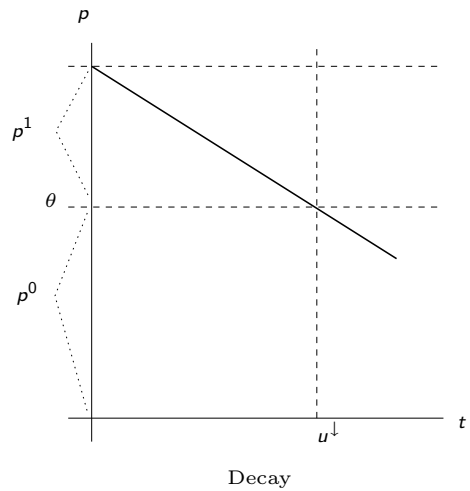


Where do Delay bounds Come From?

- These are abstractions of continuous growth and decay processes indicating the time it takes to move between points in domains $p^0 = [0, \theta]$ and $p^1 = [\theta, 1]$
- For example, for constant rates $k^{\uparrow}$ and $k^{\downarrow}$ the bounds will be $D^{\uparrow} = [0, \theta/k^{\uparrow}]$ and $D^{\downarrow} = [0, \theta/k^{\downarrow}]$



Production

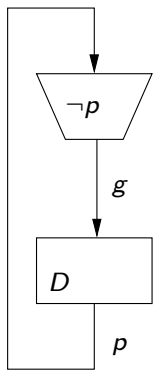


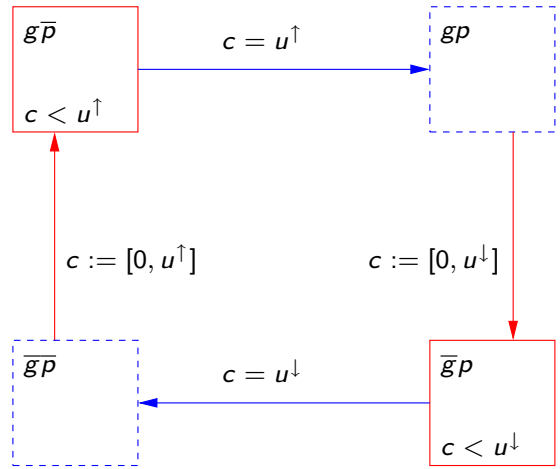
Decay

- In any case, if we want the abstraction to be conservative we should have a zero lower bound
- And this smells of Zenonism...

To Zeno or not to Zeno?

- Consider a negative feedback loop where the presence of p turns g off and its absence turns g on

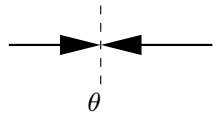




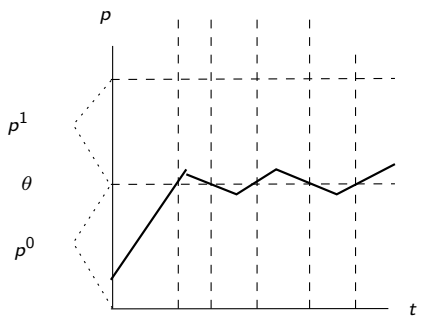
- Among the behaviors that the automaton may exhibit, if we allow a zero lower bound, is a zero time cycle
- Whether this is considered a bug or a feature depends on one's point of view
- This is related to the fundamental difference between the discrete and the continuous

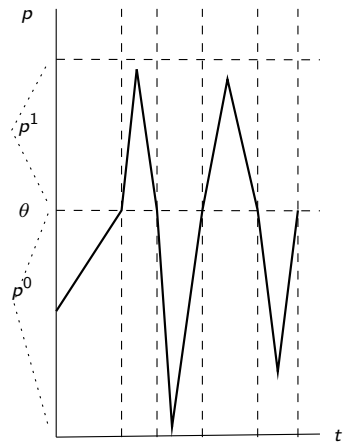
Zenonism from a Continuous Point of View

- The continuous model of the negative feedback loop is a one-dimensional vector field pointing to an equilibrium point θ



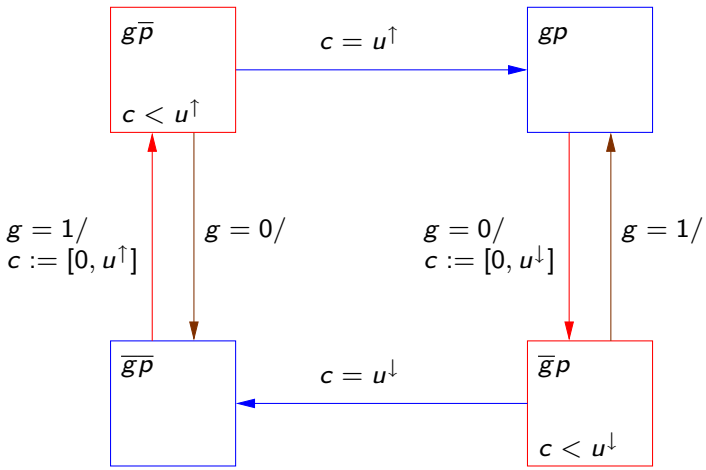
- In "reality" the value of p will have small oscillations around θ which is normal. Not much difference between θ, $\theta + \epsilon$, $\theta - \epsilon$
- Discrete abstraction amplifies this difference. The inverse image of the oscillating Boolean signal contains also large oscillations

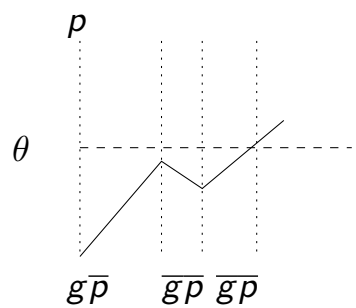


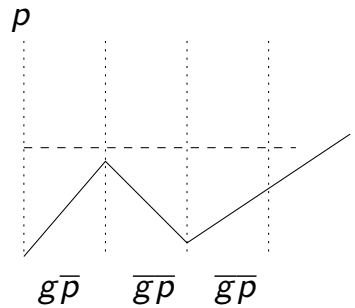


Regrets and Abortions

- Another point in favor of a zero lower bound:
- Suppose g changes, triggers a change in p and then switches back before p has stabilized, aborting the process





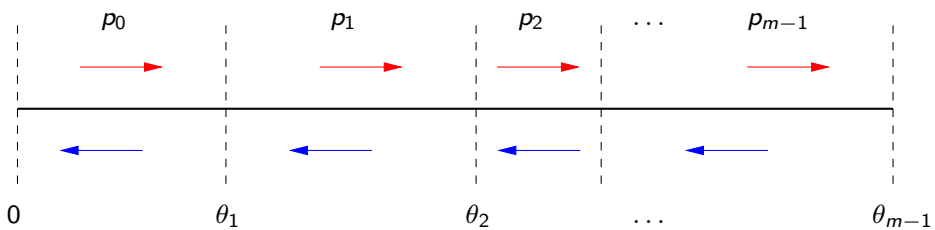


- In the "stable" state there is a decay process inside p^0
- Without additional clocks we do not now for how long
- Has the p level returned to the "nominal" low value or is still close to the threshold?

Multi-Valued Models

- The incompatibility between the discrete and the continuous is an eternal problem
- Its effect on modeling and analysis can be reduced significantly using multi-valued discrete models
- Instead of $\{0, 1\}$ we use $\{0, 1, \ldots, m-1\}$ which, via a set $0 < \theta_1 < \theta_2 < \ldots, < \theta_{m-1} < 1$ of thresholds, defines every discrete state as

$$p^i = [\theta_i, \theta_{i+1}]$$



- If you just entered p^i from p^{i-1}, you need to cross the whole p^i in order to reach p^{i+1}

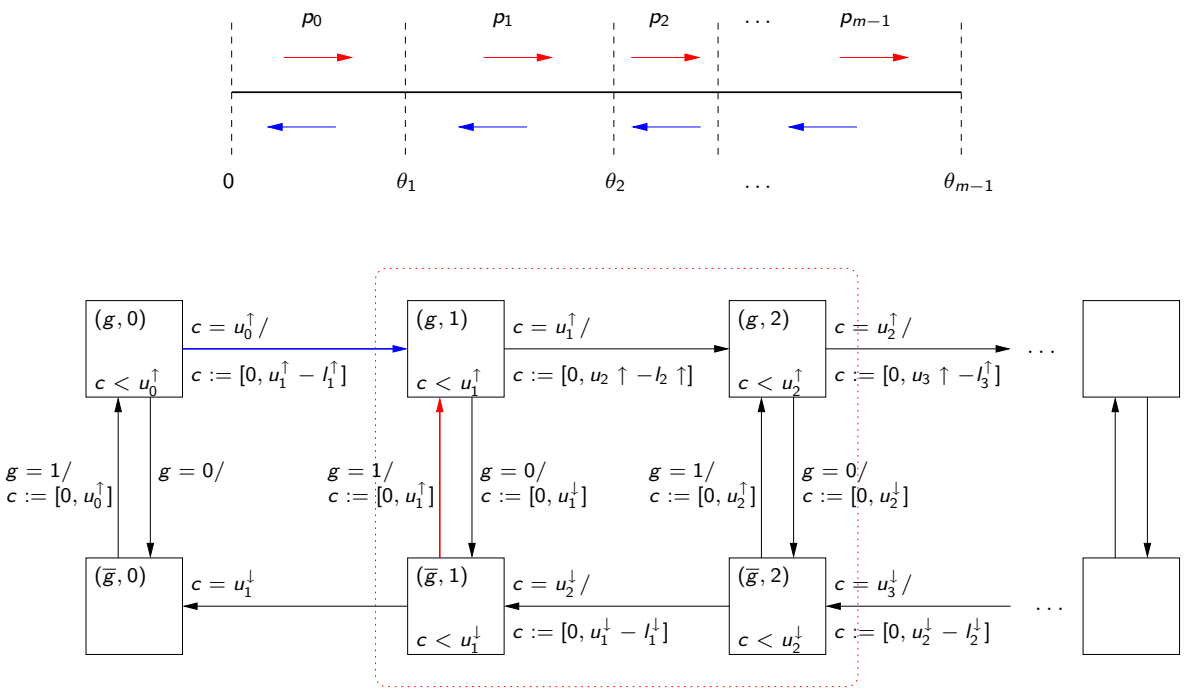
Multi-Valued Delay Operator

- The delay operator for multiple values will have $2(m-1)$ parameters in each direction.
- When $g = 1$, p will progress toward the next level and vice versa

g	p	p'	Δ	g	p	p'	Δ
0	0	0	$-$	1	0	1	$[l_0^{\uparrow}, u_0^{\uparrow}]$
0	1	0	$[l_1^{\downarrow}, u_1^{\downarrow}]$	1	1	2	$[l_1^{\uparrow}, u_1^{\uparrow}]$
0	2	1	$[l_2^{\downarrow}, u_2^{\downarrow}]$	1	2	3	$[l_2^{\uparrow}, u_2^{\uparrow}]$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
0	$m-1$	$m-2$	$[l_{m-1}^{\downarrow}, u_{m-1}^{\downarrow}]$	1	$m-1$	$m-1$	$-$

$$l_i^{\uparrow} = \min\{t : \theta_i \xrightarrow{t} \theta_{i+1}\} \qquad u_i^{\uparrow} = \max\{t : \theta_i \xrightarrow{t} \theta_{i+1}\}$$
$$l_i^{\downarrow} = \min\{t : \theta_i \xrightarrow{t} \theta_{i-1}\} \qquad u_i^{\downarrow} = \max\{t : \theta_i \xrightarrow{t} \theta_{i-1}\}$$

The Automaton for the Multi-Valued Model

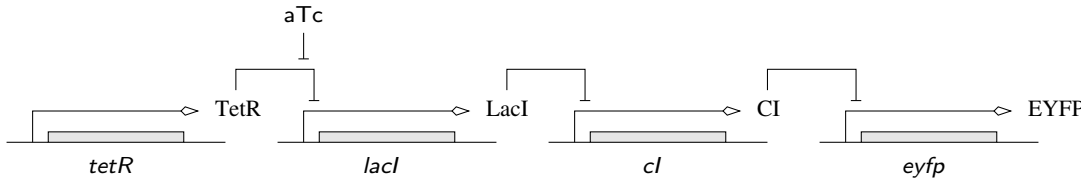


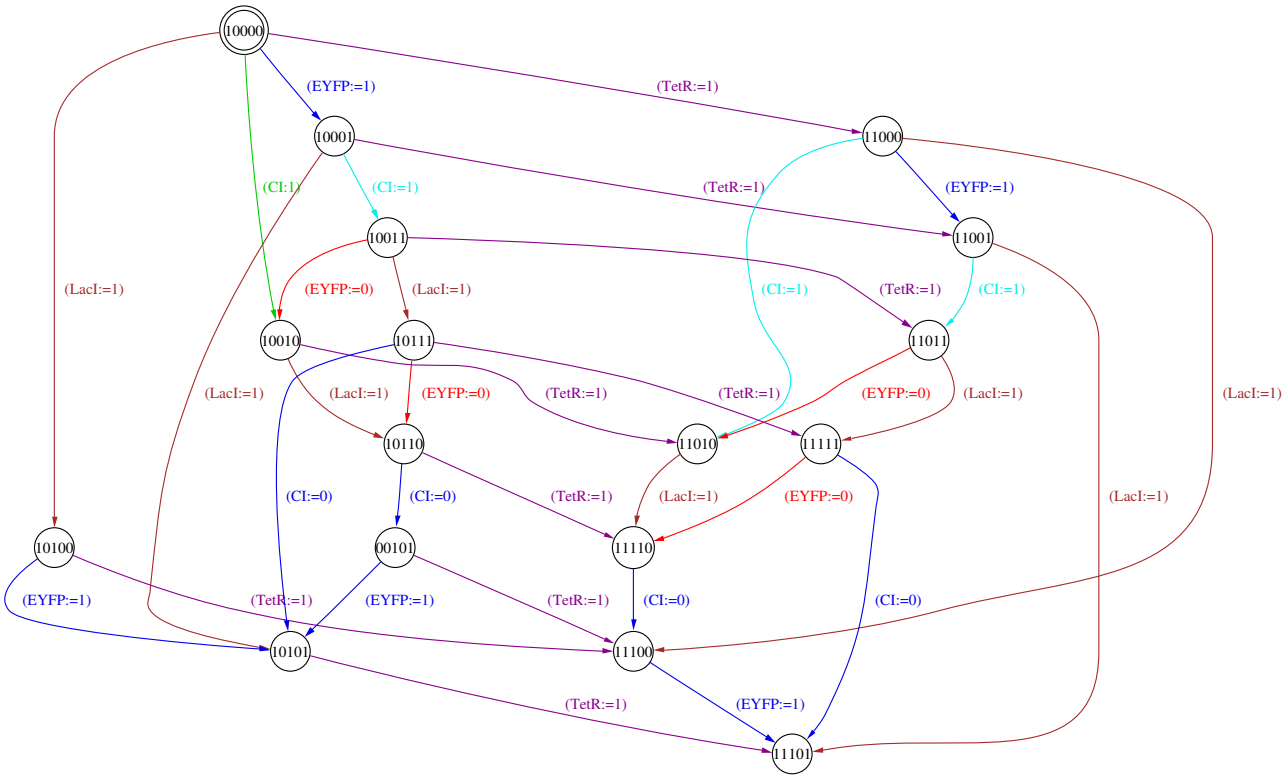
- The lower bound for moving from (g, i) to $(g, i+1)$ depends on the state from which (g, i) was entered
- If from $(g, i-1)$ (continuous evolution) then it is $l_i^{\uparrow}$
- If from $(\overline{g}, i)$ (change of direction) then it is 0
- Zero/Zeno cycles can happen only among neighbors i,$i+1$

The Global Automaton

- We then compose all these automata to obtain a global timed automaton with n clocks and roughly 2^n discrete states
- This automaton represents all the behaviors of the network while taking timing into account
- Existing tools can take a description of such a timed automaton and compute all the possible behaviors under **all** choices of delays
- We use our IF toolbox and demonstrate its capabilities on several examples
- Not much biological significance at this point (no experimental delay values available)

Example: Transcription Cascade for E. Coli

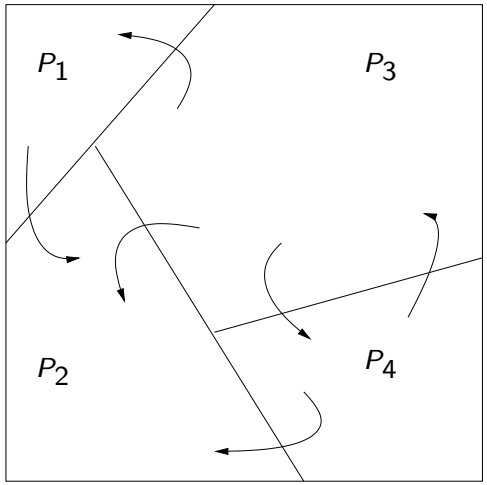


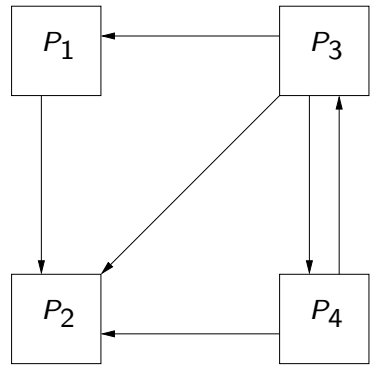


From Continuous Systems To Automata I

- Consider again a continuous dynamical system $\dot{x} = f(x)$ defined over $X \subseteq \mathbb{R}^n$
- A popular (and old) approach for analyzing such systems (qualitative physics, robotics motion planning, etc.) is to approximate it by a finite-state automaton as follows:
- Impose a finite partition $\Pi = \{P_1, \ldots P_k\}$ on X
- Define an automaton with state space Π and transition relation δ such that
- $(P, P') \in \delta$ iff P and P' are adjacent and there are points $x \in P$ and $x' \in P'$ and a trajectory leading from x to x'
- The latter fact can be sometimes determined easily by analyzing f on the boundary between P and P'

From Continuous Systems To Automata II

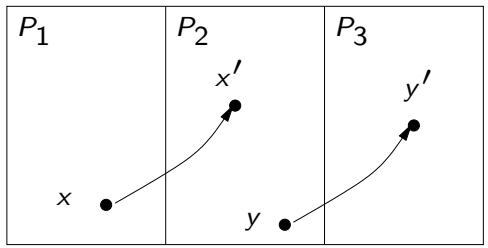


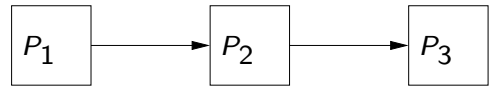


- Once you have a finite automaton you are happy because you can apply all the model-checking algorithms and tools that you already have
- But there is no free lunch

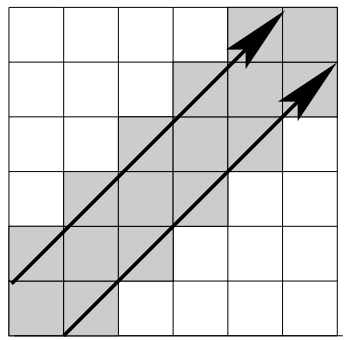
False Transitivity and Spurious Behaviors

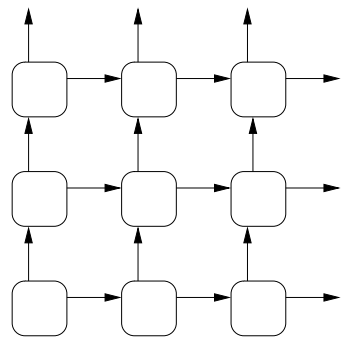
- Such abstract models often exhibit **spurious behaviors**, that are not possible in the concrete system
- You may go from $x \in P_1$ to $x' \in P_2$ and from $y \neq x' \in P_2$ to $y' \in P_3$ but not necessarily from P_1 to P_3





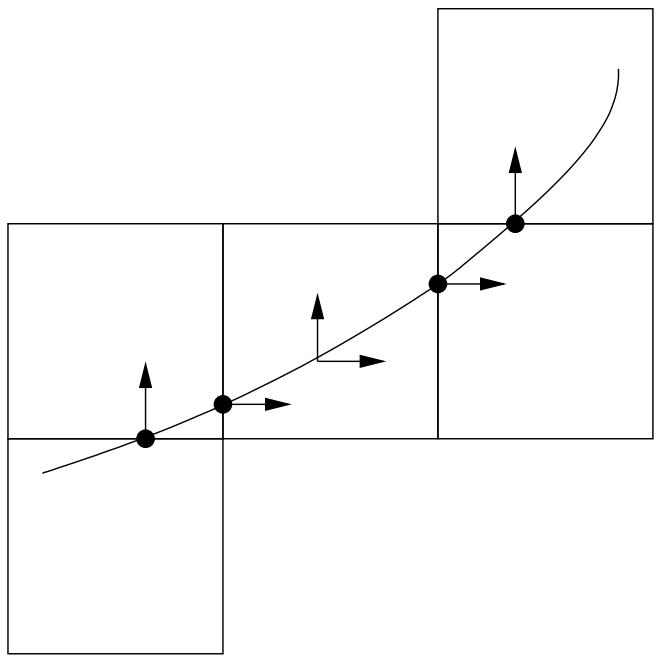
- Sometimes the approximation error renders the model useless





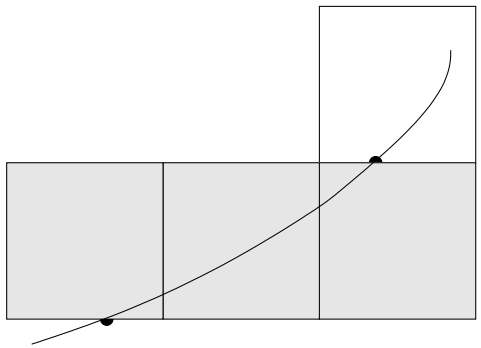
How to Reduce Spurious Behaviors

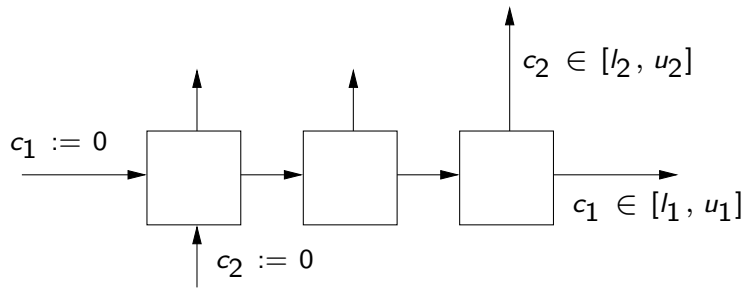
- One can see the evolution as a **competition** between state variables:
- Who will cross the next threshold in its direction
- A dimension that wins, starts the competition in the next block from an inferior position and is less likely to win again



Using Clocks

- We associate a clock c_i with each dimension which is reset whenever a boundary is crossed in direction i
- The next transition in **the same** direction is constrained to occur when $c_i \in [l_i, u_i]$





- The constants are inferred from the minimal and maximal value of f_i in the corresponding "slice" (slightly circular reasoning)
- It is easy to compute these min-max values for multi-affine systems

Current and Future Status

- Prototype implementation, does not work on the fly but generates the whole model in the IF format
- Not surprisingly, works rather well in monotone parts of the state space. In parts where some f_i admits a zero we need to be more careful
- Some examples, not yet convincing
- For the more general class of polynomial systems, extremal values of f_i should be computed numerically
- Future: a tighter tool integration, automatic choice of partition thresholds, model-checking against MITL

Back to the Big Picture

- Biology needs (among other things) more dynamic models to form verifiable predictions
- These models can benefit from the accumulated understanding of dynamical system within informatics and cannot rely only on 19th century mathematics
- The views of dynamical system developed within informatics are, sometimes, more adapted to the complexity and heterogeneity of Biological phenomena
- Biological modeling should be founded on various types of dynamical models: continuous, discrete, hybrid and timed
- These models should be strongly supported by computerized analysis tools offering a range of capabilities from simulation to verification and synthesis

Back to the Big Picture

- Systems Biology should combine insights from:
- Engineering disciplines: modeling and analysis of very complex man-made systems (chips, control systems, software, networks, cars, airplanes, chemical plants)
- Physics: experience in mathematical modeling of natural systems with measurement constraints
- Mathematics and Informatics as a unifying theoretical framework

Thank You