On Timed Models of Gene Networks

Gregory Batt, Ramzi Ben Salah, Oded Maler

Verimag

2007

Systems Biology

- Systems Biology: the new gold rush for many mathematical and technical disciplines
- Biophysics, Biomatics, Bioinformatics, ...

Systems Biology

- Systems Biology: the new gold rush for many mathematical and technical disciplines
- Biophysics, Biomatics, Bioinformatics, ...
- In our domain: application of (Petri nets, process algebras, rewriting systems, logic, probabilistic systems, hybrid systems, ...) to biological modeling
- So why not Timed Automata?

More Seriously

- The study of biological phenomena may benefit from dynamic models that allow predictions concerning the evolution of processes over time
- Complex processes with different types of state variables that may represent different types of entities (gene activation, product concentration) evolving with different time scales

More Seriously

- The study of biological phenomena may benefit from dynamic models that allow predictions concerning the evolution of processes over time
- Complex processes with different types of state variables that may represent different types of entities (gene activation, product concentration) evolving with different time scales
- The choice of dynamical models used by biologists (e.g. differential equations, Boolean networks) is sometimes accidental, not always reflecting all that exists in other mathematical and engineering disciplines and what is appropriate for the phenomena
- As in other domains, timed models can play an important role

Summary of This Work

- Motivation: genetic regulatory networks
- Existing discrete models based on asynchronous automata

Summary of This Work

- Motivation: genetic regulatory networks
- Existing discrete models based on asynchronous automata
- We use delay equations on signals and build timed automata based on previous work on asynchronous circuits

Summary of This Work

- Motivation: genetic regulatory networks
- Existing discrete models based on asynchronous automata
- We use delay equations on signals and build timed automata based on previous work on asynchronous circuits
- We extend the model from Boolean to multi-valued

Summary of This Work

- Motivation: genetic regulatory networks
- Existing discrete models based on asynchronous automata
- We use delay equations on signals and build timed automata based on previous work on asynchronous circuits
- We extend the model from Boolean to multi-valued
- Implement in IF and show feasibility on some examples

Summary of This Work

- Motivation: genetic regulatory networks
- Existing discrete models based on asynchronous automata
- We use delay equations on signals and build timed automata based on previous work on asynchronous circuits
- We extend the model from Boolean to multi-valued
- Implement in IF and show feasibility on some examples
- No new significant mathematical or biological results but interesting observations on discrete timed modeling of continuous processes

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)

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

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 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)

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 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

- Product quantities can be viewed as integer (quantity) or real (concentration of molecules in the cell) 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 on concentration

Continuous and Discrete Models

- Product quantities can be viewed as integer (quantity) or real (concentration of molecules in the cell) 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 on concentration
- 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 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

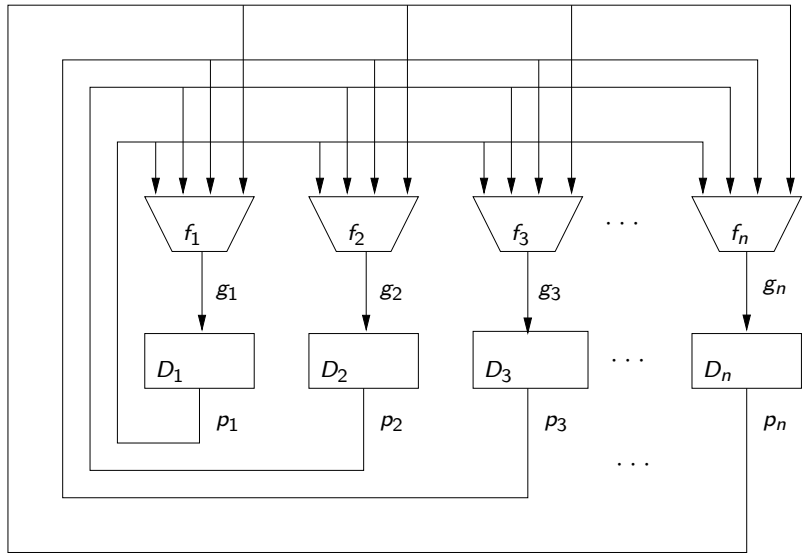
The Discrete Model of R. Thomas

- 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

The Discrete Model of R. Thomas

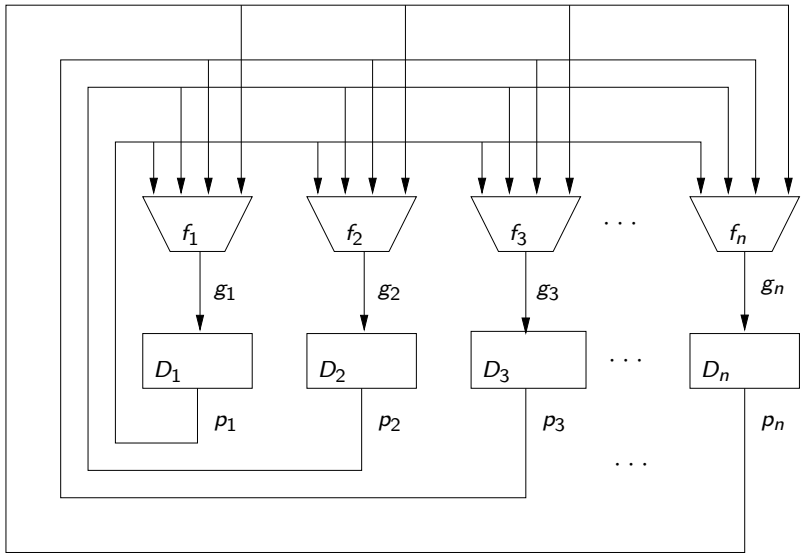
- Gene activation is specified as a Boolean function over the presence/absence of products
- When a gene changes its value, its corresponding product will follow within some unspecified delay
- The resulting model is equivalent to an asynchronous automaton
- The relative speeds of producing different products are not modeled
- The model admits many behaviors which are not possible if these speeds are taken into account
- We want to 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

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

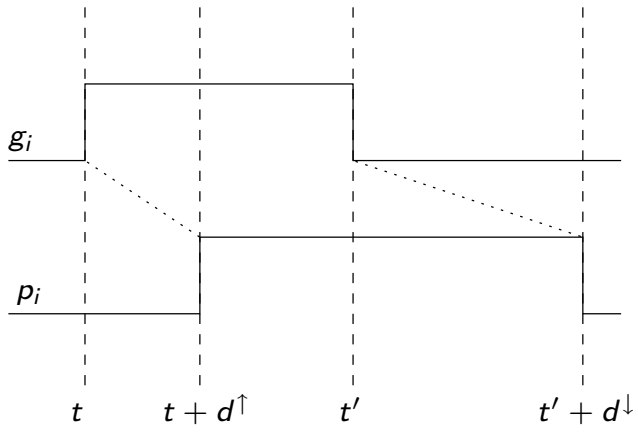
The Delay Operator

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

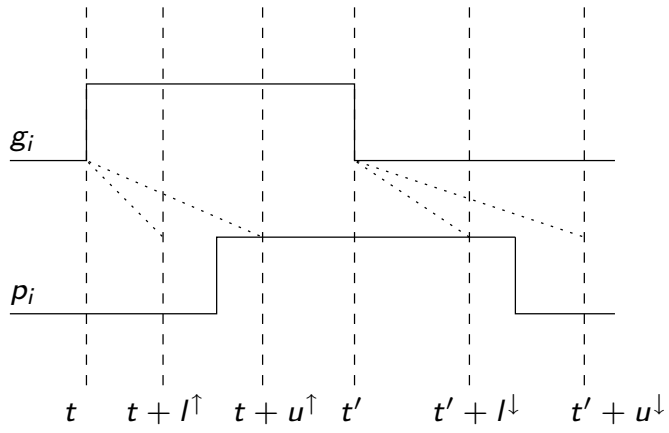
p_i	g_i	p_i'	Δ
0	0	0	–
0	1	1	$[l^{\uparrow}, u^{\uparrow}]$
1	0	0	$[l^{\downarrow}, u^{\downarrow}]$
1	1	1	–

- 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

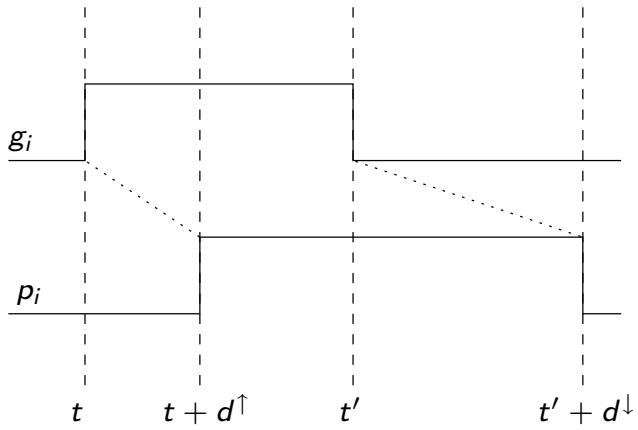


Determinisitc

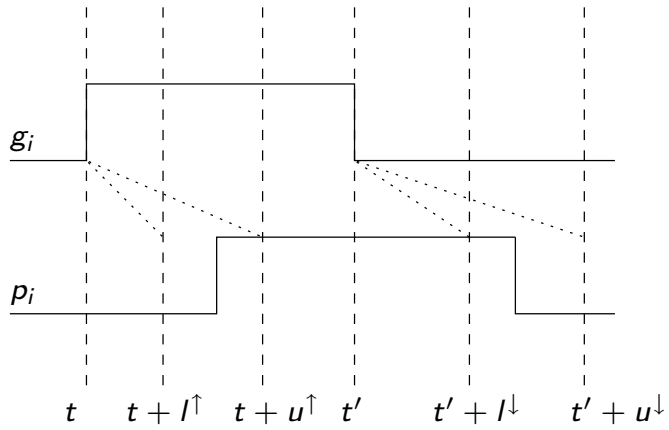


Nondeterministic

The Delay Operator



Determinisitc



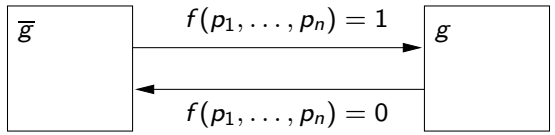
Nondeterministic

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

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

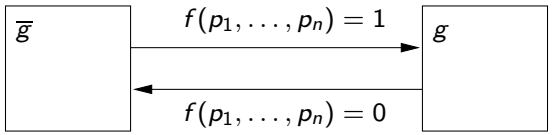
Modeling with Timed Automata

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

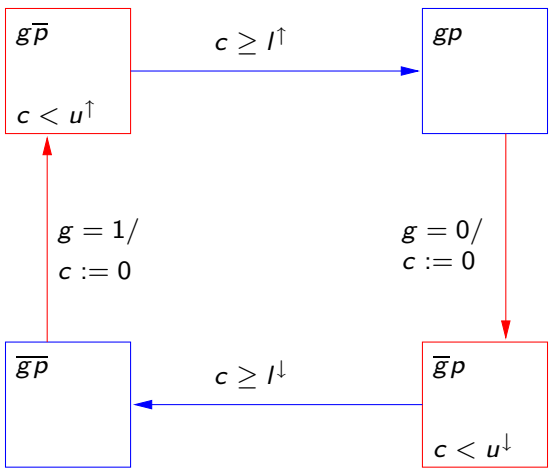


Modeling with Timed Automata

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

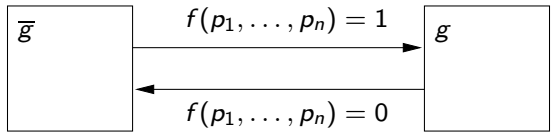


- For each delay inclusion $p_i \in D_i(g_i)$ we build the automaton

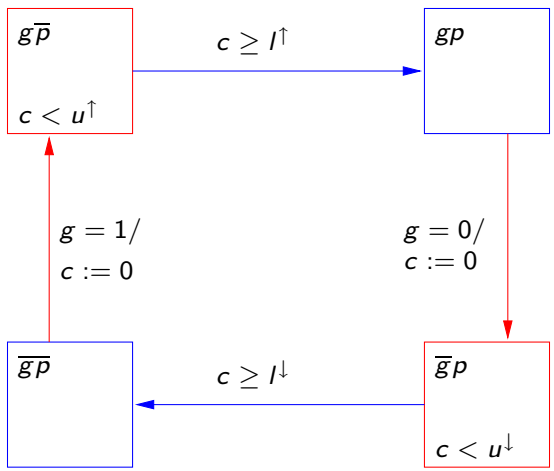


Modeling with Timed Automata

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



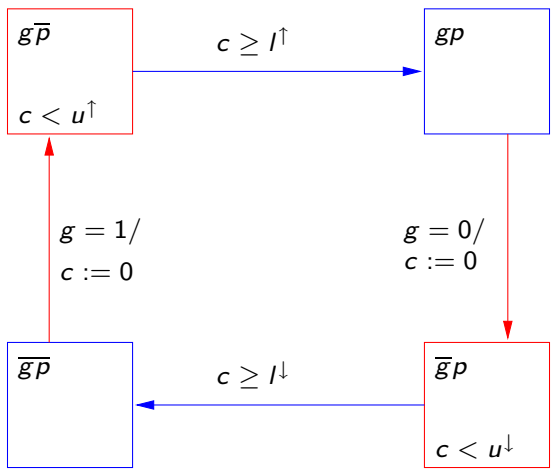
- For each delay inclusion $p_i \in D_i(g_i)$ we build the automaton



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

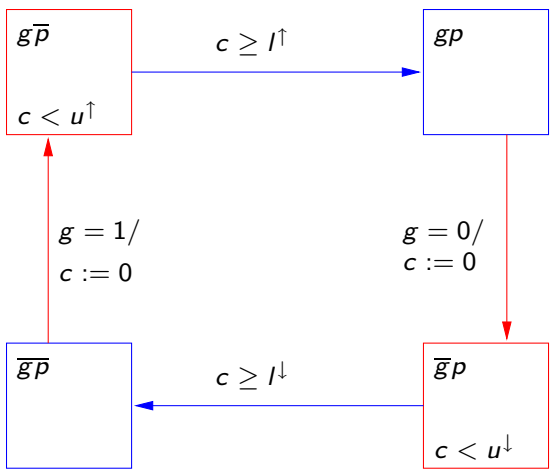
The Delay Automaton

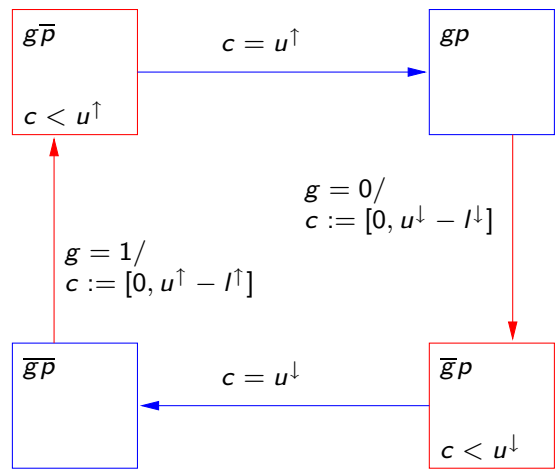
- The automaton has two stable states gp and $\overline{gp}$ where the gene and the product agree
- When g changes (excitation) it moves to the unstable state and reset a clock to zero
- It can stay in an unstable state as long as $c < u$ and can stabilize as soon as $c > l$.



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?

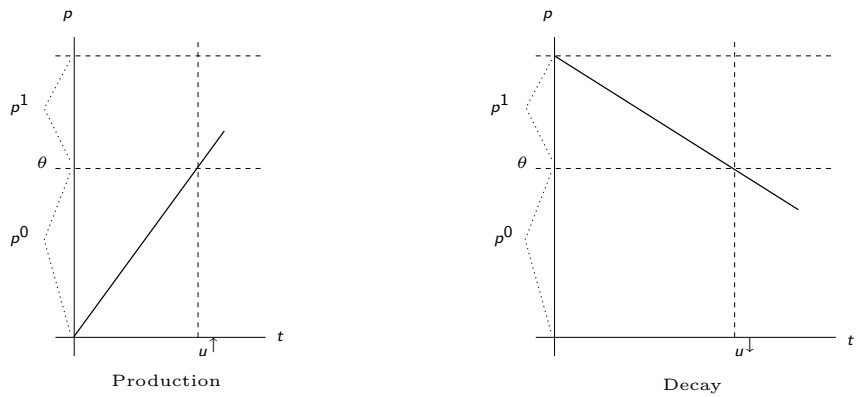
- These are abstractions of continuous growth and decay processes indicating the time it takes to move between points in domains $p^0 = [0, \theta]$ and $p^1 = [\theta, 1]$

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 $D^{\downarrow} = [0, \theta/k^{\downarrow}]$

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 $D^{\downarrow} = [0, \theta/k^{\downarrow}]$



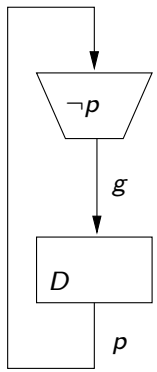
- In any case, if we want the abstraction to be conservative we should have a zero lower bound
- And this smells of Zenonism...

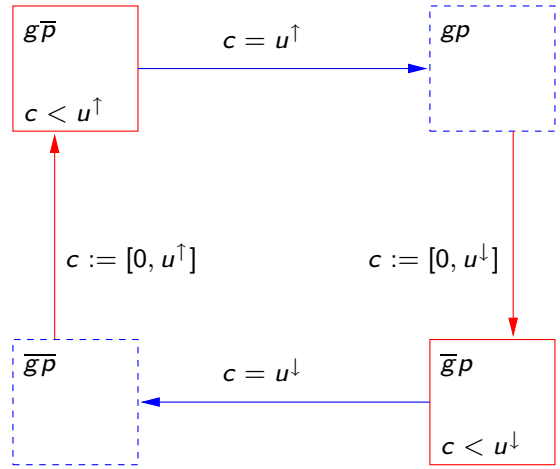
To Zeno or not to Zeno?

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

To Zeno or not to Zeno?

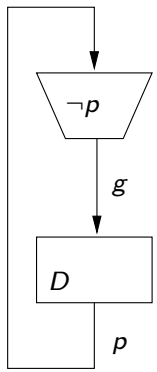
- Consider a negative feedback loop where the presence of p turns g off and its absence turns g on

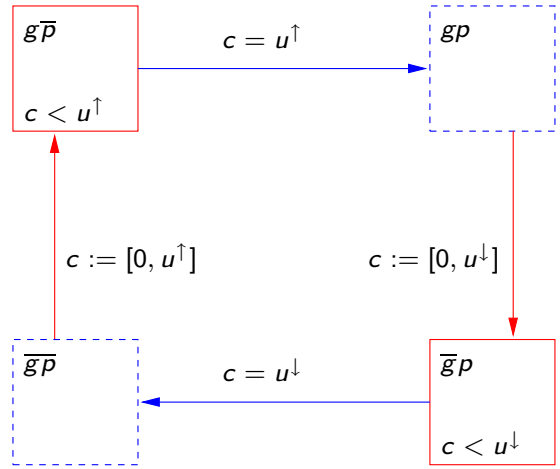




To Zeno or not to Zeno?

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

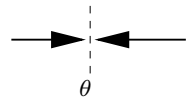




- Among the behaviors that the automaton may exhibit, if we allow a zero lower bound, is a zero time cycle
- Whether this is considered a bug or a feature depends on one's point of view
- This is related to the fundamental difference between the discrete and the continuous

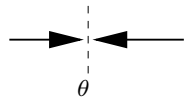
Zenonism from a Continuous Point of View

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



Zenonism from a Continuous Point of View

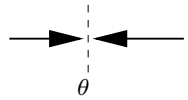
- The continuous model of the negative feedback loop is a one-dimensional vector field pointing to an equilibrium point θ



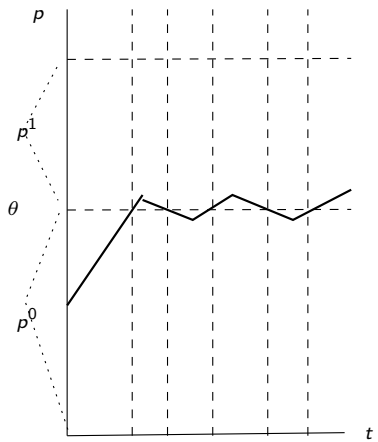
- In "reality" the value of p will have small oscillations around θ which is normal. Not much difference between θ, $\theta + \epsilon$, $\theta - \epsilon$

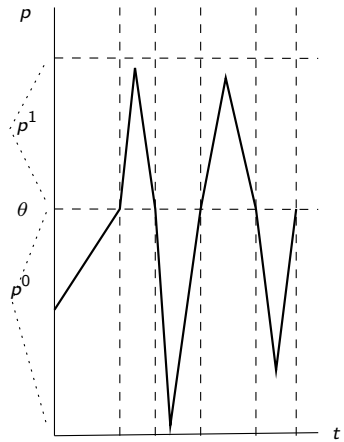
Zenonism from a Continuous Point of View

- 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



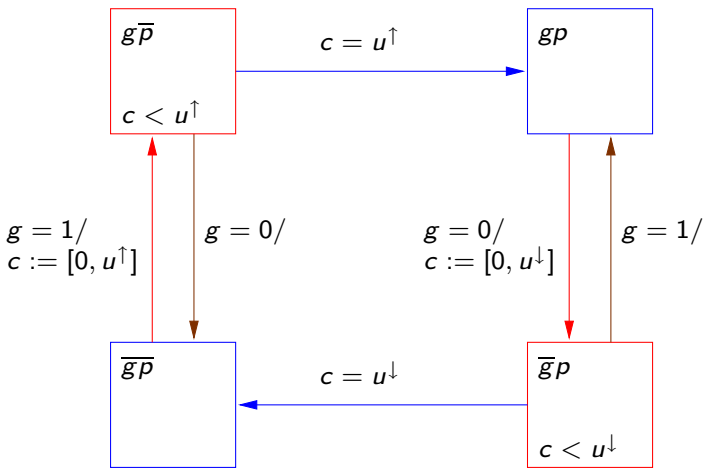


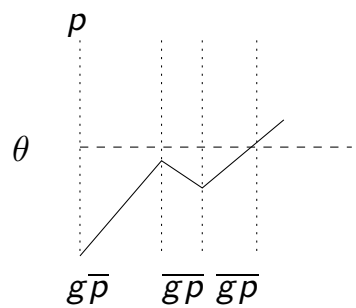
Regrets and Abortions

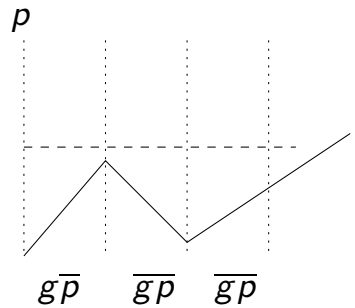
- Another point in favor of a zero lower bound:
- Suppose g changes, triggers a change in p and then switches back before p has stabilized, aborting the process

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

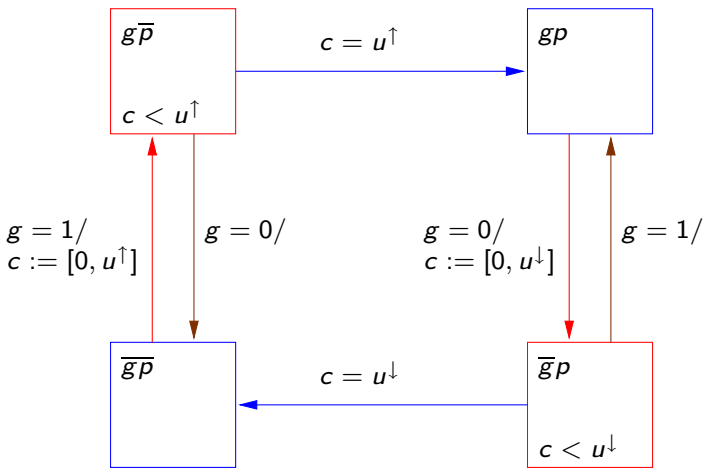


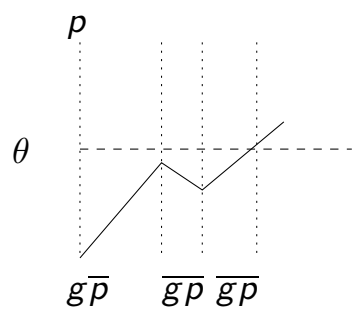


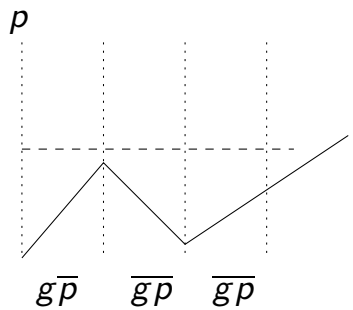


Regrets and Abortions

- 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

Multi-Valued Models

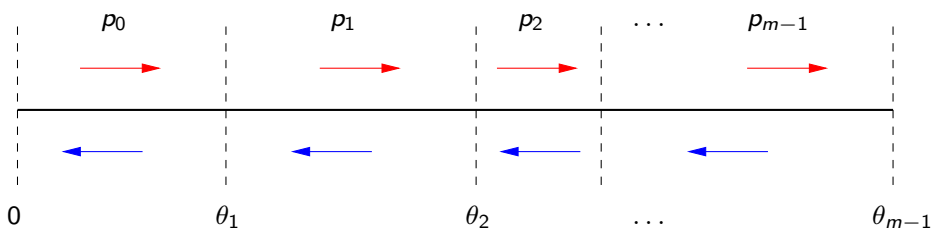
- The incompatibility between the discrete and the continuous is an eternal problem
- Its effect on modeling and analysis can be reduced significantly using multi-valued discrete models
- Instead of $\{0, 1\}$ we use $\{0, 1, \ldots, m-1\}$ which, via a set $0 < \theta_1 < \theta_2 < \ldots, < \theta_{m-1}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

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

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

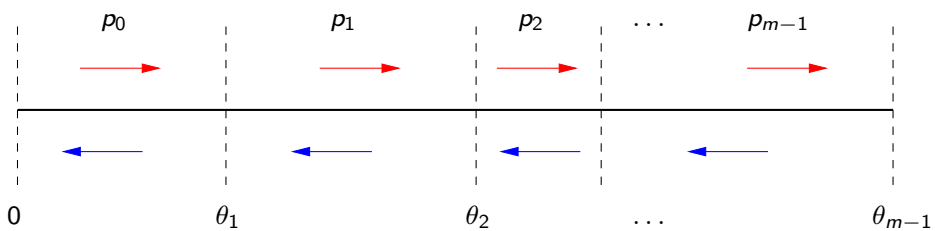
$$p^i = [\theta_i, \theta_{i+1}]$$



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	$[l_2^{\uparrow}, u_2^{\uparrow}]$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
0	$m-1$	$m-2$	$[l_{m-1}^{\downarrow}, u_{m-1}^{\downarrow}]$	1	$m-1$	$m-1$	$-$

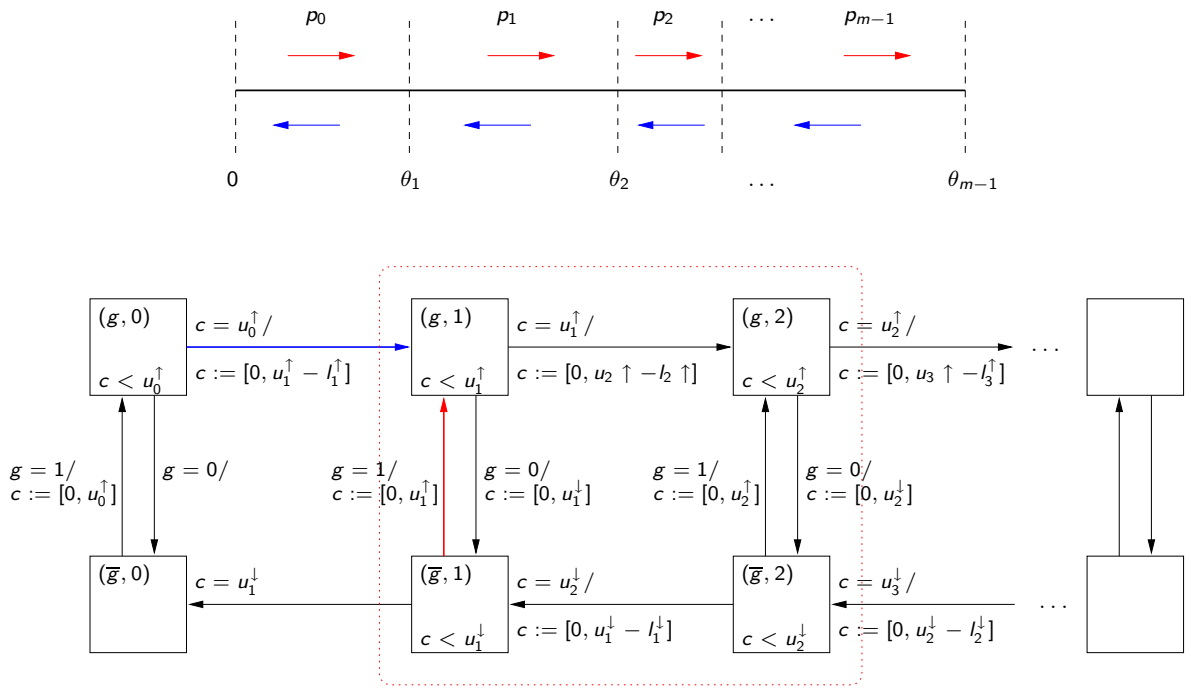
Multi-Valued Delay Operator

- 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	$[l_2^{\uparrow}, u_2^{\uparrow}]$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
0	$m-1$	$m-2$	$[l_{m-1}^{\downarrow}, u_{m-1}^{\downarrow}]$	1	$m-1$	$m-1$	$-$

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

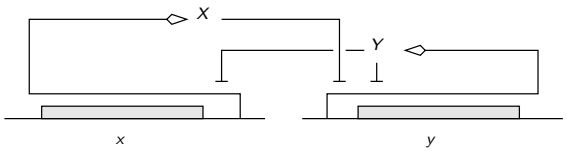
The Automaton for the Multi-Valued Model

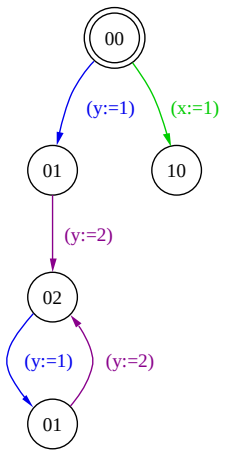


- The lower bound for moving from (g, i) to $(g, i+1)$ depends on the state from which (g, i) was entered
- If from $(g, i-1)$ (continuous evolution) then it is $l_i^{\uparrow}$
- If from $(\overline{g}, i)$ (change of direction) then it is 0
- Zero/Zeno cycles can happen only among neighbors i,$i+1$

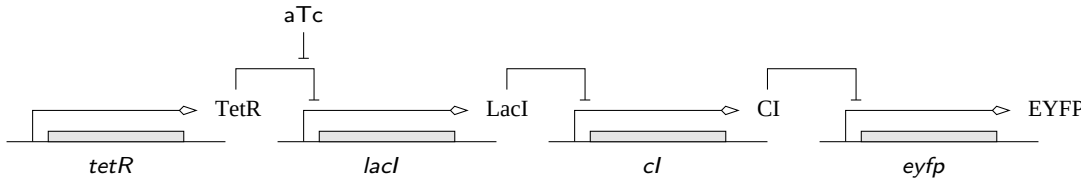
Implementation and Experiments

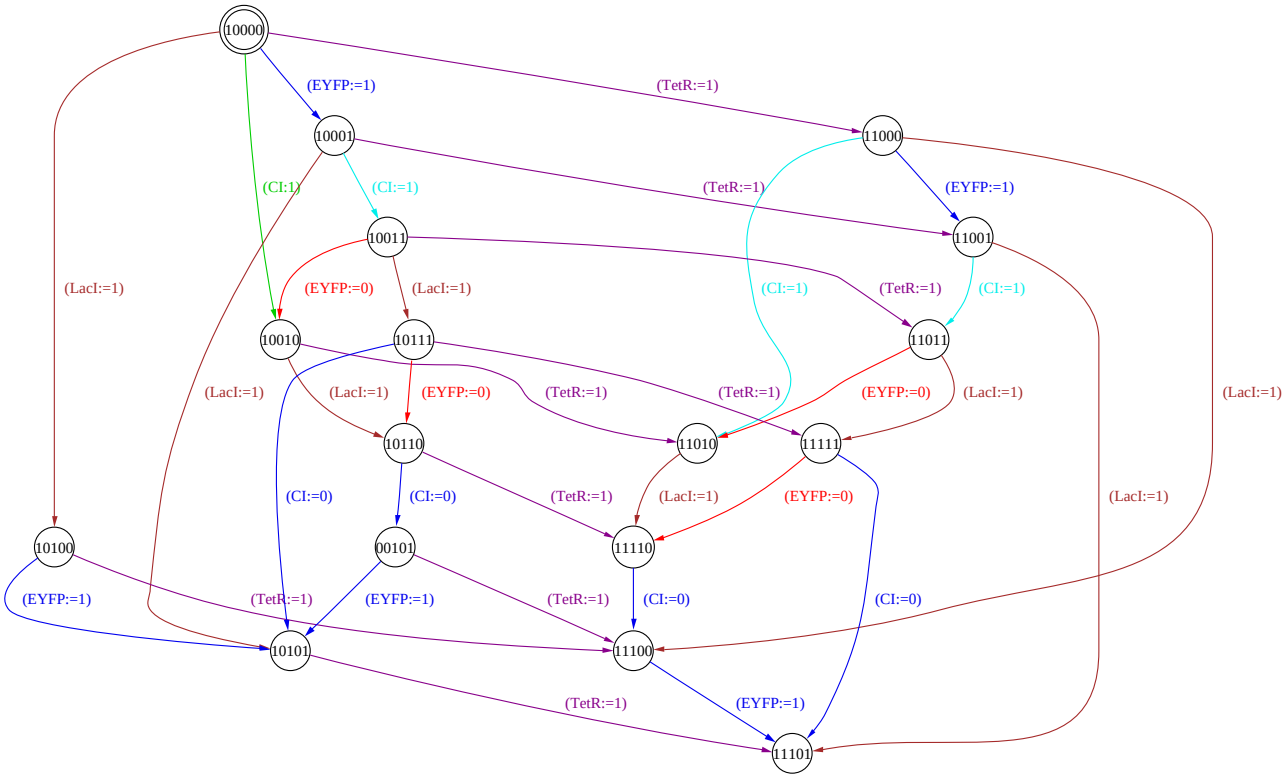
- Implementation in the IF toolbox including translation from delay inclusions to timed automata
- Analysis of several examples to show feasibility (not much biological significance at this point)
- Example 1: a cross inhibition network (also modeled by [Siebert and Bockmayr 06])





Transcription Cascade for E. Coli



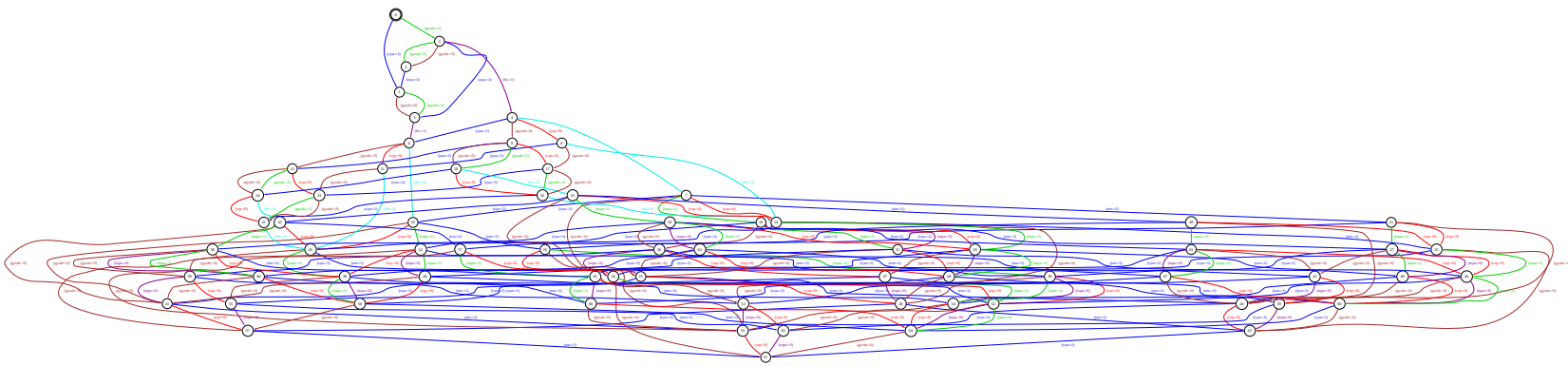


Nutritional Stress Response for E. Coli

- Six genes, six proteins and one additional variable encoding the presence or absence of nutrition
- Arbitrarily chosen delays, cyclic behaviors
- Reachability graph with 69 states and 209 transitions, computed in less than one second

Nutritional Stress Response for E. Coli

- Six genes, six proteins and one additional variable encoding the presence or absence of nutrition
- Arbitrarily chosen delays, cyclic behaviors
- Reachability graph with 69 states and 209 transitions, computed in less than one second



Conclusions and Future Work

- Provided a systematic translation from timed gene networks to timed automata
- Extended the model to include multiple-values and reduce the effect of Zeno behavior
- Demonstrated feasibility on non-trivial examples

Conclusions and Future Work

- Provided a systematic translation from timed gene networks to timed automata
- Extended the model to include multiple-values and reduce the effect of Zeno behavior
- Demonstrated feasibility on non-trivial examples
- Future: refine gene activation from binary to multi-valued. Requires refinement of the feedback function
- Future: combine with LTL and MITL model checking against the generated models
- Future: promote the idea of timed modeling among biologists and see what experiments are needed to extract timing information