Approximating Continuous Systems by Timed Automata*

Oded Maler[1] and Grégory Batt[2]

[1] Verimag-UJF-CNRS, 2 Av. de Vignate, 38610 Gières, France
Oded.Maler@imag.fr
[2] INRIA Rocquencourt, 78153 Le Chesnay, France
Gregory.Batt@inria.fr

Abstract. In this work we develop a new technique for over-approximating (in the sense of timed trace inclusion) continuous dynamical systems by timed automata. This technique refines commonly-used discrete abstractions which are often too coarse to be useful. The essence of our technique is the partition of the state space into cubes and the allocation of *a clock for each dimension*. This allows us to get much better approximations of the behavior. We specialize this technique to multi-affine systems, a class of nonlinear systems of primary importance for the analysis of biochemical systems and demonstrate its applicability on an example taken from synthetic biology.

1 Introduction

Rigorous reasoning about the behavior of continuous dynamical systems has been a topic of study within various communities including qualitative physics in AI, robotics, and hybrid control systems. A more recent motivation comes from the domain of *systems biology* which, among other things, attempts to build *quantitative dynamic models* that capture the behavior of complex networks involving a large number of biochemical substances. Due to experimental limitations, such models admit a lot of uncertainty concerning parameter values and environmental conditions. Consequently, there is a lot of ongoing effort to apply methodologies used in the design of complex artificial systems, formal verification included, to analyze the implication of proposed models and assess their plausibility. The fact that biochemical models are often described as differential equations, with state variables denoting substance *concentrations* motivates the effort to adapt algorithmic verification technology (model checking) to continuous and hybrid systems in order to prove satisfaction of temporal properties by all system behaviors (trajectories) departing from a possible set of initial state and subject to a class of admissible inputs (disturbances).

One can classify various approaches to algorithmic verification of continuous and hybrid systems as using *direct* and *indirect* methods:[3]

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

[3] Another way to view this classification is between methods based on time and space discretization.

1. Direct methods work on the original dynamics of the system, starting from a set of initial states and applying a "successor" operator that computes the set of states reachable from those by following the continuous dynamics, until a fixpoint is reached (or not). For hybrid systems with a very simple continuous dynamics in each discrete state, namely, constant derivatives as in timed automata or linear hybrid automata (LHA), such successor states can be computed exactly for all future time instants [ACH+95,HHW97,F05]. The problem however still remains undecidable for most interesting classes due to the combination of such dynamics with discrete transitions [HKPV98,AMP95]. If the system admits a more complex dynamics defined by differential equations, the successors can be computed in an approximate manner using a kind of set-based numerical integration [DM98,CK98], [ABDM00,CK03,ADF+06].
2. Indirect methods (which are the subject of this paper) transform the original system model into an abstract model belonging to a simpler class, whose verification is easier and often decidable. The most commonly-used class of abstract models are, of course, finite-state automata, used extensively in the biological context [JPHG01,BRJ+05,HKI+07], but other reduction techniques have been proposed such as using timed automata to approximate continuous systems [SKE00] and LHA [OSY94], approximating continuous systems by LHA [HHW98,F05] or approximating nonlinear systems by piecewise-affine differential equations [ADG03]. The major advantage of the indirect approach is that simpler classes of models, for example finite-state automata, admit well-known model-checking algorithms, realized by numerous mature tools, while the adaptation of such techniques to systems with non-trivial continuous dynamics is much more difficult if not impossible.

Procedures for deriving such abstract models offer a tradeoff between the accuracy of the obtained model and the difficulty in deriving and analyzing it. The most straightforward approach for constructing automata from continuous systems, defined via an equation of the form $\dot{x} = f(x)$ consists of partitioning the continuous state space into rectangular cells, and defining a transition between neighboring cells if there is a trajectory of the continuous system that goes directly from one cell to another. This latter fact can be determined *locally* by evaluating f on their common boundary. While this approach guarantees a conservative over-approximation in the sense that the existence of a trajectory from x to x' in the concrete systems implies the existence of a corresponding run in the automaton, it suffers, like any abstraction technique, from "false transitivity" leading to numerous spurious behaviors, that is, abstract behaviors that do not correspond to concrete ones.

In this paper we refine this abstraction scheme by adding clocks to the automata [AD94]. The use of timed automata brings the following advantages:

1. The added clocks keep track of the progress of the trajectories along each dimension, and their values are used to constraint the dynamics of the automaton, resulting in a significant reduction of false transitivity. Moreover, the accuracy of the model can be improved indefinitely by refining the underlying grid;
2. The timed model generates timed behaviors that can be checked against *quantitative* timing properties expressed in real-time temporal logics such as MTL [Koy90] or MITL [AFH96], while this information is absent from purely discrete models;

3. The constructed models can be handled by existing verification tools for timed automata such as Uppaal [LPY97] or IF [BGM02] that can compute reachable states and, in principle, perform model checking.

The rest of the paper is organized as follows. In Section 2 we give preliminary definitions and demonstrate the problem of false transitivity. In Section 3 we show how to derive a timed automaton from a continuous dynamical system and prove that it constitutes a conservative approximation. We then present the derivation of delay bounds for the class of multi-affine systems, a class of nonlinear dynamical system used extensively in biological modeling. Section 5 reports preliminary experimental results using a prototype implementation which generates timed automata written in the IF format. A discussion of past and future work concludes the paper.

2 Preliminaries

We start this section with some definitions concerning dynamical systems, the partition of space into cubes and related geometrical concepts and notations taken from [BMP99]. To simplify notations we consider integer grids and temporal properties generated from atomic propositions of the form $x_i \geq k$ with integer k. Of course, all the results can be adapted to non-uniform grids.

We consider a dynamical system $\mathcal{S} = (X, f)$ with state space

$$X = X_1 \times \cdots \times X_n = [0, m) \times \cdots \times [0, m) \subseteq \mathbb{R}^n$$

and dynamics is defined by

$$\dot{x} = f(x) \tag{1}$$

where $f = (f^1, \ldots, f^n)$ is a well-behaving continuous function from $\mathbb{R}^n$ to itself. A trajectory of the system starting from an initial state x is a function $\xi : \mathbb{R}_{\geq 0} \to X$ such that ξ is the solution of (1) with initial condition x_0, that is, $\xi(0) = x_0$ and for every $t \geq 0$,

$$\frac{d\xi}{dt}(t) = f(\xi(t)).$$

We impose an integer grid on X by letting $V = V_1 \times \cdots \times V_n$, $V_i = \{0, \ldots, m-1\}$ and letting $\mathcal{C}(X)$ be the set of unit cubes with integer vertices which are contained in X. We use V to represent $\mathcal{C}(X)$.

Definition 1 (Cubes, Neighbors, Facets and Slices).

1. *The cube associated with a point $v = (v_1, \ldots, v_n) \in V$ is*

$$X_v = [v_1, v_1 + 1) \times \cdots \times [v_n, v_n + 1),$$

   that is, the unit cube for which v is the leftmost corner.
2. *The successor and predecessor of a vertex/cube v in the i^{th} direction are, respectively*

$$\sigma^{+i}(v_1, \ldots, v_{i-1}, v_i, \ldots, v_n) = (v_1, \ldots, v_{i-1}, v_i + 1, \ldots, v_n)$$

and

$$\sigma^{-i}(v_1, \ldots, v_{i-1}, v_i, \ldots, v_n) = (v_1, \ldots, v_{i-1}, v_i - 1, \ldots, v_n).$$

Two cubes/vertices are neighbors if one is the i-successor/predecessor of the other;

3. *The common facet between two neighboring cubes is the $(n-1)$-dimensional cube obtained by intersecting their boundaries;*
4. *The i-slice associated with an integer r is the set $X_{i,r}$ obtained by restricting X to points satisfying $r \leq x_i < r + 1$.*

Note that a unit cube X_v, $v = (v_1, \ldots, v_n)$, is an intersection of n slices:

$$X_v = \bigcap_{i=1}^{n} X_{i,v_i}.$$

Definition 2 (Grid Based Abstraction).

1. *The abstraction function $\alpha : X \to V$ maps every point to the cube it belongs to, that is, $\alpha(x) = v$ if $x \in X_v$;*
2. *The timed abstraction of a trajectory ξ is $\xi' = \alpha(\xi)$ such that for every t, $\xi'(t) = \alpha(\xi(t))$;*
3. *The untimed abstraction $\bar{\alpha}(\xi)$ of ξ is the (stutter-free) sequence of cubes appearing in $\alpha(\xi)$.*

Definition 3 (Extremal Values of f).

1. *The extremal values of f_i in a cube v are*

$$\underline{f}^i_v = \min\{f_i(x) : x \in X_v\} \quad \textit{and} \quad \overline{f}^i_v = \max\{f_i(x) : x \in X_v\}.$$

2. *The minimal absolute velocity of f_i in a cube v is*

$$\underline{\underline{f}}^i_v = \min\{|f_i(x)| : x \in X_v\}$$

3. *The extremal values of f_i on slice $X_{i,r}$ are*

$$\underline{f}_{i,r} = \min\{f_i(x) : x \in X_{i,r}\} \quad \textit{and} \quad \overline{f}_{i,r} = \max\{f_i(x) : x \in X_{i,r}\}$$

The standard way to derive a finite-state automaton from a dynamical system is summarized by the following definition.

Definition 4 (Abstraction by Automata). *The automaton $\bar{\mathcal{A}} = (V, \bar{\delta})$ is an abstraction of $\mathcal{S}$ if $\bar{\delta}$ consists of all pairs $(v, \sigma^{+i}(v))$ of cubes such that f_i admits a positive value on their common facet and all pairs $(v, \sigma^{-}(v))$ such that f_i admits a negative value on their common facet.*

Claim (Conservativism). For every trajectory ξ of $\mathcal{S}$, there is a run $\bar{\xi}$ of $\bar{\mathcal{A}}_{\mathcal{S}}$ such that $\bar{\xi} = \bar{\alpha}(\xi)$.

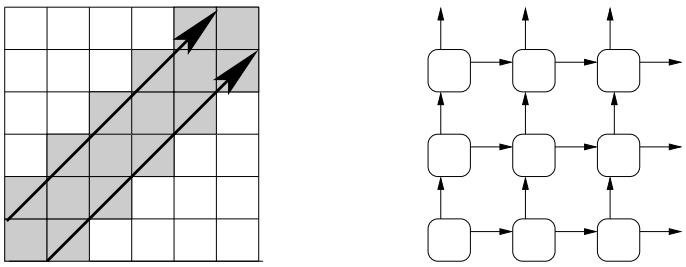

Fig. 1. (a): A simple continuous system with constant derivatives. The states reachable from the initial cube lie between the two arrows and their cube abstraction is shaded; (b) The automaton derived according to Definition 4 in which the whole state space is reachable.

This result implies that any (next-free) LTL property, generated by atoms of the form $x_i \geq k$, which is satisfied by $\bar{\mathcal{A}}_{\mathcal{S}}$ is satisfied by $\mathcal{S}$. However, as the following example shows, $\bar{\mathcal{A}}_{\mathcal{S}}$ may have so many spurious behaviors, that it might be hard to prove interesting properties based on it. Consider a system where $f = (1, 1, \ldots, 1)$ as in Figure 1-(a). Since f has a positive component for every direction everywhere, there will be a transition from each cube to each of its i-successors and the whole state space will be reachable.

As one can see, the false transitivity is due to the fact that the transition relation between neighboring cubes is computed *locally*: since it is possible to go from v to $\sigma^{+i}(v)$ and from $\sigma^{+i}(v)$ to $\sigma^{+i}(\sigma^{+i}(v))$, the automaton allows these two successive transitions to happen, ignoring timing constraints related to the fact that between these two transitions, the trajectory needs to *cross the distance* between v_i and $v_i + 1$ in direction i, a process that takes time and might be slower then the crossing in other directions. In this paper we use clocks to impose such timing constraints.

Definition 5 (Timed Automaton). *A timed automaton is a tuple $\mathcal{A} = (Q, \mathcal{C}, I, \Delta)$ where Q is a finite set of discrete states, $\mathcal{C}$ is a set of clock variables ranging over $\mathbb{R}_{\geq 0} \cup \{\perp\}$ where $\perp$ is a special symbol indicating that the clock is inactive, I is the staying condition (invariant) which assigns to every state q, a conjunction I_q of conditions of the form $c < d$ for clock c and integer d. The transition relation Δ consists of tuples of the form (q, g, ρ, q') where q and q' are discrete states, the transition guard g is a positive combination of conditions of the form $c \geq d$ or $c = \perp$, and ρ is a clock transformation defined by one or more assignments of the form $c := 0$ or $c := \perp$.*

A *configuration* of the automaton is a pair $(q, \mathbf{z})$ where $\mathbf{z}$ is a clock valuation. The behavior of a timed automaton consists of an alternation between *time progress* periods where the automaton stays in a state q and I_q continuously holds, and *discrete instantaneous transitions* guarded by clock conditions. Formally, a *step* of the automaton is one of the following:

- A time step: $(q, \mathbf{z}) \xrightarrow{t} (q, \mathbf{z} + t)$, $t \in \mathbb{R}_+$ such that $\mathbf{z} + t$ satisfies I_q, and $\mathbf{z} + t$ is the result of adding t to clocks active in $\mathbf{z}$.

– A discrete step: $(q, \mathbf{z}) \xrightarrow{\delta} (q', \mathbf{z}')$, for some transition $\delta = (q, g, \rho, q') \in \Delta$, such that $\mathbf{z}$ satisfies g and $\mathbf{z}'$ is the result of applying ρ to $\mathbf{z}$

A *run* of the automaton starting from a configuration $(q_0, \mathbf{z}_0)$ is a finite or infinite sequence of alternating time steps and discrete steps of the form

$$\xi : \quad (q_0, \mathbf{z}_0) \xrightarrow{t_1} (q_0, \mathbf{z}_0 + t_1) \xrightarrow{\delta_1} (q_1, \mathbf{z}_1) \longrightarrow \cdots$$

whose *duration* is $\sum t_i$. One can also view such a run as a function $\xi : \mathbb{R}_{\geq 0} \to Q$ with $\xi(t) = q$ if after a duration of t the run is at state[4] q.

3 From Dynamical Systems to Timed Automata

We first establish some upper bounds on the time a trajectory may stay in a cube or in a slice and lower bounds on the time that must elapse between two successive transitions in the same direction.

Claim (Cube Sojourn Bounds). A trajectory entering a cube X_v cannot stay there more that $\overline{t}_v$ time where

$$\overline{t}_v = \min\{1/\underline{f}^i_{=v} : 1 \leq i \leq n\}.$$

This implies that any cube X_v must be left in finite time unless every f_i attains a zero in it. The following definition establishes lower bounds on the time is takes a trajectory to cross a unit of distance in a positive or a negative direction based on the bounds on its derivative.

Claim (Slice Sojourn Bounds). Let ξ be a one-dimensional trajectory whose derivative in the interval $[t, t+h]$ is bounded in $[\underline{f}, \overline{f}]$. Then

$$\begin{gathered}
\xi(t+h) - \xi(t) = 1 \;\Rightarrow\; h \geq \underline{t}^+ \\
\xi(t+h) - \xi(t) = -1 \;\Rightarrow\; h \geq \underline{t}^- \\
(\forall h' \leq h\; \xi(t+h') - \xi(t) \geq -1) \;\Rightarrow\; h \leq \overline{t}^+ \\
(\forall h' \leq h\; \xi(t+h') - \xi(t) \leq -1) \;\Rightarrow\; h \leq \overline{t}^-
\end{gathered}$$

where $\underline{t}^+$, $\overline{t}^+$, $\underline{t}^-$ and $\overline{t}^-$ are computed from $[\underline{f}, \overline{f}]$ according to the following table

	$\underline{t}^+$	$\overline{t}^+$	$\underline{t}^-$	$\overline{t}^-$
$0 < \underline{f} < \overline{f}$	$1/\overline{f}$	$1/\underline{f}$	∞	∞
$\underline{f} < \overline{f} < 0$	∞	∞	$-1/\underline{f}$	$-1/\overline{f}$
$\underline{f} < 0 < \overline{f}$	$1/\overline{f}$	∞	$-1/\underline{f}$	∞

(2)

Corollary 1 (Slice Transversal). *Let $\underline{f}_{i,r}$ and $\overline{f}_{i,r}$ be the bounds for f_i in slice $X_{i,r}$ and let $\underline{t}^+_{i,r}$, $\underline{t}^-_{i,r}$, $\overline{t}^+_{i,r}$ and $\overline{t}^-_{i,r}$ be the sojourn bounds derived from them according to (2).*

[4] If one or more transitions occur at t we take $\xi(t)$ to be the state reached after the last transition.

1. *A trajectory that enters $X_{i,r}$ from the left cannot leave it from the right in time smaller then $\underline{t}^+_{i,r}$ and cannot stay in the slice more than $\overline{t}^+_{i,r}$*
2. *A trajectory that enters $X_{i,r}$ from the right cannot leave it from the left in time smaller then $\underline{t}^-_{i,r}$ and cannot stay in the slice more than $\overline{t}^-_{i,r}$.*

Based on this bounds we can now define the approximating timed automaton. Clocks z^+_i and z^-_i will be reset upon entering an i-slice from the left or from the right, respectively and will constrain further transitions in the same direction. Clock z will be reset at every transition and will be used for the invariant. We used timed automata with explicit deactivation of clocks denoted by $x := \perp$. Whenever a transition in one direction is taken, the clock in the other direction becomes inactive.

Definition 6 (Approximating TA).
Given a dynamical systems $\mathcal{S} = (X, f)$, its approximating timed automaton is $\mathcal{A}_\mathcal{S} = (V, Z, I, \Delta)$ where $Z = \{z, z^+_1, \ldots, z^+_n, z^-_1, \ldots, z^-_n\}$ is a set of clocks, I is an invariant defined for every state v as

$$I_v = z < \bar{t}_v \wedge \bigwedge_{i=1}^{n} (z^+_i < \overline{t}^+_{i,v_i}) \wedge (z^-_i < \overline{t}^-_{i,v_i})$$

with $z < \infty$ interpreted as true. *The transition relation Δ consists of the following transition types:*

$$\delta^{+i}_v : \ (v, z^+_i \geq \underline{t}^+_{i,v_i} \vee z^+_i = \perp, z^+_i := 0; z^-_i := \perp; z := 0, \sigma^{+i}(v))$$

and

$$\delta^{-i}_v : \ (v, z^-_i \geq \underline{t}^-_{i,v_i} \vee z^-_i = \perp, z^-_i := 0; z^+_i := \perp; z := 0, \sigma^{-i}(v))$$

provided that such transitions are possible in the discrete abstraction $\bar{\mathcal{A}}$.

We do not specify the initial state to provide for queries concerning different initial cubes. For every cube X_v we will use

$$Z_v = \{0\} \times [0, \underline{t}^+_{1,v_1}] \times \cdots \times [0, \underline{t}^+_{n,v_n}] \times [0, \underline{t}^-_{1,v_1}] \times \cdots \times [0, \underline{t}^-_{n,v_n}]$$

as an initial timed zone when we ask queries about trajectories starting at X_v. This way we are conservative with respect to all possible initial points in X_v which can be as close as we want to the boundary and cross as early as we want. The property of the timed automaton is summarized by the following theorem.

Theorem 1 (Neo Conservatism). *For every trajectory ξ of $\mathcal{S}$ starting from a point $x \in X_v$ there is at least one run ξ' of $\mathcal{A}_\mathcal{S}$ staring from (v, Z_v) such that $\xi' = \alpha(\xi)$.*

Proof. Note that $\xi' = \alpha(\xi)$ means that ξ' takes transitions exactly when ξ crosses grid boundaries, and that ξ' can stay in a state as long as ξ stays in a cube. We use the following auxiliary assertion that we prove by induction: for every trajectory ξ of duration t, starting from $x = (x_1, \ldots, x_n) \in X_v$ and ending in $x' = (x'_1, \ldots, x'_n) \in X_{v'}$ there is a run ξ' of $\mathcal{A}_\mathcal{S}$ starting at (v, Z_v) and ending in $(v', \mathbf{z})$ such that $\xi' = \alpha(\xi)$ and

1. The value of z is the time elapsed having entered X_v (or since time zero if we are still in the initial cube).
2. The value of each clock z_i^+ and z_i^- is equal to one of the following
   - The time elapsed since time zero if no crossing in direction i has occurred in ξ up to time t;
   - The time elapsed since the last i-crossing if it took place in the matching direction (positive for z_i^+, negative for z_i^-);
   - Otherwise, if the last i-crossing was in the opposite direction, the clock is inactive.

The proof is by induction on the number k of boundary crossings by the trajectory during the interval $[0, t]$:

- Base case: $k = 0$ and the trajectory remains in the same initial cube (Figure 2-(a)). According to Rolle's theorem each f_i has a derivative $(x'_i - x_i)/t$ in the cube v (and in each of the slices it belongs to). In other words

$$\underline{f}^i_v \leq (x'_i - x_i)/t \leq \overline{f}^i_v \quad \text{and} \quad \underline{\underline{f}}^i_v \leq |(x'_i - x_i)|/t.$$

  Taking into account that $x'_i - x_i \leq 1$ we have

$$t \leq t/(x'_i - x_i) \leq \overline{t}_{i,v_i} \quad \text{and} \quad t \leq t/|(x'_i - x_i)|/t \leq \overline{t}_v$$

  which implies that there is a run of the automaton starting at v with all clocks set to zero that will satisfy the state invariants during the whole interval $[0, t]$.
- Inductive case: assuming the claim holds for all trajectories that cross grid boundaries at most k times, we show it holds for trajectories with $k + 1$ crossings. Let the new crossing occur in dimension i and, without loss of generality, be in the positive direction. First we show that for each j the state invariant holds between the last j-crossing and the time it reached x'. Since this part of the trajectory involves a displacement of length smaller than 1 in dimension j, a reasoning similar to the base case applies (Figure 2-(b)). Concerning direction i, we need, in addition, to show that the transition guard associated with the last crossing holds. Let t' denote the time between these two crossings that occur at y and y', respectively. There are two cases:
  1. They cross in the same direction, that is, at points $y = (y_1, \ldots, y_{i-1}, r, \ldots, y_n)$ and $y' = (y'_1, \ldots, y'_{i-1}, r + 1, \ldots, y_n)$ (Figure 2-(c)). According to the inductive hypothesis the value of clock z_i^+ when the trajectory reaches y' is the time elapsed since y. Since the i-distance is 1, by Rolle's theorem there is a derivative $1/t'$ in slice $X_{i,r}$ and the transition guard on z_i^+ will be satisfied.
  2. The crossing occurs in the opposite direction (Figure 2-(d)), hence clock z_i^+ is inactive and the transition is enabled.

  The other inductive conditions concerning clock values at time t are maintained by construction. ⌟

The accuracy of the timed model can be further improved by tightening the timing constraints associated with each cube X_v. Rather then computing them based on the

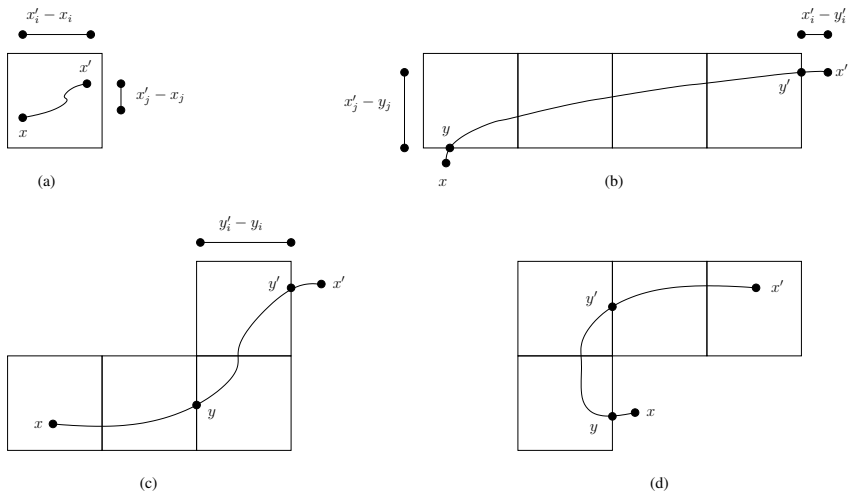

Fig. 2. (a) For a trajectory that makes no crossings, the clocks satisfy the invariant of v; (b) For a trajectory whose last crossing is in dimension i, the clocks satisfy the slice invariants associated with every j; (c) For a trajectory that crosses in direction $+i$ twice, the clocks satisfy the guard; (d) a trajectory the crosses in dimension i in two opposite directions, the guard is trivially satisfied.

extremal values of f_i in the *whole slice* X_{i,v_i} we can restrict the optimization of f_i to those cubes on the slice from which X_v is indeed reachable. For example, one can observe that if for every $j \neq i$, f_j is always positive in the slice, the only cubes in the slice from which $v = (v_1, \ldots, v_i, \ldots, v_n)$ can be reached, while staying in the slice, are those of the form $v' = (v'_1, \ldots v'_i, \ldots v'_n)$ satisfying $v'_j \leq v_j$ for every j. A systematic way to obtain such restrictions is to use the untimed abstraction $\bar{\mathcal{A}}$. Let $\pi^i(v)$ be the set of cubes v' such that there exists a run of $\bar{\mathcal{A}}$ from v' to v which stays in X_{i,v_i}. Then we can replace the slice-based bounds $\underline{t}^+_{i,r}$, $\underline{t}^-_{i,r}$, $\overline{t}^+_{i,r}$ and $\overline{t}^-_{i,r}$ by cube-specific bounds computed according to (2), but using the extremal values of f_i on $\pi^i(v)$. Note that after deriving the timed automaton, one can apply reachability analysis on the timed automaton, obtain a subset of $\pi^i(v)$, re-compute the bound according to it and so on.

As the alert reader might have noticed we have not yet specified *how* to compute the extremal values of each f_i over cubes or slices. For linear systems, extremal values are obtained on vertices while for arbitrarily nonlinear systems one can apply numerical optimization algorithms and add some error margins to the obtained results to guarantee conservativeness. In the sequel we show how the technique specializes for *multi-affine systems*, sometimes called multi-linear systems, which are based on functions whose optimization over hyperrectangles is particularly easy.

Definition 7 (Multi-Affine Functions). *A function $p : \mathbb{R}^n \to \mathbb{R}$ is multi-affine if it is a polynomial such that the maximal degree of each variable in every term is at most one. A dynamical system $\mathcal{S} = (X, f)$ with $f = (f_1, \ldots, f_n)$ is multi-affine if each f_i is a multi-affine function.*

The following result, due to Belta and Habets [BH06], provides for simple computation of of $\underline{f}$ and $\overline{f}$ over cubes and slices.

Theorem 2 (Multi-Affine functions and Rectangles). *The extremal values of a multi-affine function $p : \mathbb{R}^n \to \mathbb{R}$ over a hyperrectangle are obtained on its vertices.*

4 Implementation

Our current implementation is still in a prototype stage and it major weakness is that it works *offline*, that is, it takes a description of a piecewise[5] multi-affine dynamical systems and generates from it a timed automaton in the IF format, based on an optimized version of Definition 6. This automaton is then analyzed by the IF toolset. This implies that the timed automaton is not generated on the fly and its number of discrete states is almost the size of the grid, slightly reduced using untimed reachability analysis. A tighter integration between the approximation algorithm and the reachability computation on timed automata will allow us to restrict the generation of the TA to the reachable (under timing constraints) part of the state space.

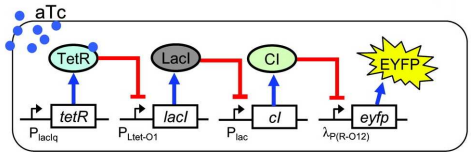

Fig. 3. The transcriptional cascade of [HTW05].

We illustrate the applicability of our approach by analyzing the timed behavior of a synthetic gene network, the cascade of transcriptional inhibitions built in *E.coli* as described in [HTW05] and illustrated in Figure 3. The cascade 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, is the measured output. The system can be controlled by the addition or removal of a small diffusible molecule aTc that binds to TetR and relieves the repression of $lacI$ in the growth media. The transient and steady-state behavior of the system was experimentally compared with that of similar, shorter cascades [HTW05]. It was found that longer cascades have a more pronounced ultrasensitive input/output responses at steady-state, but longer response times. A modifications of biological parameters that should improve the ultrasensitive response was proposed in [BYWB07], but the potential modifications of network *response times* has not been investigated.

To investigate this question we cut the 5-dimensional state space into more than 2000 cubes, from which we generate a timed automaton. We then use IF to check

[5] The extension of our results to *piecewise* multi-affine systems which are continuous on the switching boundaries is straightforward.

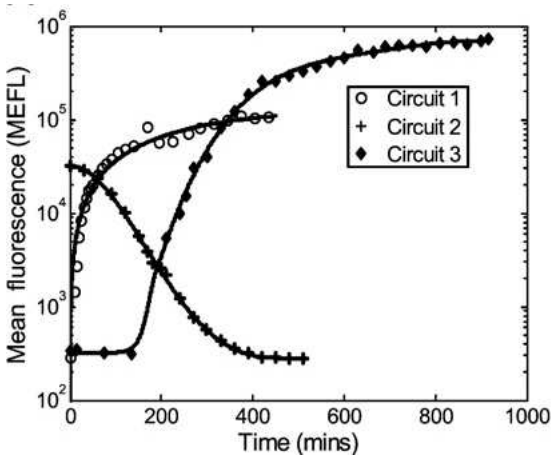

Fig. 4. Experimentally observed dealy (circuit 3).

whether a significant increase of the fluorescence of the system (from less than 500 to more than 5000 fluorescence units) is obtained in a reasonable time following the addition of aTc in the growth medium. For the original system, prior to the improvement proposed in [HTW05], our analysis shows that the required increase is guaranteed to happen in at most 2820 minutes. On the other hand, for the tuned system, we obtain a significantly smaller upper bound (1680 minutes, 40% less). This suggests that the proposed modification improves the response time of the system, in addition to improving its steady state behavior. We compare these (worst case) time bounds with observations on the actual system (Figure 4) where the fluorescence of the system reaches the target value 5000 approximatively 200 minutes after addition of aTc. This clearly reveals the conservativeness of our approach, an issue that can be addressed by using finer partitions of the state space.

5 Discussion

We have developed a technique for approximating dynamical systems by timed automata for the purpose of checking timed properties. The essence of this technique, is the use of dimension-specific clocks, in contrast with the approach of [SKE00] which uses one clock (our z) for the whole cube. These ideas are close in spirit to the rectangular hybrid automata of [HKPV98], in the sense of separating and bounding the dynamics of each dimension. In that work, the emphasis was, however, on exact decidability which required a reset (initialization) of all continuous variables when a boundary is crossed, a feature which is not useful in the continuous context.

Our approach performs reasonably well in those parts of the state space where all variables admit a monotone dynamics and the major challenge is to improve its treatment of those parts of the state space where one or more derivative changes its sign, and

hence admits a zero. In that case, the non existence of a finite upper bound can make it hard to prove eventuality properties. We also explore that extension of these techniques to even richer dynamics.

References

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, The Algorithmic Analysis of Hybrid Systems, *Theoretical Computer Science* **138**, 3-34, 1995.

[AD94] R. Alur and D.L. Dill, A Theory of Timed Automata, *Theoretical Computer Science* **126**, 183-235, 1994.

[AFH96] R. Alur, T. Feder, and T.A. Henzinger, The Benefits of Relaxing Punctuality, *Journal of the ACM* **43**, 116-146, 1996.

[ABDM00] A. Asarin, O. Bournez, T. Dang, and O. Maler, Approximate Reachability Analysis of Piecewise-linear Dynamical Systems, *HSCC'00*, LNCS 1790, 20-31, 2000.

[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.

[ADG03] E. Asarin, T. Dang, and A. Girard, Reachability Analysis of Nonlinear Systems using Conservative Approximation, *HSCC'03*, LNCS 2623, 20-35, 2003.

[AMP95] E. Asarin, O. Maler and A. Pnueli, Reachability Analysis of Dynamical Systems having Piecewise-Constant Derivatives, *Theoretical Computer Science* **138**, 35-65, 1995.

[BRJ+05] G. Batt, D. Ropers, H. de Jong, J. Geiselmann, R. Mateescu, M. Page and D. Schneider, Validation of qualitative models of genetic regulatory networks by model checking : Analysis of the nutritional stress response in *Escherichia coli*, *Bioinformatics* **21**, i19-i28, 2005.

[BYWB07] G. Batt, B. Yordanov, R. Weiss and C. Belta, Robustness analysis and tuning of synthetic gene networks, *Bioinformatics* 23, 2415-2422, 2007.

[BH06] C. Belta and L.C.G.J.M. Habets, Controlling a Class of Nonlinear Systems on Rectangles, *IEEE Trans. on Automatic Control* **51**, 1749-1759, 2006.

[BT00] O. Botchkarev and S. Tripakis, Verification of Hybrid Systems with Linear Differential Inclusions using Ellipsoidal Approximations, *HSCC'00*, LNCS 1790, 73-88, 2000.

[BMP99] O. Bournez, O. Maler and A. Pnueli, Orthogonal Polyhedra: Representation and Computation,*HSCC'99*, LNCS 1569, 46-60, 1999.

[BGM02] M. Bozga, S. Graf and L. Mounier, IF-2.0: A Validation Environment for Component-Based Real-Time Systems, *CAV'02*, LNCS 2404, 343-348, 2002.

[CK98] A. Chutinan and B.H. Krogh, Computing Polyhedral Approximations to Dynamic Flow Pipes, *CDC'98*, IEEE, 1998.

[CK03] A. Chutinan and B.H. Krogh, Computational Techniques for Hybrid System Verification, *IEEE Trans. on Automatic Control* **48**, 64-75, 2003

[DM98] T. Dang and O. Maler, Reachability Analysis via Face Lifting, *HSCC'98*, LNCS 1386, 96-109, 1998

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

[HKI+07] A. Halasz, V. Kumar, M. Imielinski, C. Belta, O. Sokolsky and S. Pathak, Analysis of Lactose Metabolism in E.coli using Reachability Analysis of Hybrid Systems, *IEE Proceedings - Systems Biology* **21**, 130-148, 2007.

[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.

[HKPV98] T.A. Henzinger, P. W. Kopke, A. Puri and P. Varaiya What's Decidable about Hybrid Automata?, *Journal of Computer and System Sciences* **57**, 94-124, 1998.

[HTW05] S. Hooshangi and S. Thiberge and R. Weiss, Ultrasensitivity and noise propagation in a synthetic transcriptional cascade, *PNAS* **102**, 3581-3586, 2005.

[HHW97] T.A Henzinger, P.-H. Ho, and H. Wong-Toi, Hytech: A Model Checker for Hybrid Systems, *Software Tools for Technology Transfer* **1**, 110-122, 1997.

[JPHG01] H. de Jong, M. Page, C. Hernandez and J. Geiselmann, Qualitative Simulation of Genetic Regulatory Networks: Method and Application, *IJCAI-01*, 67-73, 2001.

[Koy90] R. Koymans, Specifying Real-time Properties with Metric Temporal Logic, *Real-time Systems* **2**, 255-299, 1990.

[LPY97] K.G. Larsen, P. Pettersson and W. Yi, Uppaal in a Nutshell, *Software Tools for Technology Transfer* **1**, 134-152, 1997.

[OSY94] A. Olivero, J. Sifakis and S. Yovine, Using abstractions for the verification of linear hybrid systems, *CAV'94*, 81-94, LNCS 818, 1994.

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