On Timed Models of Gene Networks*

Gregory Batt, Ramzi Ben Salah and Oded Maler

VERIMAG, 2, av. de Vignate, 38610 Gieres, France
Gregory.Batt@imag.fr Ramzi.Salah@imag.fr Oded.Maler@imag.fr

Abstract. We present a systematic translation from timed models of genetic regulatory networks into products of timed automata to which one can apply verification tools in order learn about the possible qualitative behaviors of the network under a whole range of uncertain delay parameters. We have developed a tool chain starting from a high-level description of the network down to an exhaustive analysis of its behavior. We have demonstrated the potential applicability of this framework on several examples.

1 Introduction

The evolving domain of *Systems Biology* attempts to advance the quality of biological models to become closer to models of simpler hard sciences like Physics. Given the complexity of such models and the difficulty in obtaining experimental results for determining exact model parameters, it is no big surprise that the engineering disciplines in general, and various "semantics and verification" sub communities, are among those who try to sell their modeling and analysis methodologies to biologists. The current paper is no exception as it attempts to demonstrate the applicability of timed systems for modeling and analysis of genetic regulatory networks.

We are concerned with models that cover a subset of what is going on inside a single cell where the major actors are *genes* at certain levels of activation ("expression") and the *products*, typically proteins that they produce in a cell. The interaction between these entities is often modeled by biologists using various forms of *interaction diagrams* that indicate the mutual influences among these entities. At this level of abstraction one may associate continuous variables to genes (to indicate the level of gene expression) and to products (to indicate their concentration in the cell). Based on the corresponding interaction diagrams, one can, in principle, derive dynamical models that track the evolution of these quantities over time. At this level of description, the dynamical model will be essentially a system of nonlinear differential equations[1] derived from the corresponding chemical processes.

A major problem with such models is that the parameters of the equations are very difficult to obtain experimentally and can be known only within very large uncertainty

* This work was partially supported by the French-Israeli project *Computational Modeling of Incomplete Biological Regulatory Networks.*

[1] Or a hybrid automaton with nonlinear dynamics in each mode, to accommodate for discrete modeling of changes in gene expression.

margins. Such systems can, in principle, be analyzed using novel hybrid systems verification techniques, but although such techniques have recently matured for linear systems [ADF$^+$06], their adaptation to nonlinear systems is only in its infancy. An alternative well-known modeling approach is based on *discrete* models where genes can be either *on* or *off* and the values of the product concentrations are discretized into a finite number of levels. In the extreme case when products can be only "absent" and "present" one can obtain an abstract Boolean model with a finite automaton dynamics of the form we all love and appreciate. Historically a *synchronous* Boolean model was first proposed by Kauffman [K69] followed by an alternative *asynchronous* model, proposed by Thomas in a series of papers and an influential book [TD90]. The rationale for Thomas' asynchronous model is that if two processes are active in parallel, one producing product p_1 and another producing p_2, it is very unlikely that both of them will terminate simultaneously in the "next" time step. Termination here means that their product concentrations will cross their respective thresholds between *present* and *absent*. The asynchronous model allows the processes to complete in any order.

While these asynchronous networks are more faithful to reality, they ignore information which may be known about the relative *speeds* of the processes, information that can be exploited to restrict the set of possible qualitative behaviors of the automaton. And indeed, recently Siebert and Bockmayr [SB06] proposed to enrich Thomas' model with *timing information* and replace automata with timed automata. They have used the tool UPPAAL to analyze such a model of a small network. In a completely different context, Maler and Pnueli [MP95] gave a formal treatment of asynchronous networks of Boolean gates with *uncertain (bi-bounded) delays* and a systematic method for translating such networks into timed automata. This framework has been since then the basis of numerous efforts for verification and timing analysis of such circuits [BMT99,BJMY02,BBM03]. In this paper we adapt this framework for the timed modeling of genetic regulatory networks and provide a tool chain for translating such networks into products of timed automata and analyzing its behaviors. The main improvement over [SB06] is in the *systematic* translation from timed gene networks to timed automata. At this point we do not claim having discovered any new biological result but rather demonstrate that timed models of non trivial phenomena can indeed be analyzed by our existing timed automata tools.

The rest of the paper is organized as follows. In Section 2 we describe our modeling framework based on genes, products, Boolean functions and delay operators. In Section 3 we show how such models are transformed into timed automata and discuss some anomalies inherent in discrete and timed modeling of continuous processes. To make such models more faithful to reality, we extend the framework of [MP95] in Section 4 to any finite number of concentration levels. Some experimental results on several examples are reported in Section 5, followed by some discussion of ongoing and future work. We assume familiarity with timed automata and present them in a quite informal manner to facilitate their comprehension by potential users.

2 Boolean Delay Networks

Let $G = \{g_1, \ldots, g_n\}$ be a set of "genes" viewed as Boolean variables which can be either *on* or *off*. Let $P = \{p_1, \ldots, p_n\}$ be a set of "products" or "proteins" that we assume to be represented by Boolean variables as well, where $p_i = 0$ means that the corresponding product is found in low concentration and $p_i = 1$ indicates that its concentration is high (absent/present in discrete parlance). We assume here that each gene g_i is responsible for the production of one product type p_i. Intuitively when g_i is 1 it will tend to produce p_i so that the latter, if absent, will sooner or later become present. On the other hand if g_i is off, so will become p_i eventually due to degradation. In control terms the value of g_i can be seen as a *reference signal* for the desired value of p_i, a value that it will reach if the process is not disturbed.

The feedback in the system is based on changing the state of the genes according to the concentration levels of the products. Among the common types of interaction between genes and the proteins they produce we mention:

- *Self-inhibition*: the presence of p_i turns g_i off;
- *A cascade of activations*: when p_i becomes present it turns on some g_j.

More generally, we assume the value of each g_i to be a *Boolean function* f_i of $p_1, \ldots, p_n$. We assume such a change in gene activation to be *instantaneous* upon the change in (discretized) concentration, if the latter changes the value of the corresponding Boolean function. On the other hand, the production and degradation of products is modeled as taking some time between initiation and termination.

This difference in modeling is justified by the nature and time scales of the underlying chemical processes. Producing one molecule of the product is already a very complex process involving numerous reactions. Changing the state of p_i from absent to present may involve producing thousands of such molecules. Although some of this can be done in parallel (or more accurately, with *pipelining*) it still may take a lot of time. On the other hand, activating or inhibiting a gene is a relatively-simpler and faster process where some molecule binds to some site and enables or inhibits the production process.[2]

Our modeling approach is based on the premise that at every time instant t, the state of a gene g_i is determined by the values of the P-variables *at time* t via a Boolean function f_i. On the other hand, the influence of each p_i on g_i is not immediate and is best expressed via a *delay operator*, a function whose output follows the input after some delay. Figure 1 illustrates graphically the relationship between G and P using a *block-diagram* formalism. The f_i-boxes react immediately to a change in their inputs and alter g_i, and the latter influences p_i after some delay specified by the D_i-boxes. Looking closer, each delay operator D_i can be characterized by a table of the following

[2] The dependence of such a binding event upon the presence level of some p is more of a stochastic nature as the more molecules we have, more this is likely to happen. In fact, the "real" story is more complex as the activation of a gene is not a Boolean business either, and there are different levels of gene expression, each leading to different production rates and delays. These rates depend on the concentration of the products via stochastics, but all this is beyond the scope of the present paper.

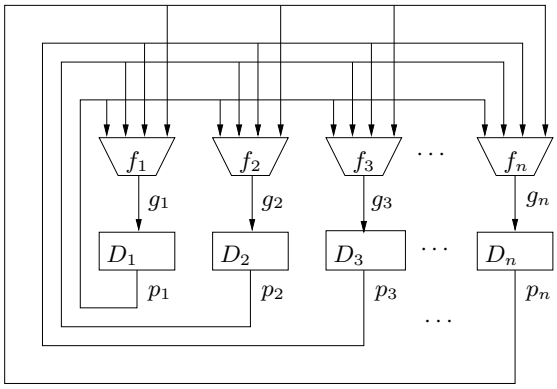


Fig. 1. A graphical representation of a gene network. The f_i's are arbitrary Boolean functions of P and the D_i's are the delay operators.

form:

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	$-$

(1)

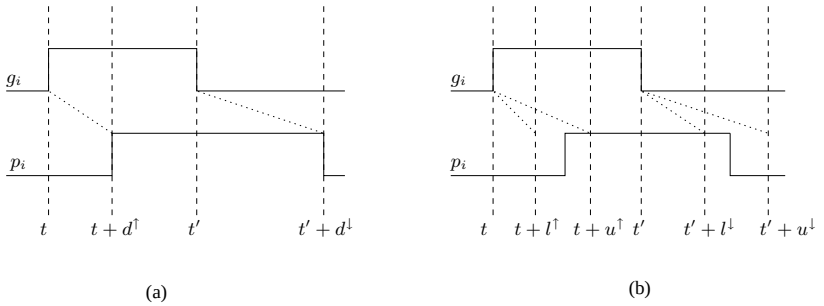
In this table, p_i' indicates the value of the product the systems "aims at" and should eventually reach. When p_i and g_i agree (00 or 11) the system is in a *stable* state and $p_i' = p_i$. In state 01 the product starts being produced and will become 1 if undisturbed. The time delay for moving from absence to presence is expressed using the interval $I^{\uparrow} = [l^{\uparrow}, u^{\uparrow}]$ to be explained in the sequel. Likewise, at state 10 the product degrades and will become 0 within $I^{\downarrow} = [l^{\downarrow}, u^{\downarrow}]$ time. To better explain the delay operator let us start with a deterministic delay model. Let $d^{\uparrow} = l^{\uparrow} = u^{\uparrow}$ and $d^{\downarrow} = l^{\downarrow} = u^{\downarrow}$. A typical behavior of this operator is illustrated in Figure 2-(a) where gene g_i is turned on at time t and then p_i follows and becomes present at time $t + d^{\uparrow}$. Later, at t', gene g_i is turned off and p_i completes its degradation to low level at $t' + d^{\downarrow}$.

It is, however, unrealistic to assume *exact* delays given the inherent noise in biological systems and general experimental limitations. Even in the absence of those, deterministic delays should be excluded due to the fact that discrete state 0 represents a *range* of concrete concentrations and it is clear that from each of them it will take a different amount of time to cross the threshold to reach the domain of 1. This feature is covered by our non-deterministic delay operator which allows the response of p_i to occur anywhere in the interval $[t + l, t + u]$, see Figure 2-(b).

We use Boolean signals[3] to formalize the D_i operators. A *Boolean signal* x is a function from the time domain $\mathbb{R}_+$ to $\mathbb{B} = \{0, 1\}$ which admits a partition of $\mathbb{R}_+$ into a countable set of (left-closed right-open) intervals $I_0, I_1, \ldots$ where each I_i is of the form $[t_i, t_{i+1})$ such that if t and t' belong to the same interval, $x(t) = x(t')$. We will assume here, just for ease of notation, that all input signals start with 0, hence x is 0 in all intervals I_j with even j, and 1 when j is odd. The set $J(x) = t_1, t_2, \ldots$ is called the *jump set* of x, that is, the set of time points where the value of the signal changes.

Definition 1 (Delay Operator). *Let x and y be two Boolean signals with $J(x) = \{t_1, t_2, \ldots\}$ and $J(y) = \{s_1, s_2, \ldots\}$, and let D_i be a delay operator characterized by the delay parameters $l_i^{\uparrow}$, $u_i^{\uparrow}$, $l_i^{\downarrow}$ and $u_i^{\downarrow}$. We say that $y \in D_i(x)$ if for every j*

$$\begin{array}{ll} s_j \in [t_j + l_i^{\uparrow}, t_j + u_i^{\uparrow}] & \textit{if } j \textit{ is odd} \\ s_j \in [t_j + l_i^{\downarrow}, t_j + u_i^{\downarrow}] & \textit{if } j \textit{ is even} \end{array}$$



(a) (b)

Fig. 2. (a) Deterministic delay $p_i = D_i(g_i)$; (b) Non-deterministic delay $p_i \in D_i(g_i)$.

We can now define the semantics of the network (1), that is all the temporal behaviors it may generate over the values of the G and P variables. These are all the signals satisfying the following system of signal inclusions[4] for $i = 1, \ldots, n$:

$$\begin{array}{l} g_i = f_i(p_1, \ldots, p_n) \\ p_i \in D_i(g_i) \end{array} \tag{2}$$

3 Modeling with Timed Automata

Our translation from delay equations to timed automata is compositional as we build a timed automaton for each equation and inclusion in (2) so that the composition of these

[3] To avoid confusion with other meanings of "signals" in Biology, we stress that the word *signal* is used here in the sense used in of signal processing, that is, a function from time to some domain, a waveform, a temporal pattern.

[4] If the delay was deterministic, the second line would be replaced by $p_i = D_i(g_i)$ and there would be a unique solution from any given initial state.

automata generates exactly the set of signals satisfying the equations. For each instantaneous relation $g_i = f_i(p_1, \ldots, p_n)$ we construct a one-state automaton over $(n+1)$-dimensional signals whose self-loop transitions are labeled by tuples $(g_i, p_1, \ldots, p_n)$ that satisfy the equation. The heart of our modeling approach is the automaton for the delay operator which can be seen as the continuous-time analog of the one-bit shift register.

The timed automaton of Figure 3-(a) realizes the delay operator $p \in D(g)$. We annotate states by the values of g and p where $\overline{p}$ stands for $p = 0$ and p stands for $p = 1$. At state $\overline{pg}$ the product is absent, the gene is off and the automaton may stay in this stable state forever as long as g, which is viewed as an *input* for this automaton, remains off. When g is turned on, the clock c is reset to zero and the automaton moves to the excited state $g\overline{p}$ in which it can stay as long as $c < u^{\uparrow}$ but can leave to stable state gp, where the gene is on and the product is present, as soon as $c \geq l^{\uparrow}$.

The uncertainty in process duration is expressed in this automaton by the possibility to stay at an unstable state until $c = u$ but leave it as soon as $c \geq l$. An alternative modeling style is to move the non determinism to the triggering transition and making the stabilizing transition deterministic as in Figure 3-(b). Here instead of being reset to zero, the clock is set non deterministically to a value in $[0, u-l]$ and the transition guard is replaced with $c = u$. In both cases the result is the same, there are uncountably-many behaviors (and outputs) that the automaton may exhibit in the presence of one input. We will use the second approach in this paper as it extends more naturally from Boolean to multi-valued domains.

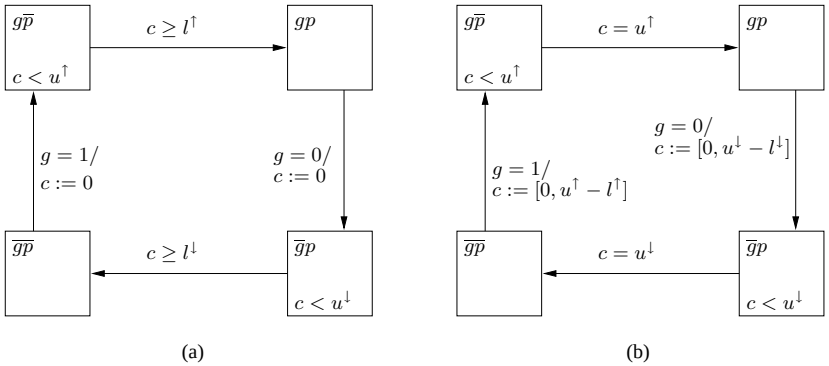


Fig. 3. Timed automaton models for the delay operator: (a) non determinism in the stabilizing transition; (b) non determinism in the exciting transition.

Before proceeding further let us contemplate a bit on the relation between the delay bounds and the underlying continuous process. Figure 4 illustrates hypothetical production and decay processes to which such a timed discrete abstraction could correspond. To simplify the discussion assume that the concentration of p grows at a fixed rate $k^{\uparrow}$

when g is turned on, and decreases with rate $k^{\downarrow}$ when g is off.[5] The mapping from the concrete domain of concentrations, the interval $[0,1]$ to $\{0,1\}$ is based on partitioning the interval into $p^0 = [0,\theta]$ and $p^1 = [\theta,1]$. The delay interval $D^{\uparrow}$ should thus indicate the minimal and maximal times it takes for a trajectory starting at *any* point in p^0 to cross the θ-threshold to p^1. Following this reasoning we obtain

$$D^{\uparrow} = [l^{\uparrow}, u^{\uparrow}] = [0, \theta/k^{\uparrow}] \quad \text{and} \quad D^{\downarrow} = [0, \theta/k^{\downarrow}].$$

The zero lower bounds come from the fact that the starting point can be a point in p^0 which is *as close as we want* to θ. Not having a positive lower bound is sometimes considered an undesirable feature in timed models as it may create zero-time oscillations between 0 and 1, also known as Zeno behaviors. However, in our context, having a positive lower bound would exclude behaviors which are legitimate in the continuous context as we illustrate below.

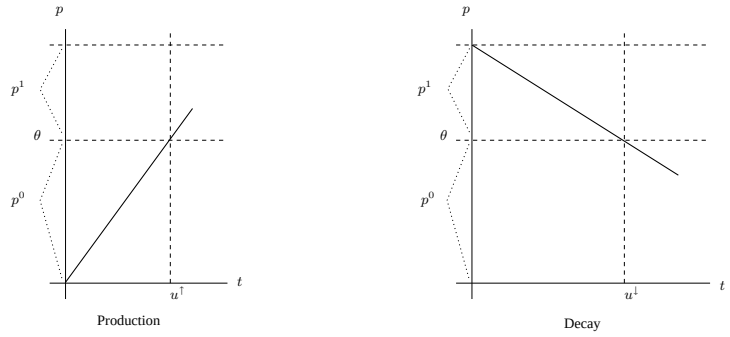


Fig. 4. An example of a continuous process which may underly the delay.

Consider, for example, the negative feedback loop of Figure 5-(a) where the presence of p turns g off and its absence turns g on. Modeling the same phenomenon as a continuous system we will have a one-dimensional vector field like the one depicted in Figure 5-(b) which points toward the equilibrium θ from both sides. In the real noisy process, it is possible that the concentration will fluctuate around the equilibrium as in Figure 5-(c), but viewed discretely this is a Zeno behavior. A positive lower bound will prevent such behaviors and will force the system to stay, quite arbitrarily, on one side of θ for some time. On the other hand, the continuous "inverse image" of an oscillation between 0 and 1 includes unrealistic behaviors such as the one of Figure 5-(d) where the system exhibits large oscillations. Our modeling strategy is to use zero lower bounds but reduce their negative effects by moving from Boolean to multi-valued abstractions as will be described in Section 4. The automaton obtained for the negative feedback loop is shown in Figure 6-(a).

[5] Approximate bounds can be derived for less trivial continuous dynamics using methods similar to those described in [HHW98,SKE00,F05].

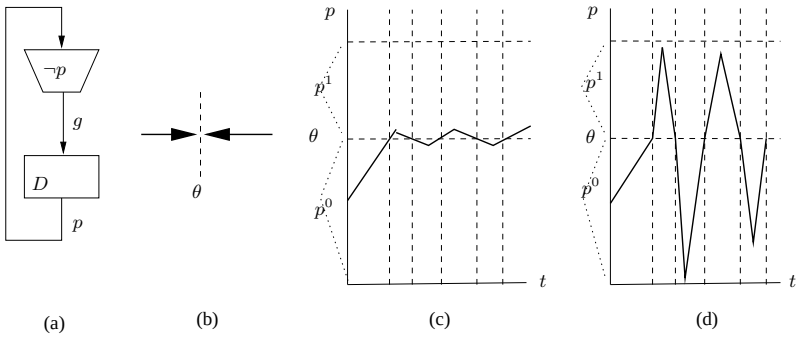


Fig. 5. (a) a negative feedback loop; (b) the corresponding one-dimensional continuous dynamical system; (c) a possible behavior of the underlying continuous dynamics; (d) a behavior which is impossible in the continuous system but is valid in the abstract timed model.

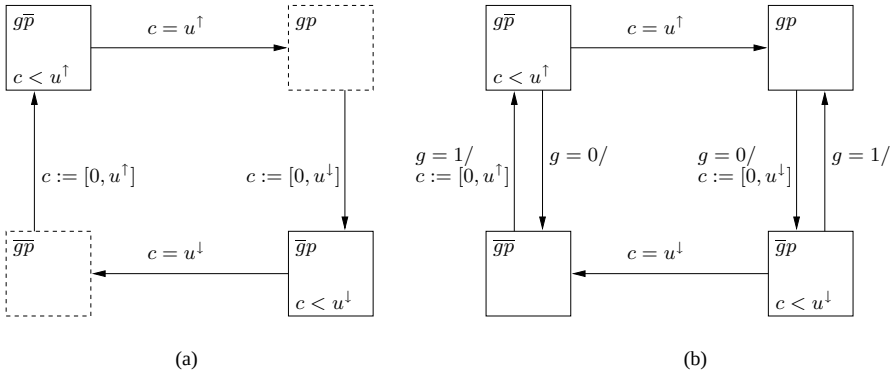


Fig. 6. (a) The automaton for the negative feedback network of Figure 5-(a). The automaton leaves states gp and $\overline{gp}$ immediately upon entrance; (b) A timed automaton model for the delay operator with zero lower bounds and the possibility of regret transitions from $g\overline{p}$ to $\overline{gp}$ and from $\overline{g}p$ to gp.

The alert reader might have noticed that we have not considered yet the case where the gene is turned off *before* the product becomes fully present. Such a situation may occur if the activation function of the gene depends on other products. In the automaton model this situation corresponds to g being turned on and then turned off at state $g\overline{p}$ before p becomes present (or the symmetric case when g is turned on at state $\overline{g}p$). This is modeled by the *regret* (or *abort*) transitions of the automaton of Figure 6-(b) that go from the excited state back to a stable state. Again, if we look at the continuous process, we cannot really know if it was aborted close to or far from the threshold, but since, using zero lower bounds, excitations are always accompanied by assignments of the form $c := [0, u]$, the timed model is conservative and covers all cases.

We construct such an automaton for each inclusion $p_i \in D_i(g_i)$ and a one-state automaton for each instantaneous relation $g_i = f_i(p_1, \ldots, p_n)$. Composing these automata we obtain a timed automaton that captures all the behaviors of the network [MP95]. We have implemented, within the IF tool suite [BGO+04], a translator from Boolean delay networks into timed automata which are then analyzed to produce the reachability graph which shows all the possible qualitative behaviors of the network when timing constraints are taken into account. These behaviors constitute a subset of what would be possible using an untimed model.

4 Multi-Valued Models

The quality of the model can be improved significantly if we refine the discrete abstraction to admit *several levels* of concentration. To this end we replace the abstract set $\{0, 1\}$ by a finite set $\{0, 1, \ldots, m-1\}$, associated with a set of thresholds $0 < \theta_1 < \theta_2 < \ldots, < \theta_{m-1} < 1$ so that state i corresponds to the interval $p^i = [\theta_i, \theta_{i+1}]$. For each direction of evolution (production and degradation) we define an upper and lower bound for moving between neighboring regions. To be more precise, let $v \xrightarrow{t} v'$ denote the fact that the underlying continuous process may go from v to v' in t time. Then the delay parameters are defined as

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

Whenever $g = 1$, the product level will move from p^i to p^{i+1} within the $[l_i^{\uparrow}, u_i^{\uparrow}]$ time interval and, likewise, it will move from p^i to p^{i-1} when $g = 0$. The extended delay operator is specified as follows:

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	$m-1$	$m-1$	$-$

(3)

The corresponding automaton is shown in Figure 7. Its upper part corresponds to states where $g = 1$ and p is increasing, and the lower part to states where $g = 0$ and

p is decreasing. The main attractive feature of this model is that it makes a distinction between entering a state via a *vertical* transition (which reverses the direction of evolution) and entering it via a horizontal transition (which is due to a monotone growth or decay process). In the latter case we *do not* need to use a zero lower bound because it is clear that the threshold was crossed *from below* (resp. above) and it will take some time between l_i and u_i to cross the next threshold in the same direction. Hence the transition from $(g, i-1)$ to (g, i) sets the clock to the interval $[0, u_i^{\uparrow} - l_i^{\uparrow}]$ while the transition from $(\overline{g}, i)$ to (g, i) sets the clock to the interval $[0, u_i^{\uparrow}]$ allowing an immediate transition from there to $(g, i+1)$. As a result, zero time cycles can now involve only states that correspond to *neighboring* levels of concentration, that is, $(g, i), (g, i+1), (\overline{g}, i+1), (\overline{g}, i)$. This way the deviation of the discrete timed model from the continuous one is reduced and tends to zero as m tends to infinity.

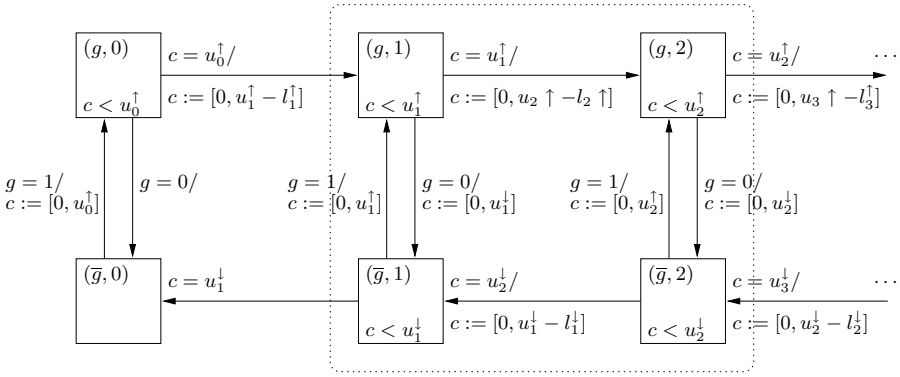


Fig. 7. A timed automaton model for the multi-valued delay operator. Zeno behaviors can now take place only among states that correspond to neighboring levels of concentration.

To complete the adaptation of the model to the multi-valued setting we replace each activation function of the form $f : \{0,1\}^n \to \{0,1\}$ by a function of the form $f' : \{0, \ldots, m-1\}^n \to \{0,1\}$ and have all the ingredients for translating the network in a timed automaton and analyzing its behaviors. Some experimental results are reported in the next section.

5 Experimental Results

We demonstrate the potential of our modeling framework and tools on several examples.

5.1 A Cross-inhibition Network

We first consider a simple model of the lysis/lysogeny decision in the λ bacteriophage, a virus that infects bacteria. Following the infection, viruses can reproduce in bacteria

in two different ways: *lysis* and *lysogeny*. In the first case, the virus multiplies in the bacterial cell and eventually kills the cell. In the second case, the genetic material of the virus integrates into the bacterial chromosome and replicates with it. A genetic regulatory network involving at least 5 viral genes is responsible for the choice between lysis and lysogeny. We study here a simple model of the core of the network that has been proposed as responsible for the lysis/lysogeny decision [TT95]. The model that we use is similar to the model used in [SB06] in their proposal to extend Thomas' model with timing information.

The network is represented by the diagram of Figure 8. It consists of two genes, *cI* and *cro*, that code for two repressor proteins, CI and Cro. More specifically, protein CI represses the expression of gene *cro*, whereas protein Cro represses the expression of gene *cI*, and at a higher concentration, the expression of its own gene. We denote the genes *cI* and *cro* by x and y and the proteins CI and Cro by X and Y.

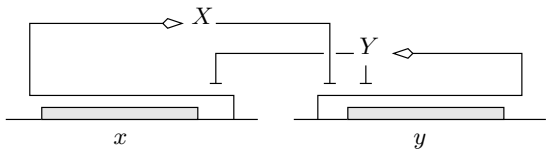


Fig. 8. A cross-inhibition network.

In our formalism, we use Boolean variables g_x and g_y for the state of the genes, and two integer variables, $p_x \in \{0, 1\}$ and $p_y \in \{0, 1, 2\}$ for the protein concentrations. The use of three concentration levels for protein Y is motivated by the fact that a moderate concentration of the protein is sufficient for the inhibition of gene x, whereas a high concentration is needed to inhibit its own gene y. The following activation functions summarize this information:

p_x	p_y	g_x	g_y
0	0	1	1
0	1	0	1
0	2	0	0
1	0	1	0
1	1	0	0
1	2	0	0

The timed automata $\mathcal{A}_x$ and $\mathcal{A}_y$ for this example are depicted in Figure 9. As in [SB06], we have used the delay intervals $[5, 10]$, but as explained in Section 3, we assumed that some changes in protein concentration can happen instantaneously in order to guarantee that the timed model is a conservative over approximation of a continuous process.

We analyzed this system using the IF toolbox [BGO+04], the successor of KRONOS [DOTY96]. The initial state of the system corresponds to the infection of a bacteria by a bacteriophage. In this state, both proteins are absent and both genes are *on*. The reachability graph generated by IF for this example is given in Figure 10-(a). Because of the interleaving semantics and the fact that changes in gene activity follow changes of protein concentration instantaneously, some states are left immediately upon entrance

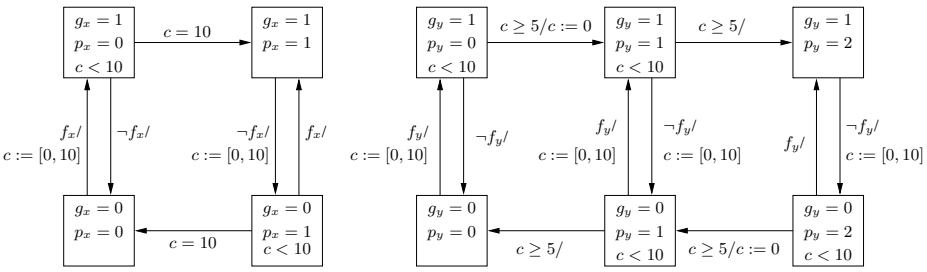


Fig. 9. The automata $\mathcal{A}_x$ and $\mathcal{A}_y$ for the cross-inhibition network.

and have no biological significance. The toolbox collapses chains of states connected by immediate transitions into single states to obtain the automaton of Figure 10-(b).

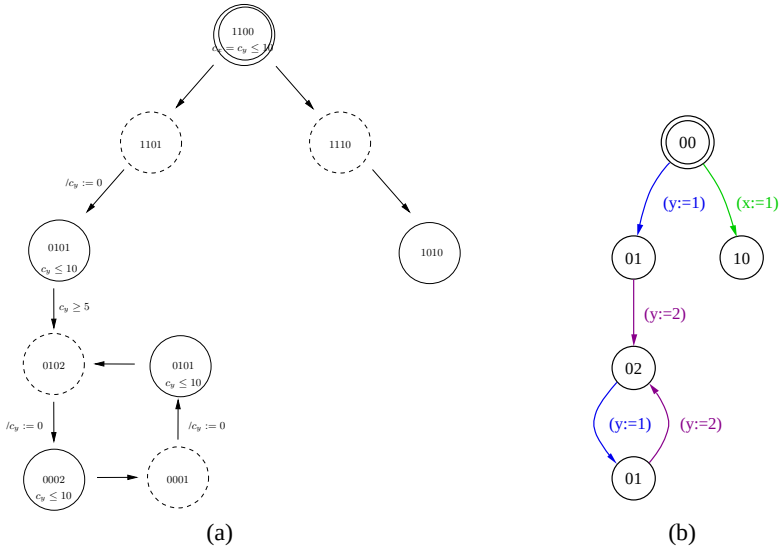


Fig. 10. (a) The reachability graph generated by IF for the cross-inhibition network. State are labeled by values of state variables (g_x, g_y, p_x, p_y). Two clocks c_x and c_y are used and the initial state is double circled. Dotted circles represent states that are left immediately. (b) The reachability graph after collapsing instantaneous transitions. State labels correspond to protein concentrations (p_x, p_y) and the transitions are labeled by changes in their values.

A manual analysis of this reachability graph reveals that the system exhibits a mutual exclusion property, in the sense that it necessarily either ends up in a state where X is present and Y is absent (state 10), or oscillates between states where X is absent and Y is present in either medium (state 01) or high (state 02) concentration. The mutual exclusion property is a well-known property of cross-inhibition networks such as the one we study. Additional interesting timing properties can be inferred from the reach-

ability graph. First, state 10 is the only state in which the system can remain forever and it can be reached from initial state 00 within time included in the interval $[0, 10]$. Secondly, it takes to the system between 5 and 20 time units to reach a state 02 having a high concentration of Y. Finally, the oscillations period is between 0 and 20 time units, that corresponds to all the cases between damped oscillations (period arbitrary close to 0) and sustained oscillations with maximum amplitude (period of 20 time units). It should be noted that the reachability graph makes a distinction between two instances of state 01, the first being reached from initial state 00 and hence having to wait at least 5 time units to move to 02, and the second reached from 02 and hence can return to it immediately. To summarize this example, we obtain the same qualitative results as in the untimed model (which suggest that this particular network is robust under delay variations), plus some additional timing information on the possible behaviors of the system.

5.2 Transcriptional Cascade in *E. coli*

As a second example, we study a transcriptional cascade of *E. coli* [HTW05] represented in Figure 11. It is made of four genes: *tetR*, *lacI*, *cI*, and *eyfp* that code respectively for three repressor proteins, TetR, LacI, and CI, and the fluorescent protein EYFP. The fluorescence of the system, due to the protein EYFP, can be measured. The system can be controlled by the addition or removal of a small diffusible molecule, aTc, in the growth media. More precisely, aTc binds to TetR and relieves the repression of *lacI*. One can check that the fluorescence of the system at steady state will be low for low aTc concentrations, and high for high aTc concentrations.

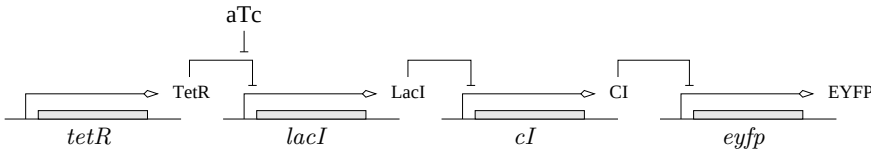


Fig. 11. A transcriptional cascade.

We have made a simple model of this system, assuming a maximal delay for protein production of 45 minutes for all proteins. Although these delays are biologically realistic they should not be considered as accurate and they are used only to demonstrate the capabilities of our tool. Using the IF toolbox, we computed the reachability graph for two different initial conditions, corresponding to high and low aTc concentrations. We obtained, respectively, graphs having 17 states and 34 transitions (see Figure 12), and 19 states and 38 transitions. The computations lasted less than one second on a standard PC. In both cases, these computations indicate that the system eventually stabilizes and always remains in a state in which the fluorescence level is consistent with what is experimentally observed: high fluorescence in presence of aTc, and low fluorescence in its absence. Manual analysis of these graphs revealed that the equilibrium state is necessarily reached in less that 6 hours. This upper bound is much larger than what has

been observed experimentally [HTW05], and is due to the fact that we have used coarse Boolean abstractions of the concentration levels. Nevertheless, proving the existence of some upper bounds, which cannot be done using untimed models, is an important step toward using such a cascade as a module in a synthetic network.

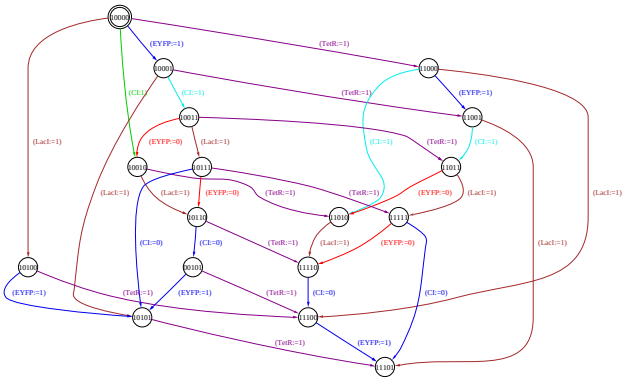


Fig. 12. The reachability graph for the transcriptional cascade in the *presence* of aTc. State labels correspond to $(p_{aTc}, p_{tetR}, p_{lacI}, p_{cI}, p_{eyfp})$. The initial state is 10000 (aTc present and EYFP absent) and the attractor state is 11101 (aTc present and EYFP present).

5.3 Nutritional Stress Response in *E. coli*

Our last example is based on a model of the nutritional stress response in *E. coli* which plays a crucial role in its survival. When confronted with a nutritional stress, this creature stops growing and enters in a dormant, resistant state. We use a simplified version of the elaborate model of this phenomenon proposed in [RJP+06]. The model consists of six genes, six proteins, and one additional variable encoding the presence or absence of nutrition. Since no timing information is available on this system we have arbitrarily used $[0, 45]$ minutes as a delay interval, just to check the feasibility of analysis.

The reachability graph, computed using the IF toolbox in less than one second, has 69 states and 209 transitions, and admits several cycles (Figure 13). However, the manual analysis of graphs of this size is less attractive and should be replaced with model checking against higher-level temporal properties that we intend to integrate into our toolbox. Nevertheless, this example shows that the exhaustive analysis of complex timed models of gene networks is feasible.

6 Discussion

We have laid down some conceptual foundations for timed modeling of genetic regulatory network and other biological processes, provided tool support for the definition and analysis of such models and demonstrated its effectiveness on several examples. In

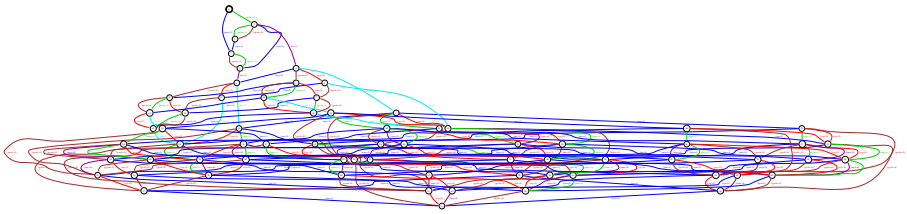


Fig. 13. The reachability graph for the nutritional stress response.

particular, we have extended the framework of [MP95] from Boolean to multi-valued domains in a clean and systematic manner which reduces the effect of Zeno behaviors on the quality of the model. We have tested the computational effectiveness of our modeling approach on several non-trivial examples.

The modeling framework can be refined further to accommodate several levels of gene activation, each with different production rates and delays. Such an extension will require a careful examination of different ways to construct functions over bounded integer domains using a combination of logical and arithmetical operations.

Finally let us not delude ourselves that after having provided modeling and tool support, all that remains is to sit and wait for biologists to submit their models for analysis. Much is still to be done in promoting this class of models among them and in orienting their experiments to yield the information required to make these models useful. We have reasons to believe that this information could be, to a certain extent, easier to obtain than what is needed for meaningful continuous models, for example from micro-array experiments with a low sampling rate. If this is the case we can hope that timed models will find a significant niche in the discrete-to-continuous spectrum of dynamical system models used for biological purposes.

References

[ADF+06] E. Asarin, T. Dang, G. Frehse, A. Girard, C. Le Guernic and O. Maler, Recent Progress in Continuous and Hybrid Reachability Analysis, *CACSD*, 2006.

[BDL+01] G. Behrmann, A. David, K.G. Larsen, O. Möller, P. Pettersson and W. Yi, UPPAAL - Present and future, *CDC'01*, 2001.

[BBM03] R. Ben Salah, M. Bozga and O. Maler, On Timing Analysis of Combinational Circuits, *FORMATS'03*, 204-219, LNCS 2791, 2003.

[BGO+04] M. Bozga, S. Graf, I. Ober, I. Ober, and J. Sifakis, The IF toolset, *SFM*, 2004.

[BJMY02] M. Bozga, H. Jianmin, O. Maler and S. Yovine, Verification of Asynchronous Circuits using Timed Automata, *ENTCS* 65, 2002.

[BMT99] M. Bozga, O. Maler and S. Tripakis, Efficient Verification of Timed Automata using Dense and Discrete Time Semantics, *CHARME'99*, 125–141, LNCS 1703, 1999.

[DOTY96] C. Daws, A. Olivero, S. Tripakis, and S. Yovine, The tool KRONOS, *Hybrid Systems III*, 208–219, LNCS 1066, 1996.

[F05] G. Frehse, PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech, *HSCC'05*, 258–273, LNCS 3414, 2005.

[HHW98] T.A Henzinger, P.-H. Ho, and H. Wong-Toi, Algorithmic Analysis of Nonlinear Hybrid Systems, *IEEE TRans. on Automatic Control* **43**, 540–554, 1998.

[HTW05] S. Hooshangi, S. Thiberge, and R. Weiss, Ultrasensitivity and Noise Propagation in a Synthetic Transcriptional Cascade, *PNAS* **102**, 3581–3586, 2005.

[K69] S.A. Kauffman, Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets, *J. of Theoretical Biology*, **22**, 437–467, 1969.

[MP95] O. Maler and A. Pnueli, Timing Analysis of Asynchronous Circuits using Timed Automata, *CHARME'95*, 189-205, LNCS 987, 1995.

[RJP+06] D. Ropers, H. de Jong, M. Page, D. Schneider, and J. Geiselmann. Qualitative Simulation of the Carbon Starvation Response in *Escherichia coli*. *BioSystems* **84**, 124–152, 2006.

[SB06] H. Siebert and A. Bockmayr, Incorporating Time Delays into the Logical Analysis of Gene Regulatory Networks, *CMSB'06*, 169–183, LNCS 4210, 2006.

[SKE00] O. Stursberg, S. Kowalewski and S. Engell, On the Generation of Timed Discrete Approximations for Continuous Systems, *Mathematical and Computer Modelling of Dynamical Systems* **6** 51–70, 2000.

[TT95] D. Thieffry and R. Thomas, Dynamical Behaviour of Biological Regulatory Networks: II. Immunity Control in Bacteriophage Lambda, *Bulletin of Mathematical Biology* **57**, 277–295, 1995.

[TD90] R .Thomas and R. D'Ari, *Biological Feedback*, CRC Press, 1990.