Abstraction-Based Parameter Synthesis for Multiaffine Systems

Sergiy Bogomolov[1(✉)], Christian Schilling[2], Ezio Bartocci[3], Gregory Batt[4], Hui Kong[1], and Radu Grosu[3]

[1] IST Austria, Klosterneuburg, Austria
sergiy.bogomolov@ist.ac.at
[2] University of Freiburg, Freiburg im Breisgau, Germany
[3] Vienna University of Technology, Vienna, Austria
[4] INRIA Paris-Rocquencourt, Paris, France

Abstract. Multiaffine hybrid automata (MHA) represent a powerful formalism to model complex dynamical systems. This formalism is particularly suited for the representation of biological systems which often exhibit highly non-linear behavior. In this paper, we consider the problem of parameter identification for MHA. We present an abstraction of MHA based on linear hybrid automata, which can be analyzed by the SpaceEx model checker. This abstraction enables a precise handling of time-dependent properties. We demonstrate the potential of our approach on a model of a genetic regulatory network and a myocyte model.

1 Introduction

Hybrid automata can model systems from a wide range of real-world domains. Due to its behavioral complexity, the biological domain can particularly benefit from the expressiveness of hybrid automata [4]. However, biological models mostly have highly non-linear dynamics.

Parameter identification is the problem where we want to find a parameter set for which a given property is satisfied by the system. In the biological domain, this problem is of large importance considering the current limitations on experimental measurement techniques [17].

In this paper, we present a novel approach to solve the parameter identification problem for the class of multiaffine hybrid automata (MHA). We reduce the parameter identification problem to solving multiple *verification* problems. In short, the algorithm consists of the following steps: We partition the parameter space into a number of equivalence classes. Given an equivalence class, we show how the system behavior can be approximated with a linear hybrid automaton (LHA), which can be analyzed by the hybrid model checker SpaceEx [11]. In addition, we utilize a hierarchical search to start the analysis with coarser regions and iteratively refine the partition based on the model structure. We are also able to prune the search when we detect that our analysis will not find any parameters in a subregion. We have implemented our approach and show its potential on a genetic regulatory network and a myocyte model.

N. Piterman (Ed.): HVC 2015, LNCS 9434, pp. 19–35, 2015.
DOI: 10.1007/978-3-319-26287-1_2

Outline. The rest of the paper is organized as follows. In Sect. 2, we introduce some preliminary notions. Then we present our new approach, first the construction of the relevant parts in Sect. 3, followed by the hierarchical search procedure in Sect. 4. In Sect. 5, we evaluate the approach on two biological models. We discuss related work in Sect. 6 and conclude in Sect. 7.

2 Preliminaries

In this section, we introduce the notions used in the rest of the paper.

Multiaffine function. A multiaffine function $f : \mathbb{R}^n \to \mathbb{R}^q (n, q \in \mathbb{N})$ is a polynomial in the variables $x_1, \ldots, x_n$ with the property that the degree of f in any of the variables is less than or equal to 1 [15]. Formally, f has the following form:

$$f(x_1, \ldots, x_n) = \sum_{i_1, \ldots, i_n \in \{0,1\}} c_{i_1, \ldots, i_n} x_1^{i_1} \cdots x_n^{i_n},$$

with $c_{i_1, \ldots, i_n} \in \mathbb{R}^q$ for all $i_1, \ldots, i_n \in \{0, 1\}$ and the convention that $x_k^0 = 1$.

Hybrid automaton. A hybrid automaton (HA) [1] is a mathematical model with both continuous and discrete behavior. It is represented by the tuple $\mathcal{H} = (Loc, Var, Inv, Flow, Trans, Init)$. *Loc* is a set of discrete *locations*. *Var* is a set of real-valued variables $x_1, \ldots, x_n$. Each $\ell \in Loc$ is associated with a set of differential equations (or inclusions) $Flow(\ell)$ that defines the time-driven evolution of the continuous variables. A *state* $s \in Loc \times \mathbb{R}^n$ consists of a location and values of the continuous variables $x_1, \ldots, x_n$. The set of *discrete transitions Trans* defines how the state can jump between locations when inside the transition's *guard* set. The system can remain in a location ℓ while the state is inside the *invariant* set $Inv(\ell)$. All behavior originates from the set of *initial states Init*. A *trajectory* is a function which defines the state of the HA for every time moment. In the verification setting, we are interested in whether there exists a trajectory from the set *Init* to a set *Bad* which defines the *bad states* to be avoided.

Let $x(t) \in \mathbb{R}^n$ denote the values of the continuous variables at time t. We consider continuous dynamics *Flow* of the following two forms. If $\dot{x}(t) = f(x, t)$ where $f(x, t)$ is a multiaffine function, then the HA is called a *multiaffine hybrid automaton* (MHA). If $\dot{x}(t) \in P$ where P is a polytope, then the HA is called a *linear hybrid automaton* (LHA). We always consider *convex* polytopes and omit the dependence of f on t in what follows.

Genetic regulatory network. A genetic regulatory network [5] is defined by the dynamics of the following form:

$$\dot{x}_i = f_i(x, p) = \sum_{j \in P_i} \kappa_{ij}\, r_{ij}^P(x) - \sum_{j \in D_i} \gamma_{ij}\, r_{ij}^D(x)\, x_i, \quad i = 1, \ldots, n \tag{1}$$

Here x_i is the i-th component of the state vector $x \in \mathcal{X} \subset \mathbb{R}^n$. P_i and D_i are sets of indices. κ_{ij} and γ_{ij} are production and degradation rate parameters, respectively. We assume that some parameters are uncertain, i.e., are defined on

proper intervals. We denote the number of uncertain parameters by m. Therefore, the system is parametrized by the vector $p = (p_1, ..., p_m) \in \mathcal{D}$, where $\mathcal{D}$ is the hyper-rectangular domain of uncertain parameters.

The terms r_{ij} are continuous piecewise-multiaffine functions arising from products of ramp functions r^+ and r^- of the form shown in Fig. 1(a). Each r_{ij} captures the combined impact of several regulatory proteins in the sets P_i and D_i, respectively, on the control of the production or degradation of protein i. Assuming protein i does not regulate its own degradation, i.e., x_i does not occur in $r_{ij}^D(x)$ for $j \in D_i$, function $f = (f_1, \ldots, f_n)$ is multiaffine in x and affine in p.

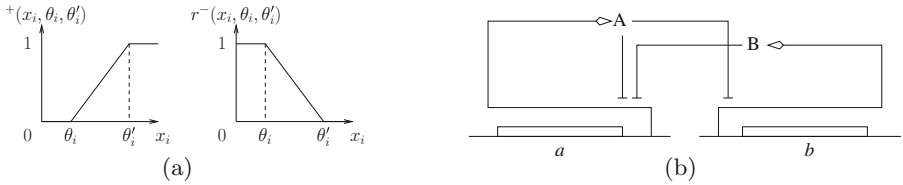

Fig. 1. (a) Ramp functions r^+ and r^-. (b) The two-genes network model.

We note that the ramp functions induce a partition of the state space $\mathcal{X}$ into a grid of hyper-rectangular regions H. The values of the separating hyperplanes are called *thresholds* θ. Let $\bar{\theta}_i$ be the number of thresholds in dimension i.

Definition 1. *Let* $\mathfrak{H} := \{H_c \mid c = (c_1, \ldots, c_n), c_i \in \{1, \ldots, \bar{\theta}_i - 1\}, i = 1, \ldots, n\}$ *be the set of hyper-rectangles* H_c *with coordinates* c *in the grid. Furthermore, let* $coord : \mathfrak{H} \to \prod_{i=1}^{n} \{1, \ldots, \bar{\theta}_i - 1\}$ *map hyper-rectangle* H_c *to its coordinate* c.

Problem statement. A property φ specifies the desired system behavior. In this paper, we consider properties of the form $R_{Init} \to \neg\Diamond R_{Bad}$ where the region R_{Init} denotes the *initial* system states and the region R_{Bad} denotes the states to be avoided. Note that φ belongs to the class of *safety* properties. Our goal is to identify a subset of the parameter domain $\mathcal{D}$ which ensures that the property φ holds for a given MHA, i.e., R_{Bad} is avoided when starting in R_{Init}.

Note that an MHA provides a semantically equivalent representation of the dynamics (1). We consider an abstraction of a parametric MHA $M(p)$ to an infinite transition system.

Definition 2. *Let* H_x *be the hyper-rectangle strictly containing* x. *Given a parameter* p *and an MHA* $M(p)$, *the* embedding transition system $T_M(p)$ *is defined as follows. The states* $\mathcal{X}$ *are the same as the continuous states of* $M(p)$. *There is a transition* $x \to x'$ *iff: (1)* H_x *and* $H_{x'}$ *are either equal or adjacent. (2) There is a solution* ξ *of* (1) *and time points* $t_0 < t_1$ *such that* $\xi(t_0) = x$, $\xi(t_1) = x'$ *and for all* t *in* $[t_0, t_1]$, $\xi(t)$ *stays within* H_x *or* $H_{x'}$.

Almost all trajectories in $M(p)$ are represented in $T_M(p)$. The exception are trajectories not passing through a common facet of two hyper-rectangles. In what follows, we consider the representation of the system $T_M(p)$.

Example (two-genes network). In the following, we illustrate our approach on the *two-genes network* [6] (also called *toggle switch* or *cross-inhibition network*).

$$\dot{x}_a = \kappa_a \cdot r^-(x_a, \theta_a^4, \theta_a^5) \cdot r^-(x_b, \theta_b^2, \theta_b^3) - \gamma_a x_a \qquad \kappa_a \in [0, 30], \gamma_a = 1$$
$$\dot{x}_b = \kappa_b \cdot r^-(x_a, \theta_a^2, \theta_a^3) - \gamma_b x_b \qquad \kappa_b \in [0, 40], \gamma_b = 2$$
$$(\theta_a^1, \theta_a^2, \theta_a^3, \theta_a^4, \theta_a^5, \theta_a^6) = (0, 8, 12, 18, 22, 30) \qquad (\theta_b^1, \theta_b^2, \theta_b^3, \theta_b^4) = (0, 8, 12, 20)$$

Here x_a and x_b define the concentrations of the proteins A and B, respectively. The uncertain parameters κ_a and κ_b define the range of their production rates in the given intervals. As Fig. 1(b) shows, protein A inhibits the production of both proteins A and B, while protein B only inhibits the production of protein A. We are interested in checking whether the protein concentrations cannot reach some specific threshold values when starting in a given initial region.

In addition, we consider an extended version of the dynamics (1) which features a *stimulus*. The stimulus is a time-dependent function which models an external influence on the system.

Example (two-genes network with stimulus). We extend the previous example with the first equation now featuring some stimulus u:

$$\dot{x}_a = \kappa_a \cdot r^-(x_a, \theta_a^4, \theta_a^5) \cdot (1 - r^+(x_b, \theta_b^2, \theta_b^3) \cdot (1 - u)) - x_a$$
$$\dot{u} = r^-(t, t_2, t_3) \qquad (t_1, t_2, t_3, t_4) = (0, 0.29, 0.3, 1)$$

We use a stimulus which is 1 at the beginning up until 0.29 ms and then drops linearly to 0 within 0.01 ms, expressed by the ramp function of time $r^-(t, 0.29, 0.3)$. This stimulus regulates the production of the protein A together with the protein B. The term $(1 - r^+(x_b, \theta_b^1, \theta_b^2) \cdot (1 - u))$ encodes the logical formula $\neg(x_b \wedge \neg u)$, which is equivalent to $\neg x_b \vee u$. Thus, this term contributes to the production of the protein A whenever the protein B is absent or the stimulus is present. Since the stimulus is time-dependent and decreasing, this means that the inhibitory effect of the protein B is only relevant for the production of the protein A after 0.29 ms.

3 Abstraction of MHA

In this section, we first introduce an LHA $L_M(p)$ which overapproximates the behavior of the transition system $T_M(p)$ for a particular parameter value $p \in \mathcal{D}$. In order to provide efficient exploration of the parameter state space, we then lift this definition to parameter sets. We use the two-genes model as a running example throughout this section.

3.1 Pointwise LHA Abstraction

State space and invariants. We use the same continuous variables for $L_M(p)$ as in the MHA $M(p)$. Thresholds θ partition the continuous state space into a grid of hyper-rectangular regions, which naturally induces a discrete structure of the LHA and location invariants. In particular, we map every hyper-rectangular region to a location in the LHA and use the bounds on the regions as invariants.

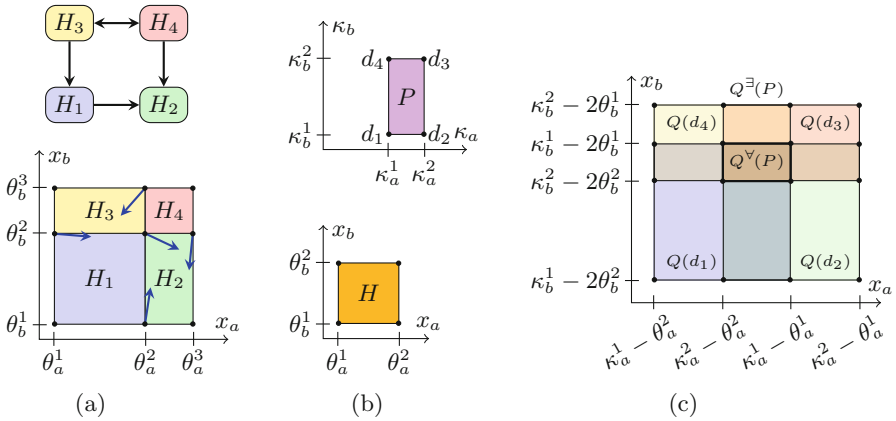

Fig. 2. (a) From state space partition to transition system. Example for $\kappa_a = 10$ and $\kappa_b = 15$ (arrows normalized). (b–c) Flow computation for $P = [8, 12] \times [15, 20]$ (Color figure online).

Example. In the two-genes example we get a 2D-partition by the planes at $x_a = \theta_a^i$ and $x_b = \theta_b^j$. Parts of the state space and the associated locations are shown in Fig. 2(a). The invariant of location H_3 is $\theta_a^1 \leq x_1 \leq \theta_a^2 \wedge \theta_b^2 \leq x_2 \leq \theta_b^3$.

Discrete transitions. We use the quotient of $T_M(p)$ with respect to the state space partition to define the discrete transitions.

Definition 3. *Let H_ℓ and $H_{\ell'}$ be the regions associated with the location ℓ and ℓ', respectively. $L_M(p)$ has a transition from location ℓ to ℓ' if*

- *ℓ and ℓ' are adjacent,*
- *there is a solution ξ of (1) and time points $t_0 < t_1 < t_2$ such that $\xi(t) \in Inv(\ell)$ for all t in $[t_0, t_1]$, and $\xi(t) \in Inv(\ell')$ for all t in $[t_1, t_2]$.*

We do not add any guards on the transitions as the chosen invariants already ensure that a transition between two locations can only be taken on the common facet of two adjacent regions H_ℓ and $H_{\ell'}$.

In order to effectively construct the transitions, we use the facts that the dynamics $f(x, p)$ are multiaffine in x and we consider only hyper-rectangular regions for the locations in $L_M(p)$. In the following, let *hull* denote the convex hull operator.

Theorem 1. *[7] Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be a multiaffine function and $H \subset \mathbb{R}^n$ be a hyper-rectangle with corner set C_H. Then*

$$f(H) \subseteq hull(\{f(v) \mid v \in C_H\}).$$

Intuitively, this theorem says that the behavior of f inside a hyper-rectangle H is completely determined by the behavior of f in the corners of H. As a consequence, the following proposition along the lines of a similar proposition for Kripke structures [5] can be proven.

Proposition 1. *$L_M(p)$ has a transition from location ℓ to ℓ' associated with hyper-rectangles H_ℓ and $H_{\ell'}$ only if the projection of $f(x,p)$ on the $H_\ell \to H_{\ell'}$ direction is positive in at least one corner of the facet separating H_ℓ from $H_{\ell'}$.*

Direction and strength of the derivative $\dot{x}_i = f_i(x,p)$ in a corner v of a hyper-rectangle depends linearly on parameter vector p. As a consequence, $f_i(v,p) = 0$ is the hyperplane separating parameter values p where $\dot{x}_i$ is positive from the ones where $\dot{x}_i$ is negative. Thus, Proposition 1 allows us to construct the transitions of $L_M(p)$ based on the sign of the function f at the vertices of the hyper-rectangles.

Example. The transitions for the excerpt shown in Fig. 2(a) are determined by the direction of the derivatives in the corners (shown in blue). For instance, we add a transition from H_1 to H_2 because there is a corner, e.g., (θ_a^2, θ_b^1), which point to this direction, but there is no transition from H_2 to H_1.

Continuous flows. For computing the flows of the LHA we again use the multi-affine dependence of $f(x,p)$ on x and the affine dependence on p.

For a fixed hyper-rectangle H with corner set C_H and a fixed parameter vector p, by Theorem 1 we know that $f(x,p)$ is included in the convex hull of the hyper-rectangle corners $v \in C_H$. Therefore, we can bound the flow of $L_M(p)$ by a polytope, i.e., the dynamics can be represented in the form of differential inclusion.

Definition 4. *The flow of $L_M(p)$ is defined as $Q(p) := hull(\{f(v,p) \mid v \in C_H\})$.*

3.2 Set-Based LHA Abstraction

In order to handle infinite sets of parameters, we lift the pointwise definition of $L_M(p)$ to sets of parameters. In particular, given a parameter polytope P, we introduce an LHA $L_M^{\exists}(P)$ which overapproximates the behavior of $L_M(p)$ for all $p \in P$. Therefore, if $L_M^{\exists}(P)$ satisfies a property φ, we can conclude that P is a valid parameter set. Otherwise, we partition the parameter set P into two subsets P_1 and P_2 and proceed with their analysis. In order to prune the parts of the parameter space where our analysis will not provide any valid parameters, we introduce a further LHA called $L_M^{\forall}(P)$ which underapproximates the LHA $L_M(p)$ for the parameter class P. In the following, we assume a parameter $p \in P$.

State space and invariants. We use the same state space and invariants for both $L_M^{\exists}(P)$ and $L_M^{\forall}(P)$ as defined for $L_M(p)$.

Discrete transitions. The theorem below provides an effective way to compute the image of $f(v,p)$ for a particular state space corner v and a parameter vector $p \in P$.

Theorem 2. *[14] Let $f : \mathbb{R}^m \to \mathbb{R}^m$ be an affine function and $P \subset \mathbb{R}^m$ be a convex polytope with corner set C_P. Then*

$$f(P) = hull(\{f(d) \mid d \in C_P\}).$$

Now, $L_M^{\exists}(P)$ has a transition from location ℓ to ℓ' if there is a transition from ℓ to ℓ' in $L_M(p)$ for *some* $p \in P$. Analogously, $L_M^{\forall}(P)$ has a transition from location ℓ to ℓ' if there is a transition from ℓ to ℓ' in $L_M(p)$ for *all* $p \in P$.

Definition 5. *Let $c := coord(H)$ and $c' := coord(H')$ be the coordinates of two adjacent hyper-rectangles. Furthermore, let $V := C_H \cap C_{H'}$ be the corners on the separating facet and let ℓ and ℓ' be the locations associated to H and H', respectively.*

We define $g(\ell,\ell') := \bigcup_{v \in V} \{p \in P \mid f_i(v,p) \cdot (c'_i - c_i) > 0\}$, where $i \in \{1,\dots,n\}$ such that $c'_i - c_i \neq 0$. A transition $\ell \to \ell'$ belongs to the LHA

- *$L_M^{\exists}(P)$ if $g(\ell,\ell') \neq \emptyset$, and to*
- *$L_M^{\forall}(P)$ if $g(\ell,\ell') = P$.*

Note that Theorem 2 allows us to construct the parameter set satisfying the constraint $f_i(v,p) \cdot (c'_i - c_i) > 0$ by only considering the vertices of P. The term $c'_i - c_i = \pm 1$ is used to express the direction of the transition. The construction uses the union operation and test for equality and emptiness for polytopes.

Continuous flows. For $L_M^{\exists}(P)$ to be an overapproximation of all $L_M(p)$ and $L_M^{\forall}(P)$ to be an underapproximation, the tightest definition we can find is the union and intersection of all $Q(p)$, respectively. Let $Q^*_{\exists}(P) := \bigcup_{p \in P} Q(p)$ and $Q^*_{\forall}(P) := \bigcap_{p \in P} Q(p)$. Computing $Q^*_{\exists}(P)$ and $Q^*_{\forall}(P)$ is, however, infeasible. Therefore, similarly to the transition construction, we propose an approximation which relies on the values of the derivatives in the corners of state space partitions H and parameter sets P.

Theorem 3. *Let $f(x,p) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ with $x \in \mathbb{R}^n$ and $p \in \mathbb{R}^m$ be a multiaffine function which is affine in p, $H \subset \mathbb{R}^n$ be a hyper-rectangle with a corner set C_H and $P \subset \mathbb{R}^m$ be a convex polytope with a corner set C_P. Then the following holds:*

- $\bigcup_{p \in P} Q(p) \subseteq hull\left(\bigcup_{d \in C_P} Q(d)\right)$,
- $\bigcap_{p \in P} Q(p) \subseteq \bigcap_{d \in C_P} Q(d)$.

Note that the left-hand sides are $Q^*_{\exists}(P)$ and $Q^*_{\forall}(P)$, respectively. Based on this theorem, we define the flows in the following way.

Definition 6. *The flow of $L_M^{\exists}(P)$ is defined as $Q_{\exists}(P) := hull\left(\bigcup_{d \in C_P} Q(d)\right)$. The flow of $L_M^{\forall}(P)$ is defined as $Q_{\forall}(P) := \bigcap_{d \in C_P} Q(d)$.*

We obtain an algorithm for computing $Q_{\exists}(P)$ and $Q_{\forall}(P)$ by first traversing all vertices $v \in C_H$ and $d \in C_P$ and collecting $f(v,d)$ in order to compute the polytope $Q(d)$. In the end, we take the finite union and intersection of those polytopes. Note that similar to $L_M(p)$ we end up with LHA whose dynamics are defined by differential inclusions.

The following proposition relates the flow representations we have introduced.

Proposition 2. *The sets $Q(p)$, $Q^*_{\exists}(P)$, $Q^*_{\forall}(P)$, $Q_{\exists}(P)$ and $Q_{\forall}(P)$ are related as follows.*

- $Q^*_{\forall}(P) \subseteq Q_{\forall}(P)$.
- $Q^*_{\forall}(P) \subseteq Q(p) \subseteq Q^*_{\exists}(P) \subseteq Q_{\exists}(P)$ *for all* $p \in P$.

Thus, $Q_{\exists}(P)$ is indeed an overapproximation of $L_M(p)$ as required. However, while $Q^*_{\forall}(P)$ is an underapproximation of $L_M(p)$, $Q_{\forall}(P)$ is not necessarily an underapproximation of $L_M(p)$ for $p \in P$. We discuss this issue in the next section.

We define the automaton $L^{\exists *}_M(P)$ by replacing $Q_{\exists}(P)$ with $Q^*_{\exists}(P)$ in the continuous flow of the automaton $L^{\exists}_M(P)$. We derive $L^{\forall *}_M(P)$ from $L^{\forall}_M(P)$ in the analogous way, i.e., by replacing $Q_{\forall}(P)$ with $Q^*_{\forall}(P)$.

Example. Consider the state space rectangle H and the parameter space rectangle P in Fig. 2(b). Recall that the state equation $\dot{x} = f(x,p)$ is given as

$$\dot{x}_a = f_a(x,p) = \kappa_a - x_a \qquad \dot{x}_b = f_b(x,p) = \kappa_b - 2x_b.$$

Hence the dynamics $f(v,d)$, for v and d in the corner sets C_H and C_P of H and P, respectively, are of the form $(\kappa^i_a - \theta^j_a, \quad \kappa^k_b - 2\theta^\ell_b)$.

Let the corners of the parameter space rectangle P be denoted in anti-clockwise order as d_1, d_2, d_3 and d_4. Now construct the state space rectangles $Q(d_1)$, $Q(d_2)$, $Q(d_3)$ and $Q(d_4)$. The intersection of all these rectangles results in the rectangle $Q^{\forall}(P)$, while the union is the rectangle $Q^{\exists}(P)$. Since it is already convex, *hull* is the identity operation in this case. The results are visualized in Fig. 2(c).

We observe that, by construction, $L_M(p)$ is a conservative abstraction of $T_M(p)$, and $L^{\exists}_M(P)$ is a conservative abstraction of $L_M(p)$, i.e., if $L^{\exists}_M(P)$ satisfies a safety property, then so does $T_M(p)$. This fact ensures the soundness of our approach.

Proposition 3. $L^{\exists}_M(P)$ *is an overapproximation of* $T_M(p)$ *for any* $p \in P$.

4 Hierarchical Parameter Search

In this section, we first show that by using sampling techniques we can compute the automaton $L^{\forall *}_M(P)$ with arbitrary precision. Afterwards, in order to leverage different levels of abstractions during the parameter space exploration, we introduce a discrete abstraction of MHA. Finally, we describe an abstraction-based parameter search procedure which explores the parameter domain in a hierarchical fashion.

4.1 Computation of Underapproximative Abstractions

As outlined in the previous section, the role of the underapproximation $L_M^{\forall *}(P)$ is to detect parameter regions which our approach cannot classify as valid ones. The LHA $L_M^{\forall}(P)$, an overapproximated version of $L_M^{\forall *}(P)$, does not generally underapproximate $L_M(p)$ for all $p \in P$. The reason is that the flows $Q_\forall(P)$ do not necessarily underapproximate the flows $Q(p)$ of all $L_M(p)$. In particular, as we take an intersection only over corner points of a considered parameter set, there might exist some $p \in P$ such that $Q_\forall(P) \not\subseteq Q(p)$.

Note that as we only use $L_M^{\forall}(P)$ for pruning purposes, soundness of our approach is not affected by imprecision. We might at most ignore some parameter region which could have been classified as valid by our approach. Still, in order to improve the precision of $L_M^{\forall}(P)$, we can randomly sample parameter vectors $p \in P$ and consider the intersection of $Q(p)$ with $Q_\forall(P)$.

The following theorem describes the possible improvements by sampling. For notational convenience, let $\|\cdot\|$ denote the Euclidean norm, ∂S denote the border of a closed set S, $d(t, S) := \min_{s \in S} \|t - s\|$ denote the distance of t to the border of S, and $Ker_\epsilon(S) := \{s \in S \mid d(s, \partial S) \geq \epsilon\}$.

Theorem 4. *Let $f(x, p) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ with $x \in \mathbb{R}^n$ and $p \in \mathbb{R}^m$ be a multiaffine function which is affine in p, $H \subset \mathbb{R}^n$ be a hyper-rectangle and $P \subset \mathbb{R}^m$ be a convex polytope. Then the following formulae hold:*

$$\forall p, p' \in P. \lim_{\|p - p'\| \to 0} Q(p) = Q(p') \tag{2}$$

$$\begin{aligned} &\forall p \in P, \epsilon > 0. \exists \delta > 0. \forall p' \in P. \|p - p'\| < \delta \\ &\quad \implies Ker_{2\epsilon}(Q(p)) \subseteq Q(p) \cap Q(p') \wedge Ker_{2\epsilon}(Q(p')) \subseteq Q(p) \cap Q(p') \end{aligned} \tag{3}$$

This theorem asserts that if we sample sufficiently many points in a uniform way in parameter space, then we can approximate $L_M^{\forall *}(P)$ infinitely closely for non-degenerate cases, i.e., when $Ker_{2\epsilon}(Q(p)) \neq \emptyset$ for all $p \in P$. However, for practical purposes, sampling can clearly shrink the overapproximation even in degenerate cases.

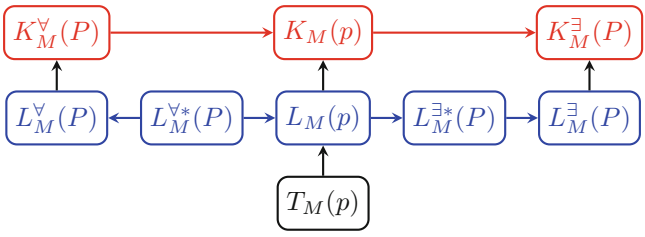

Fig. 3. Relations of the systems presented in this paper. Let $p \in P$. An arrow $S_1 \longrightarrow S_2$ indicates that S_2 is an overapproximation of S_1.

Figure 3 shows the relationship between the systems considered in this paper.

4.2 Discrete Abstraction of MHA

For later discussion, it is useful to define the induced Kripke structures (KS) of the LHA $L_M(p)$, $L_M^{\exists}(P)$ and $L_M^{\forall}(P)$. Basically, we drop the continuous behavior and map initial and bad states to the locations with non-empty intersection.

Definition 7. *Given an LHA $\mathcal{H}$ with set of locations Loc, let S be a set of discrete states with $|S| = |Loc|$ and let $disc : Loc \to S$ be a bijection which maps every location to a discrete state. In addition, let Bad be the bad states of $\mathcal{H}$.*

The pair $(\mathcal{H}, Bad)$ induces a Kripke structure *$\mathcal{K} = (S, S_0, S_B, T)$, where $S = \{disc(\ell) \mid \ell \in Loc\}$ is the set of states, $S_0 = \{disc(\ell) \mid inv(\ell) \cap Init \neq \emptyset\}$ is the set of initial states, $S_B = \{disc(\ell) \mid inv(\ell) \cap Bad \neq \emptyset\}$ are the bad states, and $T = \{disc(\ell) \to_{\mathcal{K}} disc(\ell') \mid \ell \to_{\mathcal{H}} \ell' \in Trans\}$ is the set of transitions.*

We denote by $K_M(p)$, $K_M^{\exists}(P)$, $K_M^{\forall}(P)$ the Kripke structures induced by $L_M(p)$, $L_M^{\exists}(P)$, $L_M^{\forall}(P)$, respectively. Clearly, the induced Kripke structure is a conservative abstraction of the LHA as it allows for additional trajectories to the bad states for two reasons. The behavior in the states is unconstrained due to the absence of flows, and the initial and bad states of the Kripke structure overapproximate their LHA counterparts.

Proposition 4. *$K_M(p)$ ($K_M^{\exists}(P)$, $K_M^{\forall}(P)$, respectively) is an overapproximation of $L_M(p)$ ($L_M^{\exists}(P)$, $L_M^{\forall}(P)$, respectively) for any $p \in \mathcal{D}$ ($P \subseteq \mathcal{D}$, respectively).*

We incorporate the Kripke structures into our approach in the following way. Whenever we analyze an LHA, we first analyze the respective KS. If the KS *satisfies* the given property, we skip the LHA analysis. This is justified because by Proposition 4 we know that the respective LHA will also satisfy the property. In this way, we improve analysis performance as the reachability problem for LHA is computationally harder to solve. We note that the construction of the KS does not impose any further computational efforts since we need to construct the locations and transitions for the LHA anyway.

4.3 Parameter Identification

Given an MHA M with $\dot{x} = f(x, p)$, a safety property φ and the domain of uncertain parameters $\mathcal{D}$, we explore the parameter space in a hierarchical way. Recall from the construction of the transitions that the constraints $f(x, p) = 0$ are the separating hyperplanes responsible for adding transitions. Based on those constraints, the instantiation of $f(x, p)$ in every corner v of the state space leads to a parameter space partition into polytopes.

We now explain the algorithm with the help of the pseudocode given in Fig. 4.

In a preprocessing step, the algorithm examines the corners of the state space partition and collects the constraints Ψ over the parameters in the function ***CollectConstraintsList*** (line 3) such that $f(x, p) = 0$.

```
1  ParameterIdentification(M, φ, D)
2  % M: MHA, φ: property, D: uncertain parameters
3    Ψ := CollectConstraintsList(M, D);
4    global V := ∅;
5    Explore(ε, Ψ, M, φ, D, ⊤); % start at root (ε = empty list)
6    return V; % found valid parameters
```

```
6  Explore(CL, Ψ, M, φ, D, b) % CL: list of current constraints
7    P := PolytopeFromConstraintsList(CL, D);
8    K^∃_M, K^∀_M, L^∃_M, L^∀_M = ConstructSystems(M, P);
9    if (b ∧ ¬Reach(K^∃_M, φ))
10     V := V ∪ P; return; % valid set found by the KS analysis
11   elseif (¬Reach(L^∃_M, φ))
12     V := V ∪ P; return; % valid set found by the LHA analysis
13   elseif (¬b ∨ Reach(K^∀_M, φ))
14     b := ⊥; % no future valid sets for the KS analysis
15     if (Reach(L^∀_M, φ))
16       return; % no future valid sets for the LHA analysis
17   % partition P and descend to child nodes in the search tree
18   c := first(Ψ); Ψ := rest(Ψ);
19   Explore(concatenate(CL, c ≥ 0), Ψ, M, φ, D, b);
20   Explore(concatenate(CL, c ≤ 0), Ψ, M, φ, D, b);
```

Fig. 4. The algorithm in pseudocode.

Next, the algorithm moves on to the function ***Explore*** (line 5) which actually implements the search in the parameter space. This function successively builds a number of abstractions of the MHA for a considered parameter set in order to find valid subsets. It takes a list CL of constraints which encodes hyperplanes used to define the current parameter set. We initially call the function with $CL = \varepsilon$ as we first consider the whole parameter space. Now we look at the function ***Explore*** in more detail.

We start by calling the function ***PolytopeFromConstraintsList*** (line 7) which builds a parameter polytope P based on the provided list CL of constraints over parameters and the parameter space domain $\mathcal{D}$. In line 8 we compute the KS $K^{\exists}_M(P)$, $K^{\forall}_M(P)$ and the LHA $L^{\exists}_M(P)$, $L^{\forall}_M(P)$. Note that on the implementation level we compute them only *on demand*. The computed approximations are analyzed in the following way:

1. If the property φ holds for the KS $K^{\exists}_M(P)$ already (line 9), we conclude that the current parameter set P is valid. Therefore, we add P to the set of valid parameters V and stop considering the current branch in the search tree (line 10).
2. If the discrete abstraction $K^{\exists}_M(P)$ was too coarse to prove the validity of P, we continue with the finer analysis using $L^{\exists}_M(P)$ (line 11). Similar to step 1, in the case of property satisfaction we add P to the valid parameters (line 12).

3. If the parameter set validity has not been shown up to now, we proceed to the pruning phase by considering $K_M^{\forall}(P)$ and $L_M^{\forall}(P)$. If both of them *violate* the property φ, we prune the search tree as we expect that no valid parameter sets can be found for any subset of P.
   Note that due to efficiency reasons we first analyze $K_M^{\forall}(P)$ (line 13) and move on to $L_M^{\forall}(P)$ (line 15) only if the KS is not safe with respect to the property φ. If $L_M^{\forall}(P)$ is not safe either, we prune the current subtree. However, if $L_M^{\forall}(P)$ is safe, we continue with the search. In this case, we can omit the KS analysis for all nodes in the current subtree as it will always give the same result. We assume the conditions in lines 9 and 13 are evaluated in a lazy fashion and therefore we only use the KS analysis based on the value of the Boolean switch b.
4. If $K_M^{\forall}(P)$ or $L_M^{\forall}(P)$ are safe, we partition the parameter set P into two subsets by considering a further constraint from the list Ψ (line 18). Those two subsets correspond to the positive and negative values of the chosen constraint, respectively. We proceed by recursively analyzing both subsets (lines 19–20).

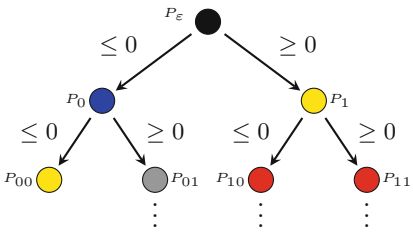

Fig. 5. Search tree in the parameter space. **yellow**: $L_M^{\exists}(P)$ satisfies the property φ while $K_M^{\exists}(P)$ violates it. **blue**: $L_M^{\forall}(P)$ satisfies the property φ while $K_M^{\forall}(P)$ violates it. **gray**: The node is only explored by the LHA analysis. **red**: The node is only explored by the KS analysis (Color figure online).

Search tree implications. In Fig. 5, we illustrate the potential impact of the LHA $L_M^{\exists}(P)$ and $L_M^{\forall}(P)$ on the structure of the search tree compared to an approach only using the KS analysis. We observe that $L_M^{\exists}(P)$ satisfies the property φ in the node P_1, whereas $K_M^{\exists}(P)$ violates the property. The LHA analysis benefits in two ways from this result. Firstly, it finds a large parameter set P_1, whereas the KS analysis can at most find valid sets in some of the child nodes, which are subsets of P_1. Secondly, the LHA analysis does not explore the children of P_1 in the search tree, which improves the algorithm performance. Moreover, $L_M^{\forall}(P)$ satisfies the property φ in the node P_0, whereas $K_M^{\forall}(P)$ violates the property. Therefore, the KS analysis prunes the subtree P_0, but a valid parameter set P_{00} can be found by the LHA analysis.

5 Evaluation

We have implemented the algorithm in MATLAB in the tool Hydentify. We use the library PPL [2] for the operations on polytopes and the SpaceEx model

checker [11] for the analysis of the LHA. Note that the default version of SpaceEx does not stop immediately after having found a property violation. Therefore, we have modified the version of SpaceEx so that it stops as soon as a property violation has been detected. This adjustment lets us improve the analysis performance. We apply the PHAVer scenario of SpaceEx which uses constraint polytopes to represent reachable regions to precisely analyze LHA.

Batt *et al.* [5] have presented a parameter identification approach for multiaffine systems implemented in a tool called RoVerGeNe. They approximate the system on the level of the induced Kripke structures. In the following evaluation, we compare the parameter identification results of our approach and RoVerGeNe. The integration of the induced KS into our algorithm allows for both a qualitative and a quantitative comparison of the two approaches. The implementation and models we used for the evaluation are available online[1].

Two-genes network model. We evaluate our tool on a number of models from the class of genetic regulatory networks. The experiments have been performed on a notebook with an Intel Core 2 Duo @ 2.26 GHz processor and 4 GB RAM.

For the evaluation purpose, we consider two classes of the two-genes network model introduced in Sect. 2. The first model class is the original system, while the second class is augmented by a stimulus. Note that our LHA framework enables an easy modeling and analysis of models with time dependent stimuli. In particular, we model a stimulus as a ramp function of an auxiliary variable t defined by the differential equation $\dot{t} = 1$. For every model class, we present two model instances. We look for parameters which lead to the repression of a given protein. For every instance and parameter identification algorithm, we report the following data: the coverage of the parameter domain, the number of the valid parameter sets found, the number of nodes in the search tree considered, the number of KS and LHA analyzed, and the runtime in seconds. By the term *parameter coverage* we denote the relation of the volume of the found valid parameters to the volume of the whole parameter domain $\mathcal{D}$. The results are provided in Table 1. The instances 1–2 correspond to the model class without a stimulus, whereas the other two instances belong to the class with a stimulus.

We first observe that the valid parameter regions found by our algorithm are usually much larger than the ones found by RoVerGeNe for both the models with and without the stimulus. Instance 2 provides a particularly illustrative example for the difference. Here, RoVerGeNe does not find any valid parameters, whereas our approach discovers valid parameter regions covering 38 % of the whole parameter domain. This behavior can be justified as follows. On the one hand, RoVerGeNe reports that both $K_M^{\exists}(P)$ and $K_M^{\forall}(P)$ at the root level reach the bad states. This results in analysis termination of RoVerGeNe. On the other hand, our algorithm proceeds in-depth with the analysis of the parameter space and detects 5 valid parameter regions. In instance 3, we see similar impact of taking LHA into account. In particular, both approaches consider 5 $K_M^{\forall}(P)$. However, our approach additionally considers 3 $L_M^{\forall}(P)$ which allow to extra unfold the parameter space. In this way, our approach analyzes 15 nodes and

[1] http://swt.informatik.uni-freiburg.de/tool/spaceex/hydentify.

Table 1. model ID: 1–2: without stimulus; 3–4: with stimulus; **% valid**: percentage of parameter space verified; **# sets**: number of parameter sets found; **# nodes**: number of nodes in the search tree; **# ∃-KS/∃-LHA**: number of $K_M^{\exists}/L_M^{\exists}$ analyzed; **# ∀-KS/∀-LHA**: number of $K_M^{\forall}/L_M^{\forall}$ analyzed; **runtime**: runtime in seconds

model ID	% valid		# sets		# nodes		# ∃-KS/∃-LHA			# ∀-KS/∀-LHA			runtime [s]	
								Hydentify			Hydentify			
	RoVerGeNe	Hydentify	RoVerGeNe	Hydentify	RoVerGeNe	Hydentify	RoVerGeNe	KS	LHA	RoVerGeNe	KS	LHA	RoVerGeNe	Hydentify
1	60	85	7	5	23	23	23	9	21	16	5	10	11	32
2	0	38	0	5	1	79	1	1	79	1	1	62	2	95
3	65	73	3	5	9	15	9	9	12	5	5	3	6	22
4	60	84	4	3	13	9	13	7	7	9	4	1	8	18

finds 5 valid regions compared to 9 nodes and 3 valid regions for RoVerGeNe, respectively. At the same time, the refined precision of LHA can shrink the search space. For example, in instance 4, the new algorithm achieves the parameter coverage of 84 % vs. 60 % by RoVerGeNe having considered only 9 nodes vs. 13 nodes in case of RoVerGeNe. We note that the valid parameter sets which are near the search tree root lead to larger parameter coverage with only a few parameter sets. This fact is confirmed by instance 4 where our approach finds 3 valid sets which cover a bigger region than the 4 valid sets found by RoVerGeNe.

Myocyte model. A fundamental question in the treatment of cardiac disorders, such as tachycardia and fibrillation [8], is the identification of circumstances under which such a disorder arises. Cardiac contraction is electrically regulated by particular cells, known as myocytes. For each electric stimulus originating in the sino-atrial node of the heart (its natural pacemaking unit), the myocytes propagate this stimulus and enforce the contraction of the cardiac muscle, known as a heart beat. Grosu *et al.* [13] have identified an MHA model for human ventricular myocytes and recast the biological investigation of lack of excitability to a computational investigation of the parameter ranges for which the MHA accurately reproduces lack of excitability. We apply our algorithm to this model and compare its performance with RoVerGeNe. The model has 4 continuous variables and 4 parameters. In our setting, a valid parameter set ensures that the myocyte is not excited.

We remark that our parameter identification approach has a large potential with respect to *parallelization* as the LHA and KS can be analyzed independently. We made use of this property and utilized a parallel version of our implementation for the analysis of the myocyte model. The experiments have been run on a Linux cluster with 32 AMD @ 2.3 GHz cores and 256 GB RAM. The model behavior is analyzed within a biologically reasonable time span of 1 ms. We note that the stimulus and particularly its duration require a special treatment as it strongly impacts the myocyte behavior. The stimulus in our model starts with

the value 1 and linearly drops to 0. We explore the impact of the stimulus length on the myocytes excitement.

Our approach empirically shows that the *whole* parameter domain is valid for *all* stimuli of length up to approximately 0.12 ms. In other words, we can provide a *lower bound* on the stimulus length which makes the myocyte model *excitable*. For this purpose, we have discretized the stimulus length with a step of 0.1 ms, i.e., we have considered stimuli of the length $0, 0.1, \ldots, 1$ ms. Having identified an interval of interest $[0.1; 0.2]$, we have discretized it in a finer way with a step of 0.02 ms. The new analysis takes 187 s and detects that the *whole* parameter domain is valid for the stimulus of length 0.12 ms, whereas RoVerGeNe reports the coverage of 29 % after 48 s. The parameter coverage computed by our algorithm drops to 30 % for the stimulus length of 0.14 ms and the analysis takes 1785 s. We note that the coverage computed by RoVerGeNe stays the same for all stimulus lengths as it cannot reason about time. This is a conceptual improvement over RoVerGeNe.

6 Related Work

A number of approaches have been developed to solve the parameter identification problem for hybrid automata. First, as already outlined in the previous section, Batt *et al.* [5] presented a parameter identification approach based on the abstraction of MHA by Kripke structures. By using our LHA abstraction, we improve the abstraction precision and in this way find more valid parameters. Dang *et al.* [9] introduced a "sensitive barbarian" approach. Bartocci *et al.* [3] consider a modular version of this approach. The main idea is to combine *numerical* simulation with sensitivity analysis to reduce the considered parameter space. A crucial difference to our approach lies in the fact that we utilize a *symbolic* analysis of the reachable states. In a further approach, Dreossi *et al.* [10] provide a parameter synthesis algorithm for polynomial dynamical systems. Their synthesis technique uses the Bernstein polynomial representation and recasts the synthesis problem as a linear programming problem. Note that they consider only *discrete* time dynamical systems, whereas we treat time as a *continuous* entity. The work by Liu *et al.* [16] tackles the parameter synthesis problem using δ-complete decision procedures [12] for first-order logic (FOL) formulae to overcome undecidability issues. In this setting, a FOL formula describes the states reachable with a finite number of steps. Therefore, the parameter identification problem is reduced to finding a satisfying valuation of the parameters for this formula. This approach requires enumerating *all the discrete paths* of a particular length, which leads to performance degradation for large models. In our approach, we employ the *symbolic* model checker SpaceEx, which prunes the state space exploration by checking whether the currently considered states have already been visited.

7 Conclusion

We have presented a novel parameter identification algorithm for multiaffine hybrid automata. In our algorithm, we compute equivalence classes in the parameter space and explore them in a hierarchical way. The approximation of the system dynamics with linear hybrid automata lets us keep the timing information in our abstraction. This allows us to precisely treat time-dependent properties such as a stimulus.

Given a parameter polytope P, we compute an LHA which overapproximates the system behavior for P. Furthermore, we compute another LHA which enables us to prune the search tree. We have evaluated our approach on a model of a genetic regulatory network and a myocyte model and demonstrated its improvement over RoVerGeNe, a tool for parameter identification based on a purely discrete abstraction.

In the future, we plan to investigate the application of hybrid model checkers which support more expressive continuous dynamics. This enables approximating the parametrized system dynamics with a hybrid automaton class featuring dynamics beyond the ones of LHA.

Acknowledgments. This work was partly supported by the European Research Council (ERC) under grant 267989 (QUAREM), by the Austrian Science Fund (FWF) under grants S11402-N23, S11405-N23 and S11412-N23 (RiSE/SHiNE) and Z211-N23 (Wittgenstein Award), and by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS, http://www.avacs.org/).

References

1. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theor. Comput. Sci. **138**(1), 3–34 (1995)
2. Bagnara, R., Hill, P.M., Zaffanella, E.: The parma polyhedra library: toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. Sci. Comput. Program. **72**(1), 3–21 (2008)
3. Bartocci, E., Bortolussi, L., Nenzi, L.: A temporal logic approach to modular design of synthetic biological circuits. In: Gupta, A., Henzinger, T.A. (eds.) CMSB 2013. LNCS, vol. 8130, pp. 164–177. Springer, Heidelberg (2013)
4. Bartocci, E., Corradini, F., Di Berardini, M.R., Entcheva, E., Smolka, S., Grosu, R.: Modeling and simulation of cardiac tissue using hybrid I/O automata. Theor. Comput. Sci. **410**(410), 3149–3165 (2009)
5. Batt, G., Belta, C., Weiss, R.: Temporal logic analysis of gene networks under parameter uncertainty. IEEE Trans. Autom. Control **53**, 215–229 (2008)
6. Batt, G., Yordanov, B., Weiss, R., Belta, C.: Robustness analysis and tuning of synthetic gene networks. Bioinformatics **23**(18), 2415–2422 (2007)
7. Belta, C., Habets, L.: Controlling a class of nonlinear systems on rectangles. IEEE Trans. Autom. Control **51**(11), 1749–1759 (2006)

8. Cherry, E.M., Fenton, F.H.: Visualization of spiral and scroll waves in simulated and experimental cardiac tissue. New J. Phys. **10**, 125016 (2008)
9. Dang, T., Donzé, A., Maler, O., Shalev, N.: Sensitive state-space exploration. In: CDC, pp. 4049–4054 (2008)
10. Dreossi, T., Dang, T.: Parameter synthesis for polynomial biological models. In: Proceedings of HSCC 2014: The 17th International Conference on Hybrid Systems: Computation and Control, HSCC 2014, pp. 233–242. ACM, New York, NY, USA (2014)
11. Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: SpaceEx: scalable verification of hybrid systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 379–395. Springer, Heidelberg (2011)
12. Gao, S., Avigad, J., Clarke, E.M.: δ-complete decision procedures for satisfiability over the reals. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS, vol. 7364, pp. 286–300. Springer, Heidelberg (2012)
13. Grosu, R., Batt, G., Fenton, F.H., Glimm, J., Le Guernic, C., Smolka, S.A., Bartocci, E.: From cardiac cells to genetic regulatory networks. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 396–411. Springer, Heidelberg (2011)
14. Habets, L.C.G.J.M., Collins, P.J., van Schuppen, J.H.: Reachability and control synthesis for piecewise-affine hybrid systems on simplices. IEEE Trans. Autom. Control **51**(6), 938–948 (2006)
15. Kloetzer, M., Belta, C.: Reachability analysis of multi-affine systems. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 348–362. Springer, Heidelberg (2006)
16. Liu, B., Kong, S., Gao, S., Zuliani, P., Clarke, E.M.: Parameter synthesis for cardiac cell hybrid models using delta-decisions. CoRR, abs/1407.1524 (2014)
17. Myers, C.J.: Engineering Genetic Circuits. Chapman and Hall/CRC (2010)

http://www.springer.com/978-3-319-26286-4

Hardware and Software: Verification and Testing
11th International Haifa Verification Conference, HVC 2015, Haifa, Israel, November 17-19, 2015, Proceedings
Piterman, N. (Ed.)
2015, XVI, 293 p. 88 illus. in color., Softcover
ISBN: 978-3-319-26286-4