SpaceRover: Parameter Synthesis for Multiaffine Systems beyond RoVerGeNe

Sergiy Bogomolov[1,2], Christian Schilling[2], Ezio Bartocci[3], Gregory Batt[4], Andreas Podelski[2], and Radu Grosu[3]

[1] IST Austria, Austria
[2] University of Freiburg, Germany
[3] Vienna University of Technology, Austria
[4] INRIA Paris-Rocquencourt, 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 usually exhibit a highly non-linear behavior. In this paper, we consider the problem of *parameter identification* for MHA. For this purpose, we extend the algorithm implemented in the tool RoVerGeNe. We lift the purely discrete abstractions of RoVerGeNe to linear hybrid automata (LHA), which can be analyzed by the SpaceEx model checker. Besides a general increase in precision, this enables the 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 such as embedded systems [2], biology [6], etc. Due to its behavioral complexity, the biological domain can particularly benefit from the expressiveness of hybrid automata. However, in this setting the models mostly have highly non-linear dynamics. In the last decade, a number of powerful hybrid model checkers, e.g., SpaceEx [15], d/dt [3], Flow* [11], dReach [17], which can handle different classes of hybrid automata, has evolved. These tools have been developed to *verify* whether a given *safety* property holds for a considered hybrid automaton.

Parameter identification is the complementary problem to verification. Here we want to find a parameter set for which a given property is fulfilled by the system. In the biological domain, this problem is of large importance considering the current limitations on experimental measurement techniques [23].

In this paper, we present a novel approach to solve the parameter identification problem for the class of multiaffine hybrid automata (MHA), which we have implemented in the tool SpaceRover. We reduce the parameter identification problem to solving multiple *verification* problems. We extend the approach originally presented by Batt et. al. [7] and implemented in the tool RoVerGeNe. In short, the algorithm consists of the following steps: We divide the parameter

space into a number of equivalence classes. Given an equivalence class, we *approximate* the behavior of the system with a linear hybrid automaton (LHA), which can be analyzed by the hybrid model checker SpaceEx [15]. In addition, we utilize a hierarchical search to prune the search tree.

Our reachability analysis is less conservative compared to RoVerGeNe where time-dependent information is lost. This enables us to find larger parameter sets and to analyze systems featuring an additional *time-dependent* stimulus. SpaceRover still uses RoVerGeNe to abstract the MHA with a Kripke structure (KS) analyzed thereafter by the tool SMV [10]. However, if the resulting precision is unsatisfactory, we apply our new LHA analysis using the SpaceEx model checker. In that way, we leverage different abstraction levels and find a "sweet spot" between precision and performance.

Related work. Several approaches have been developed to solve the parameter identification problem for hybrid automata. To begin with, a "sensitive barbarian" approach was introduced by Dang et al. [13]. Bartocci et al. [5] 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 at al. [14] 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. [22] tackles the parameter synthesis problem using $\delta-$complete decision procedures [16] 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.

The rest of the paper is organized as follows. In Section 2, we introduce some preliminary notions. In Section 3, we recapitulate the algorithms behind RoVerGeNe. In Section 4, we present the theoretical foundations of our new approach. In Section 5, we evaluate it on two biological models. In Section 6, we conclude and discuss future work.

2 Preliminaries

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

Kripke structure. Given a set of atomic propositions A, a Kripke structure (KS) [21] is an abstract machine for modeling system behavior, represented by the tuple $K = (S, S_0, T, L)$ where S is a finite set of states, $S_0 \subseteq S$ is a set of initial states, $T \subseteq S \times S$ is a transition relation and $L : S \to 2^A$ is an interpretation function that defines which atomic propositions hold in the states.

Multiaffine function. A multiaffine function $f : \mathbb{R}^n \to \mathbb{R}^p$ ($p \in \mathbb{N}$) 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 [20]. 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 $c_{i_1,\ldots,i_n} \in \mathbb{R}^p$ 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 formalism to model complex dynamical systems. It is represented by the tuple $H = (Loc, Inv, Flow, Trans, Init)$. *Loc* is a set of discrete *locations*. Each $\ell \in Loc$ is associated with a set of differential equations (or inclusions) $Flow(l)$ 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 $x_1, \ldots, x_n$. A set of *discrete transitions Trans* defines how the state can jump between locations when the state is 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* of a hybrid automaton is a function which defines the state of the hybrid automaton for every time moment. We consider $Flow(l)$ to be continuous dynamics of the following two forms:

1. 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).
2. If $\dot{x}(t) \in P$ where P is a polytope, then the HA is called a *linear hybrid automaton* (LHA).

Here $x(t) \in \mathbb{R}^n$ denotes the values of the continuous variables.

In the verification setting, we are interested in whether there exists a trajectory from the set *Init* to the set *Bad* which defines the *bad states* to be avoided.

Genetic regulatory network. A genetic regulatory network [7] is defined by the dynamics of the following form:

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

where x_i is the i-th component of the state vector $x \in \mathbb{R}^n$, $\kappa_{ij} \in \mathcal{D}_{ij}^P$ and $\gamma_{ij} \in \mathcal{D}_{ij}^D$ are production and degradation rate parameters of the parameter vector $p \in \mathbb{R}^m$, and r_{ij} are continuous piecewise-multiaffine functions arising from products of ramp functions r^+ and r^- of the form shown in Figure 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 the protein i. Assuming that the protein i does not regulate its own degradation, i.e., for $j \in D_i$, x_i does not occur in $r_{ij}^D(x)$, function $f = (f_1, \ldots, f_n)$ is multiaffine in x and affine in p. We note that the ramp functions induce a partition of the state space into a *grid* of hyper-rectangular regions. The values of the separating hyperplanes are called *thresholds* θ. In that way, we can define an MHA with locations induced by the state space partition. Note that an MHA provides a semantically equivalent representation of the dynamics (1). In the following, we

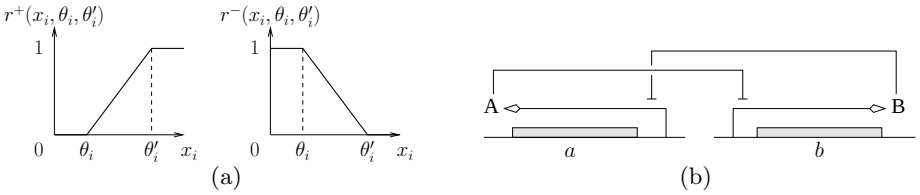

Fig. 1. (a) Ramp functions r^+ and r^-. (b) The two-genes network model.

will analyze the MHA representation of the system. The parameters allowed to vary in an interval are called *uncertain* parameters. The parameters with fixed values are semantically equivalent to constants. $\mathcal{D}_{ij}^P$ and $\mathcal{D}_{ij}^D$ define the domains of the uncertain parameters for the production and degradation terms, respectively. We denote the domain of all the uncertain parameters by $\mathcal{D}$.

Finally, a property φ describing the desired system behavior is provided. In this paper, we consider properties of the form $R_{init} \rightarrow \neg \mathsf{F} R_{bad}$ where the region R_{init} denotes the set of 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. We are interested in *identifying* a subset of the parameter domain $\mathcal{D}$ which ensures that the property φ holds for a given MHA. More precisely, we look for the parameters which ensure to not reach the region R_{bad} when starting in R_{init}.

Example (two-genes network). In the following, we illustrate our approach on the *two-genes network* [8] (also called *toggle switch* or *cross-inhibition network*).

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

where x_a and x_b define the concentrations of the proteins A and B, respectively. The parameters κ_a and κ_b define their range of production rates in the given intervals. As Figure 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 want to check a property which asks whether the protein concentrations cannot reach some specific threshold values when starting in an initial region.

Furthermore, 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:

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

We use a stimulus which is 1 at the beginning up until 0.29s and then drops linearly to 0 within 10ms, expressed by the ramp function of time $r^-(t, 0.29, 0.3)$.

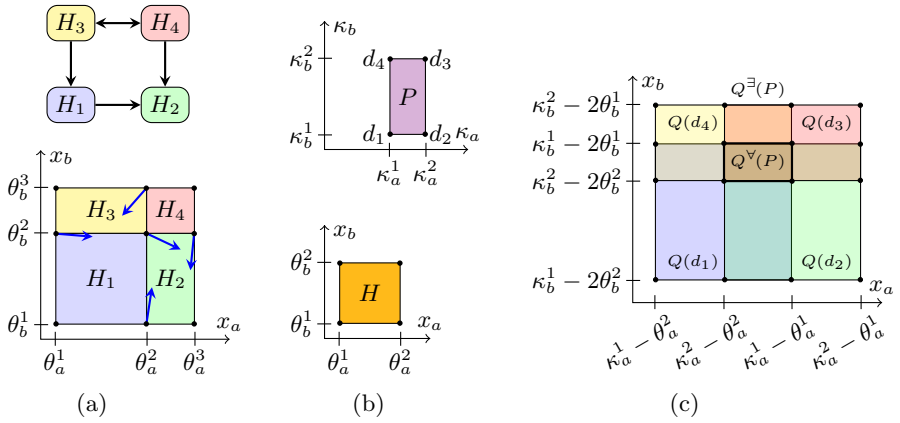

Fig. 2. (a) From state space partition to Kripke structure. Example for $\kappa_a = 10$ and $\kappa_b = 15$ (arrows normalized). (b-c) Flow computation for $P = [8, 12] \times [15, 20]$.

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, we can reformulate the description and also state that the inhibitory effect of the protein B is only relevant for the production of the protein A after 0.29s.

3 RoVerGeNe

The main idea behind RoVerGeNe is to generate a parameter space partition and iteratively compute Kripke structures (KS) which approximate the system dynamics for a particular set of parameters.

Constructing the states of a Kripke structure. In RoVerGeNe, the thresholds θ are considered as fixed, elementary controls of the PMA dynamics. They partition the state space of the system into hyper-rectangular regions. Each of them is associated to a state of a KS. We can formally capture this association by introducing the following equivalence relation: $x \cong x'$ iff x and x' are strictly inside the same hyper-rectangle. Let H_x be the equivalence class for x, i.e., the hyper-rectangle strictly containing x. Note that the equivalence classes are independent of p.

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 it and the associated KS are shown in Figure 2 (a).

Constructing the transitions of a Kripke structure. Defining the transitions of the KS requires some formal machinery. In contrast to states, they depend on p. Let A be a set of atomic propositions of the form $x_i \bowtie \theta_i$ where $\bowtie$ is in $\{>, <\}$ and θ_i is a threshold of x_i. Let $\models$ be the satisfaction relation with the

usual semantics. By definition, all states strictly inside a hyper-rectangle either satisfy or do not satisfy an atomic proposition $a \in A$.

Definition 1. *Given a parameter p and an MHA $M(p)$ whose dynamics depend on p, the* embedding transition system $T_M(p)$ *is defined as follows:*

- $x \to_p x'$ *iff: 1) H_x and $H_{x'}$ are either equal or adjacent. 2) There is a solution ξ of (1) and a positive time τ such that $\xi(0) = x$, $\xi(\tau) = x'$ and for all t in $[0, \tau]$, $\xi(t)$ stays within H_x and $H_{x'}$.*
- $x \models a$ *in $T_M(p)$ if $x \models a$ in $M(p)$.*

Almost all trajectories in $M(p)$ are also represented in $T_M(p)$. The exception are trajectories not passing through a common facet of two hyper-rectangles.

Given $T_M(p)$, we can now formally capture the transitions of the Kripke structure $K_M(p)$ by the quotient of $T_M(p)$ w.r.t. the state space partition.

Definition 2. *The* discrete abstraction $K_M(p)$ *of the MHA $M(p)$ is $T_M(p) / \cong$.*

From the properties of a quotient structure we can immediately infer that $K_M(p)$ simulates $T_M(p)$. In particular, if $K_M(p)$ satisfies a safety property, then so does $T_M(p)$, i.e., $K_M(p)$ is a conservative abstraction of $T_M(p)$.

Proposition 1. *[7] $T_M(p) \preceq K_M(p)$.*

An effective procedure for constructing transitions. Summarizing, Definition 2 captures the semantics of $K_M(p)$ and Proposition 1 shows that $K_M(p)$ conservatively abstracts $T_M(p)$. However, they do not provide an effective way of computing $K_M(p)$. For this purpose, we will use the fact that in $M(p)$, the dynamics $f(x, p)$ are multiaffine in x. Let *hull* be the convex hull operator.

Theorem 1. *[9] Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be a multiaffine function, H be a hyper-rectangle in $\mathbb{R}^n$ with corner set C_H, and Q be the interior of H. Then*

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

Intuitively, the theorem says that the behavior of $f(x, p)$ inside a hyper-rectangle is completely determined by its behavior in the corners of the hyper-rectangle. As a consequence, the following proposition can be proven.

Proposition 2. *[7] $T_M(p)$ and $K_M(p)$ have a transition between adjacent hyper-rectangles H and H' only if the projection of $f(x, p)$ on the $H \to H'$ direction is positive in at least one corner of the facet separating H from H'.*

The direction and strength of the derivative $\dot{x} = f(x, p)$ in a corner v of a hyper-rectangle depends in an affine way on the parameter values 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.

In the following, by *polytope* we always refer to a *convex polytope*. The instantiation of $f(x, p)$ in every corner v of the state space leads to a parameter space partition into polytopes. The theorem below provides an efficient way to compute the image of $f(x, p)$ for a particular state space corner v and a parameter vector $p \in P$ where P is a polytope.

Theorem 2. *[19] Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be an affine function, P be a convex polytope in $\mathbb{R}^n$ with corner set C_P, and Q be the interior of P. Then*

$$f(Q) \subseteq hull\{f(v) \mid v \in C_P\}.$$

Determining the parameters of interest. Given an MHA M with system dynamics $\dot{x} = f(x, p)$ and a property φ, we can find parameter sets P such that $M(p)$ satisfies φ for all $p \in P$ as follows:

1. Construct a hyper-rectangular partition Π_S of the state space of M.
2. Use Π_S to construct the states of a KS K, and to obtain a partition Π_P of M's parameter space $\mathcal{D}$.
3. For each pair of adjacent hyper-rectangles H_i and H_j in Π_S, and for each polytope Q in Π_P, construct the transitions of K between the states corresponding to H_i and H_j by examining $f(x, p)$ in the corners of Q, and the corners of the facet separating H_i from H_j.
4. Call SMV with input K and the *abstraction* of φ, and add Q to the desired parameter set if SMV reports that φ is satisfied.

By *abstraction of* φ we mean the following. Note that a Kripke structure does not allow to state quantitative properties, e.g., whether a particular threshold can be exceeded. Therefore, we abstract the originally quantitative property by mapping it to the corresponding regions in Π_S.

While sound, this approach is, however, too naive, as in general the number of polytopes in Π_P is very large. Therefore, RoVerGeNe explores larger groups of polytopes in Π_P in a tree-like fashion. To reduce the depth of the search tree, an additional *pruning* technique is employed.

The exploration makes use of two KS: $K_M^{\forall}(P)$ is an *under-approximation* of $K_M(p)$, and $K_M^{\exists}(P)$ is an *over-approximation* of $K_M(p)$, where P is a polytope in the parameter space containing p.

Definition 3. *Let $K_M(p)$ be the KS associated to MHA $M(p)$, where p is a parameter vector in the polytope P. The states of the KS $K_M^{\forall}(P)$ and $K_M^{\exists}(P)$ are the same as for $K_M(p)$. The transitions of $K_M^{\forall}(P)$ and $K_M^{\exists}(P)$ differ and are defined as follows:*

- $H_1 \to_P^{\exists} H_2$ *iff* $H_1 \to_p H_2$ *in* $K_M(p)$ *for some* $p \in P$.
- $H_1 \to_P^{\forall} H_2$ *iff* $H_1 \to_p H_2$ *in* $K_M(p)$ *for all* $p \in P$.

Proposition 3. *[7]* $K_M^{\forall}(P) \preceq K_M(p) \preceq K_M^{\exists}(P)$.

To construct the transitions of $K_M^{\exists}(P)$ and $K_M^{\forall}(P)$, we apply the following steps for each pair of adjacent hyper-rectangles H_1 and H_2 and each corner v on their respective separating facet F:

- Let $g(H_1, H_2) = \cup_{v \in F}\{p \in P \mid f(v, p) > 0\}$.
- Add $H_1 \to_P^{\forall} H_2$ and $H_1 \to_P^{\exists} H_2$ if $H_1 = H_2$.
- Add $H_1 \to_P^{\forall} H_2$ to $K_M^{\forall}(P)$ if $P \subseteq g(H_1, H_2)$.
- Add $H_1 \to_P^{\exists} H_2$ to $K_M^{\exists}(P)$ if $P \cap g(H_1, H_2) \neq \emptyset$.

Note that we use Theorem 2 to construct the parameter set satisfying the constraint $f(v,p)>0$. The above construction uses the following operations on polytopes: union, intersection, and test for equality, inclusion and emptiness.

From Propositions 1 and 3, and the transitivity of the simulation relation, we can immediately infer that $K_M^{\exists}(P)$ simulates $T_M(p)$ for any $p \in P$. Hence, if SMV returns true for $K_M^{\exists}(P)$, the parameter set P can be safely added to the set of parameters of interest.

Proposition 4. *[7]* $T_M(p) \preceq K_M^{\exists}(P)$.

However, we cannot infer that $T_M(p)$ simulates $K_M^{\forall}(P)$. Hence, while $K_M^{\forall}(P)$ is an under-approximation of $K_M(p)$ for $p \in P$, it is not an under-approximation of $T_M(p)$. The role of $K_M^{\forall}(P)$ is to *prune* the search tree in the parameter space exploration. In particular, RoVerGeNe prunes the parameter sets for which the abstraction $K_M(p)$ is too coarse for every $p \in P$. This renders the consideration of subsets of such a parameter set P useless as the violation of the property φ by $K_M^{\forall}(P)$ already implies a property violation by $K_M(p)$ for every $p \in P$.

Hierarchical parameter search. In more detail, RoVerGeNe works as follows:

1. Construct a list Ψ of all constraints $f(v,p) = 0$ where v is a corner in the hyper-rectangular state space partition of M.
2. Start with the parameter set P equal to the whole parameter space $\mathcal{D}$.
3. Call SMV to check whether $K_M^{\exists}(P) \models \varphi$ holds. If this is the case, then add P to the list of valid parameter sets. Otherwise, let SMV check whether $K_M^{\forall}(P) \not\models \varphi$ holds. In this case, prune the considered search subtree.
4. If $K_M^{\exists}(P) \not\models \varphi$ and $K_M^{\forall}(P) \models \varphi$, split polytope P into P_1 and P_2 using the next constraint in Ψ. Recursively continue from step 3 for both P_1 and P_2.

4 SpaceRover

Although very encouraging, the RoVerGeNe results are too conservative, because RoVerGeNe loses almost all information about the continuous dynamics during the abstraction process to the Kripke structures.

In our novel approach we keep the continuous dynamics of the original MHA by abstracting it to an LHA. Furthermore, we incorporate RoVerGeNe within the SpaceRover framework to leverage different abstraction levels.

In a nutshell, the LHA are constructed as follows. We decorate the states and transitions of the KS $K_M^{\exists}(P)$ and $K_M^{\forall}(P)$ with continuous variables, invariants and flows, respectively. In this way, we obtain the LHA $L_M^{\exists}(P)$ and $L_M^{\forall}(P)$ which play the same role in the hierarchical parameter search. We have implemented this approach in the tool SpaceRover which uses SpaceEx for the LHA analysis.

The state space of the LHA. Given a parameter vector p in a parameter set P, the states of the LHA $L_M(p)$ consist of pairs (ℓ, x), where ℓ is a *location* of the LHA, and x is a value of the *continuous* variables. The locations are the same as the states of the KS $K_M(p)$. The continuous state variables x correspond to the ones of the MHA $M(p)$. The LHA $L_M^{\exists}(P)$ and $L_M^{\forall}(P)$ have the same state space as the LHA $L_M(p)$.

The location invariants. The invariant associated to location ℓ of the LHA $L_M(p)$, $L_M^{\exists}(P)$ and $L_M^{\forall}(P)$ is given by the facets of the hyper-rectangle H_ℓ corresponding to this location.

The location flows. The flows of the LHA $L_M(p)$, $L_M^{\exists}(P)$ and $L_M^{\forall}(P)$ are computed in such a way that these LHA are in simulation relations similar to the ones for $K_M(p)$, $K_M^{\exists}(P)$ and $K_M^{\forall}(P)$. The computation again uses the multiaffine dependence of $f(x,p)$ on x and the affine dependence of $f(x,p)$ on p.

For a fixed parameter vector p and a fixed hyper-rectangle H with corner set C_H, Theorem 1 states that $f(x,p) \in Q = hull(\{f(v,p) \mid v \in C_H\})$ for each $x \in H$. In other words, $f(x,p)$ is within the convex hull Q of the hyper-rectangle corners v. Therefore, we end up with a polytope bounding the flow, i.e., the dynamics are represented in the form of differential inclusion.

We lift the definition of the LHA $L_M(p)$ to parameter sets by introducing the LHA $L_M^{\exists}(P)$ and $L_M^{\forall}(P)$. Here, we proceed similarly to the KS abstractions $K_M^{\exists}(P)$ and $K_M^{\forall}(P)$.

To obtain the flows, we take advantage of Theorem 2. For fixed continuous state x, polytope P with corner set C_P, and parameter vector $p \in P$, it states that $f(x,p) \in Q = hull(\{f(x,d) \mid d \in C_P\})$.

Putting Theorems 1 and 2 together, we obtain an algorithm for computing the flows of the LHA $L_M^{\exists}(P)$ and $L_M^{\forall}(P)$, given a hyper-rectangle H and a polytope P with corner sets C_H and C_P, respectively.

- For each $d \in C_P$ compute polytope $Q(d)$ by traversing all vertices $v \in C_H$ and collecting $f(v,d)$.
- For $L_M^{\exists}(P)$, the polytope $Q^{\exists}(P) = hull(\cup_{d \in C_P} Q(d))$ is the convex hull of all polytopes $Q(d)$ where $d \in C_P$.
- For $L_M^{\forall}(P)$, the polytope $Q^{\forall}(P) = \cap_{d \in C_P} Q(d)$ is the intersection of all polytopes $Q(d)$ where $d \in C_P$.

Note that similarly to the LHA $L_M(p)$ we end up with LHA whose dynamics are defined by differential inclusions.

Using Theorems 1 and 2, we obtain the following relation between the polytopes $Q(p)$, $Q^{\forall}(P)$ and $Q^{\exists}(P)$ for some parameter vector p of a polytope P.

Proposition 5. $Q^{\forall}(P) \subseteq Q(p) \subseteq Q^{\exists}(P)$.

Example. Consider the state space rectangle H and the parameter space rectangle P in Figure 2(c). The state equation $\dot{x} = f(x,p)$ is:

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

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: $(\kappa_a^i - \theta_a^j, \quad \kappa_b^k - 2\theta_b^\ell)$.

Let the corners of the parameter space rectangle P in Figure 2(b) 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 state space 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 transitions of the LHA. The transitions of the LHA $L_M(p)$ are the same as the ones of $K_M(p)$. Similarly, the transitions of $L_M^{\exists}(P)$ and $L_M^{\forall}(P)$ are the same as the transitions of $K_M^{\exists}(P)$ and $K_M^{\forall}(P)$, respectively. We do not add any constraints (guards) on the transitions, because the invariants already guarantee that a transition between two locations can only be taken if the continuous state is a point on the facet (which is the intersection of the invariants).

The simulation relations. On the one hand, Kripke structures and LHA we construct share the same discrete structure. On the other hand, LHA additionally define continuous dynamics in every location. Therefore, by construction we obtain the following simulation relation.

Proposition 6. $T_M(p) \preceq L_M(p) \preceq K_M(p)$.

This proposition states that the LHA $L_M(p)$ is a conservative abstraction of the transition system $T_M(p)$, and that the KS $K_M(p)$ is in turn a conservative abstraction of $L_M(p)$.

Proposition 7. $L_M^{\forall}(P) \preceq L_M(p) \preceq L_M^{\exists}(P)$.

This proposition states that $L_M^{\forall}(P)$ is an under-approximation of $L_M(P)$ and $L_M^{\exists}(p)$ is an over-approximation of $L_M(P)$. It follows from Propositions 3 and 5 and the construction of the LHA. In particular, Proposition 3 states that every transition in $K_M^{\exists}(p)$ is also present in $K_M(p)$ and likewise every transition in $K_M(p)$ is also present in $K_M^{\forall}(P)$. Furthermore, by construction, the LHA $L_M^{\exists}(P)$ and $L_M^{\forall}(P)$ have the same transitions as $K_M^{\exists}(P)$ and $K_M^{\forall}(P)$, respectively. In a similar fashion, Proposition 5 states that the flows of $L_M^{\forall}(P)$ are included in the flows of $L_M(P)$, which again are included in the flows of $L_M^{\exists}(p)$. Finally, we note that all the three LHA we consider have the same sets of locations and the location invariants do not depend on the parameters. Therefore, for every particular location, the invariants are the same for $L_M(p)$, $L_M^{\exists}(P)$ and $L_M^{\forall}(p)$.

From Propositions 6 and 7 we can finally derive the following statement.

Proposition 8. $T_M(p) \preceq L_M^{\exists}(P)$.

This proposition is important as it justifies the use of $L_M^{\exists}(P)$ when searching for parameters for which $T_M(p)$ satisfies a given property φ.

Proposition 9. $L_M^{\forall}(P) \preceq K_M^{\forall}(P)$ *and* $L_M^{\exists}(P) \preceq K_M^{\exists}(P)$.

This proposition relates the LHA and KS abstractions. It states that $K_M^{\exists}(P)$ and $K_M^{\forall}(P)$ are conservative abstractions of the LHA abstractions and thus allow more behavior than $L_M^{\exists}(P)$ and $L_M^{\forall}(P)$, respectively. Similarly to the case of KS, while $L_M^{\forall}(P)$ is an under-approximation of $L_M(p)$ for $p \in P$, it is not an under-approximation of $T_M(p)$.

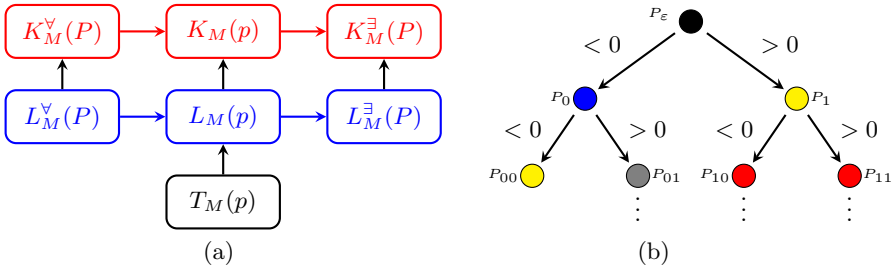

Fig. 3. (a) Summary of simulation relations for some p in P. (b) Search tree in the parameter space. $L_M^{\exists}$ satisfies the property φ in the nodes P_{00} and P_1 whereas $K_M^{\exists}$ violates it. $L_M^{\forall}$ satisfies the property φ in the node P_0 whereas $K_M^{\forall}$ violates it. P_{00} and P_{01} are only explored by SpaceRover. P_{10} and P_{11} are only explored by RoVerGeNe.

The simulation relationship between all the transition systems considered in this paper are shown in Figure 3(a). An arrow $S_i \to S_j$ in this figure means that transition system S_j simulates transition system S_i.

Stimulus treatment. We consider a time-dependent stimulus in our SpaceRover framework. It is handled in the following way. As already outlined, we assume the stimulus function to be a ramp function of time. Therefore, the stimulus induces the new dimension *time*. We note that the nature of the time dimension is different from other state dimensions in the sense that the time can *only* grow. In other words, the flow for time is always $\dot{t} = 1$. Based on the time partition and the time flow, SpaceRover can compute $L_M^{\exists}(P)$ and $L_M^{\forall}(P)$ and analyze them in a precise manner.

Determining the parameters of interest. Given an MHA M with $\dot{x} = f(x, p)$, a safety property φ and the domain of uncertain parameters $\mathcal{D}$, SpaceRover explores the parameter space of M in a hierarchical way similar to RoVerGeNe.

The pseudocode of the algorithm is given in Figure 4. In the following, we explain the main steps of the algorithm. Initially, SpaceRover investigates the corners of the state space partition and collects the constraints Ψ over the parameters in the function ***CollectConstraintsList*** (line 2). Now, the algorithm moves on to the main function ***Synthesis*** (line 4) 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 parameter sets. It takes a list L of constraints which encodes hyperplanes used to define the current parameter set. We initially call the synthesis function with $L = \varepsilon$ as we first consider the whole parameter space. Now we look at the function ***Synthesis*** in more detail. We start by calling the function ***PolytopeFromConstraintsList*** (line 7) which builds a parameter polytope P based on the provided list L of constraints over parameters and the parameter space domain $\mathcal{D}$. In line 8 we build the KS $K_M^{\exists}, K_M^{\forall}$ and the LHA $L_M^{\exists}, L_M^{\forall}$. Note that on the implementation level we compute them only *on demand*. The computed approximations are analyzed in the following way:

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

6  Synthesis(L, Ψ, M, φ, 𝒟) % constr. list L ...
7    P := PolytopeFromConstraintsList(L, 𝒟);
8    K_M^∃, K_M^∀, L_M^∃, L_M^∀ = ConstructSystems(M, P);
9    if (SMV(K_M^∃, φ))
10     V := V ∪ P; return; % valid set
11   elseif (SpaceEx(L_M^∃, φ))
12     V := V ∪ P; return; % valid set
13   elseif (¬SMV(K_M^∀, φ))
14     if (¬SpaceEx(L_M^∀, φ))
15       return; % no future valid sets
16   % descend to child nodes in search tree
17   c := first(Ψ); Ψ := rest(Ψ);
18   Synthesis(concatenate(L, ¬c), Ψ, M, φ, 𝒟);
19   Synthesis(concatenate(L, c), Ψ, M, φ, 𝒟);
```

Fig. 4. The SpaceRover algorithm in pseudocode.

1. If the property φ holds for the KS $K_M^{\exists}$ (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 RoVerGeNe was not able to prove the validity of P, we continue with the finer analysis using $L_M^{\exists}$. We forward it to SpaceEx to check the property φ (line 11). Similarly to the step 1, in case of the 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 $K_M^{\forall}$ and $L_M^{\forall}$. If both of them *violate* the property φ, we prune the search tree as no valid parameter sets can be found for the subsets of P. In more detail, the LHA $L_M^{\forall}$ for the parameter set P has the behavior of *all* $L_M(p)$ for $p \in P$. Therefore, if SpaceEx reports a property violation for the set P, the property will not hold for all $L_M(p)$. Note that due to efficiency reasons we first analyze $K_M^{\forall}$ (line 13) and move on to $L_M^{\forall}$ (line 14) only if the KS $K_M^{\forall}$ is safe with respect to the property φ. Furthermore, we observe that the property satisfaction for either $K_M^{\forall}$ or $L_M^{\forall}$ does not lead to the property satisfaction for the original system as the $K_M^{\forall}/L_M^{\forall}$ *neither* over- nor under-approximate the original dynamics.
4. Otherwise, we partition the parameter set P into two subsets by considering a further constraint from the list Ψ (line 17). Those two subsets correspond to the positive and negative values of the chosen constraint, respectively. We proceed by recursively analyzing both subsets (lines 18–19).

Example. In Figure 3(b), we illustrate the impact of the LHA $L_M^{\exists}/L_M^{\forall}$ on the structure of the search tree. We observe that $L_M^{\exists}$ satisfies the property φ in the node P_1 whereas $K_M^{\exists}$ violates the property. SpaceRover benefits in two ways from this result. Firstly, it finds a large parameter set P_1, whereas RoVerGeNe can at most find valid sets in some of the child nodes, which are *subsets* of P_1. Secondly, SpaceRover does not explore the children of P_1 in the search tree, which improves the algorithm performance. Moreover, $L_M^{\forall}$ satisfies the property φ in the node P_0 whereas $K_M^{\forall}$ violates the property. Therefore, RoVerGeNe prunes the subtree P_0 although a valid parameter set P_{00} can be found by SpaceRover.

5 Implementation and Results

We use SpaceEx v0.9.8 for the analysis of the LHA constructed by SpaceRover. Note that the default version of SpaceEx is meant primarily for the *verification* setting and thus 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 is particularly suited for the analysis of LHA. In our evaluation, we compare the parameter identification results of SpaceRover and RoVerGeNe. furthermore investigate the benefits of the RoVerGeNe integration

Two-genes network model. In this section, we first evaluate our tool SpaceRover on a number of models from the class of genetic regulatory networks. We have implemented SpaceRover in MATLAB. The tool relies on the library PPL [4] for the operations on polytopes. The SpaceRover implementation and the models we used for the evaluation are available online[1]. The experiments have been performed on a notebook running Fedora on an Intel Core 2 Duo @ 2.26GHz processor with 4 GB RAM.

For the evaluation purpose, we consider two classes of the two-genes network model introduced in Section 2. The first model class is the original system, while the second class is augmented by a stimulus. 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 finally 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 SpaceRover 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

[1] `http://swt.informatik.uni-freiburg.de/tool/spacerover`

model ID	% valid		# sets		# nodes		# ∃-KS/∃-LHA			# ∀-KS/∀-LHA			runtime [s]	
	RoVerGeNe	SpaceRover	RoVerGeNe	SpaceRover	RoVerGeNe	SpaceRover	RoVerGeNe	SpaceRover		RoVerGeNe	SpaceRover		RoVerGeNe	SpaceRover
								KS	LHA		KS	LHA		
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

Table 1. model ID: 1–2: without stimulus; 3–4: with stimulus; **granularity**: granularity of considered model (High/Low); **% 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;

example for the difference between RoVerGeNe and SpaceRover. Here, RoVerGeNe does not find any valid parameters, whereas SpaceRover 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 the root ∀-KS and ∃-KS have intersection with the bad states. This results in the analysis termination of RoVerGeNe. On the other hand, SpaceRover proceeds in-depth with the analysis of the parameter state space and detects 5 valid parameter regions. In instance 3, we see similar impact of taking LHA into account. In particular, we see that both RoVerGeNe and SpaceRover consider 5 ∀-KS. However, SpaceRover additionally considers 3 ∀-LS which allow to extra unfold the parameter state space. In this way, SpaceRover analyzes 15 nodes and finds 5 valid regions compared to 9 nodes and 3 valid regions for RoVerGeNe, respectively. At the same time, the refined precision of LHA might shrink the search space. For example, in instance 4, SpaceRover 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 SpaceRover finds 3 valid sets vs. 4 valid sets for RoVerGeNe, whereas the coverage of SpaceRover is still higher than the one of RoVerGeNe as outlined above.

Myocyte model. A fundamental question in the treatment of cardiac disorders, such as tachycardia and fibrillation [12], 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. [18] have identified an MHA model for human ven-

tricular myocytes, and recasted 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 SpaceRover 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 the *parallelization* as induced LHA and KS can be analyzed independently. We make use of this property and utilize a parallel version of our SpaceRover algorithms for the analysis of the myocyte model. We have run our experiments on a machine running Debian with a 32 cores AMD @ 2.3Ghz and 256GB RAM. We analyze the model behavior within a biologically reasonable time span of 1 ms. We note that the stimulus and particularly its length require a special treatment as the stimulus strongly impacts the myocyte behavior. The stimulus in our model starts with the value one and linearly drops to zero. We explore the impact of the stimulus length on the myocytes excitement.

SpaceRover empirically shows that the *whole* parameter domain is valid for *all* stimuli of length up to approximately 0.12 ms. For this purpose, we have discretized the stimulus length with a step of 0.1 ms, i.e., we have considered the stimuli of the length 0, 0.1, ..., 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 SpaceRover analysis takes 187 seconds 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 seconds. The parameter coverage computed by SpaceRover drops to 30% for the stimulus length of 0.14 ms and the analysis takes 1785 seconds. RoVerGeNe achieves 29% of the parameter coverage in this setting and takes 48 seconds. We note that the coverage computed by RoVerGeNe stays the same for all the stimuli length, as it cannot reason about their length. This is a conceptual improvement of SpaceRover over RoVerGeNe in the sense that RoVerGeNe cannot take *time-dependent* properties of the stimulus into account.

6 Conclusion

In this paper, we have presented a novel parameter identification algorithm for multiaffine hybrid automata. The proposed algorithm has been implemented in the tool SpaceRover. We compute equivalence classes in the parameter space and explore them in a hierarchical way. The approximation of the system dynamics with LHA lets us keep the timing information in our abstraction. This allows us to precisely treat time-dependent properties such as a stimulus. We have evaluated SpaceRover on a model of a genetic regulatory network and a myocyte model and demonstrated its improvement over RoVerGeNe. For the future, it will be interesting to investigate the application of hybrid model checkers which support more expressive continuous dynamics. For this purpose, we need to look at the ways to approximate the parametrized system dynamics with a hybrid automaton class featuring dynamics beyond the ones of LHA.

References

1. 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(1):3–34, 1995.
2. R. Alur, H. T.A., and P. H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22:181–201, 1996.
3. E. Asarin, T. Dang, and O. Maler. The d/dt Tool for Verification of Hybrid Systems. In *In Proc. of CAV'02, the 14th International Conference on Computer Aided Verification*, volume 2404 of *LNCS*, pages 365–370. Springer, 2002.
4. R. Bagnara, P. M. Hill, and E. Zaffanella. The parma polyhedra library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1):3–21, 2008.
5. E. Bartocci, L. Bortolussi, and L. Nenzi. A temporal logic approach to modular design of synthetic biological circuits. In *Proc. of CMSB 2013: 11th International Conference on Computational Methods in Systems Biology*, volume 8130 of *LNCS*, pages 164–177. Springer, 2013.
6. E. Bartocci, F. Corradini, M. R. Di Berardini, E. Entcheva, S. Smolka, and R. Grosu. Modeling and simulation of cardiac tissue using hybrid I/O automata. *Theoretical Computer Science*, 410(410):3149–3165, 2009.
7. G. Batt, C. Belta, and R. Weiss. Temporal logic analysis of gene networks under parameter uncertainty. *IEEE Trans. of Automatic Control*, 53:215–229, 2008.
8. G. Batt, B. Yordanov, R. Weiss, and C. Belta. Robustness analysis and tuning of synthetic gene networks. *Bioinformatics*, 23(18):2415–2422, 2007.
9. C. Belta and L. Habets. Controlling a class of nonlinear systems on rectangles. *IEEE Trans. of Automatic Control*, 51(11):1749–1759, 2006.
10. J. Burch, E. Clarke, K. McMillan, D. Dill, and L. Hwang. Symbolic model checking: 10^{20} states and beyond. In *LICS*, pages 428–439, 1990.
11. X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *In Proc. of CAV'13, the 25th International Conference on Computer Aided Verification*, volume 8044 of *LNCS*, pages 258–263, 2013.
12. E. M. Cherry and F. H. Fenton. Visualization of spiral and scroll waves in simulated and experimental cardiac tissue. *New Journal of Physics*, 10:125016, 2008.
13. T. Dang, A. Donzé, O. Maler, and N. Shalev. Sensitive state-space exploration. In *CDC*, pages 4049–4054, 2008.
14. T. Dreossi and T. Dang. Parameter synthesis for polynomial biological models. In *Proc. of HSCC 2014: the 17th International Conference on Hybrid Systems: Computation and Control*, HSCC '14, pages 233–242, New York, NY, USA, 2014. ACM.
15. G. Frehse, C. L. Guernic, A. Donz, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable verification of hybrid systems. In *In Proc. of CAV'11, the 23rd International Conference on Computer Aided Verification*, LNCS, Cliff Lodge, Snowbird, July 2011. Springer Verlag.
16. S. Gao, J. Avigad, and E. M. Clarke. δ-complete decision procedures for satisfiability over the reals. In *Proc. of IJCAR'12: the 6th International Joint Conference on Automated Reasoning*, volume 7364 of *LNCS*, pages 286–300. Springer, 2012.
17. S. Gao, S. Kong, and E. M. Clarke. dreal: An SMT solver for nonlinear theories over the reals. In *In Proc. of CADE'13, the 24th International Conference on Automated Deduction*, volume 7898 of *LNCS*, pages 208–214, 2013.

18. R. Grosu, G. Batt, F. Fenton, J. Glimm, C. L. Guernic, S. Smolka, and E. Bartocci. From cardiac cells to genetic regulatory networks. In *In Proc. of CAV'11, the 23rd International Conference on Computer Aided Verification*, LNCS, Cliff Lodge, Snowbird, July 2011. Springer Verlag.
19. L. C. G. J. M. Habets, P. J. Collins, and J. H. van Schuppen. Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *IEEE Trans. Automat. Contr.*, 51(6):938–948, 2006.
20. M. Kloetzer and C. Belta. Reachability analysis of multi-affine systems. In *Hybrid Systems: Computation and Control*, pages 348–362. Springer, 2006.
21. S. A. Kripke. Semantical considerations on modal logic. 1971.
22. B. Liu, S. Kong, S. Gao, P. Zuliani, and E. M. Clarke. Parameter synthesis for cardiac cell hybrid models using delta-decisions. *CoRR*, abs/1407.1524, 2014.
23. C. J. Myers. *Engineering genetic circuits.* Chapman and Hall/CRC, 2010.