Parametric Verification and Test Coverage for Hybrid Automata Using the Inverse Method

Laurent Fribourg and Ulrich Kühne

LSV ENS de Cachan, 94235 Cachan, France
{kuehne,fribourg}@lsv.ens-cachan.fr

Abstract. Hybrid systems combine continuous and discrete behavior. Hybrid Automata are a powerful formalism for the modeling and verification of such systems. A common problem in hybrid system verification is the good parameters problem, which consists in identifying a set of parameter valuations which guarantee a certain behavior of a system. Recently, a method has been presented for attacking this problem for Timed Automata. In this paper, we show the extension of this methodology for hybrid automata with linear and affine dynamics. The method is demonstrated with a hybrid system benchmark from the literature.

1 Introduction

Hybrid systems combine continuous and discrete behavior. They are especially useful for the verification of embedded systems, as they allow the unified modeling and the interaction of discrete control and the continuous environment or system state such as position, temperature or pressure.

There are several classes of formal models for hybrid systems. In general, there is a trade-off between the expressivity of the model and the complexity of the algorithmic apparatus that is needed for its formal analysis. Linear Hybrid Automata (LHA) provide a good compromise. In contrast to more general hybrid automata models, which allow arbitrary dynamics of the continuous state variables, LHA are restricted to linear dynamics. This allows the use of efficient algorithms based on convex polyhedra. Furthermore, more complex dynamics – like hybrid automata with affine dynamics (AHA) – can easily be approximated conservatively by LHA. Although reachability is undecidable for LHA [12], practically relevant results have been obtained using this formalism [11].

For the modeling of embedded systems it is handy to use *parameters* either to describe uncertainties or to introduce tuning parameters that are subject to optimization. Instead of setting these parameters manually and then verifying the resulting concrete system, parameterized models are used to perform automatic *parameter synthesis*. A common assumption is the existence of a set of bad states that should never be reached. Then the parameter synthesis can be solved by treating the parameters as additional state variables and computing the reachable states of the parameterized system in a standard manner[11]. However, this standard approach is not feasible except for very simple cases. It is therefore essential to dynamically prune the search space. The method presented in [9] is based on the CEGAR approach, iteratively refining a constraint over the parameters by discarding states that violate a given property. A similar refinement scheme has already been used for (non-parameterized) reachability problems of hybrid systems (see e.g. [14]), starting with an abstraction and refining until the property has been proved or a counterexample has been found.

While these traditional approaches to parameter synthesis are based on the analysis of bad states or failure traces, a complementary – or *inverse* – method has been proposed in [4]. It uses a parameter instantiation that is known to guarantee a *good* behavior in order to derive a constraint on the parameters that leads to the same behavior. While the algorithm in [4] is restricted to Timed Automata (TA), we present its extension to LHA in this paper.

There are different scenarios for the application of the presented approach. If a given parameter instantiation is known to guarantee certain properties, the inverse method can be used to derive an enlarged area of the parameter space that preserves these properties, while possibly allowing for enhanced performance of the system. In the The inverse method can also be used to obtain a measure of *coverage* of the parameter space by computing the zones of equivalent behavior for each point. This approach is also known as *behavioral cartography* [5] and will be discussed in this paper. While the natural extension of these algorithms works well for simple LHA, it does not scale well to LHA models that approximate more complex dynamics. Therefore, we present an enhanced algorithm that can be applied on affine hybrid automata.

The presented algorithms are implemented in a tool called IMITATOR (*Inverse Method for Inferring Time AbstracT behaviOR*) [3]. The tool has originally been developed for the analysis of TA. The new version IMITATOR 3 implements the semantics of LHA as presented in Sect. 3. The manipulation of symbolic states is based on the polyhedral operations of the Parma Polyhedra Library [6].

Throughout the paper, we will use a running example – a distributed temperature control system – to illustrate the presented concepts. Further examples can be found in [10].

2 Related Work

The presented approach exhibits the same general differences with the CEGAR-based approach of [9] at the LHA level as formerly at the TA level. First, the input of CEGAR-based methods is a bad location to be avoided while the input of our inverse method is a good reference valuation for the parameters; second, the constraint in CEGAR-based methods guarantees the avoidance of bad locations while the constraint generated by the inverse method guarantees the same behavior (in terms of discrete moves) as under the reference valuation.

Additionally, our inverse method based approach for LHA is comparable to the symbolic analysis presented in [1] for improving the simulation coverage of hybrid systems. In their work, Alur et al. start from an initial state x and a discrete-time simulation trajectory, and compute a constraint describing those initial states that are guaranteed to be equivalent to x, where two initial states are considered to be equivalent if the resulting trajectories contain the same locations at each discrete step of execution. The same kind of constraint can be generated by our inverse method when initial values of the continuous variables are defined using parameters. The two methods are however methodologically different. On the one hand, the generalization process done by the inverse method works, using forward analysis, by refining the current constraint over the parameters that repeatedly discards the generated states that are incompatible with the initial valuation of x; on the other hand, the method of Alur et al. generalizes the initial value of x by performing a backward propagation of sets of equivalent states. This latter approach can be practically done because the system is supposed to be *deterministic*, thus making easy the identification of transitions between discrete states during the execution. Our inverse method, in contrast, can also treat nondeterministic systems.

The approach presented in [15] shares a similar goal, namely identifying for single test cases a robust environment that leads to the same qualitative behavior. Instead of using symbolic reachability techniques, their approach is based on the stability of the continuous dynamics. By using a bisimulation function (or contraction map), a robust neighborhood can be constructed for each test point. As traditional numeric simulation can be used, this makes the technique computationally effective. But, for weakly stable systems, a lot of test points have to be considered in order to achieve a reasonable coverage. For some of the examples in [15], we achieve better or comparable results (see [10]).

3 Hybrid Automata with Parameters

3.1 Basic Definitions

In the sequel, we will refer to a set of continuous variables $X = x_1, \ldots, x_N$ and a set of parameters $P = p_1, \ldots, p_M$. Continuous variables can take any real value. We define a valuation as a function $w : X \to \mathbb{R}$, and the set of valuations over variables X is denoted by $\mathcal{V}(X)$. A valuation w will often be identified with the point $(w(x_1), \ldots, w(x_N)) \in \mathbb{R}^N$. A parameter valuation is a function $\pi : P \to \mathbb{R}$ mapping the parameters to the real numbers.

Given a set of variables X, a linear inequality has the form $\sum_{i=1}^{N} \alpha_i x_i \bowtie \beta$, where $x_i \in X$, $\alpha_i, \beta \in \mathbb{Z}$ and $\bowtie \in \{<, \leq, =\}$. A convex linear constraint is a finite conjunction of linear inequalities. The set of convex linear constraints over X is denoted by $\mathcal{L}(X)$. For a constraint $C \in \mathcal{L}(X)$ satisfied by a valuation $w \in \mathcal{V}(X)$, we write $w \models C$. For a constraint over continuous variables and parameters $C \in \mathcal{L}(X \cup P)$ satisfied by a valuation w and a parameter valuation π, we write $\langle w, \pi \rangle \models C$. By convention, we also write $w \models C$ for partial valuations. For example, a valuation $w \in \mathcal{V}(X)$ is said to satisfy a constraint $C \in \mathcal{L}(X \cup P)$ iff it can be extended with at least one parameter valuation π such that $\langle w, \pi \rangle \models C$.

Sometimes we will refer to a variable domain X', which is obtained by renaming the variables in X. Explicit renaming of variables is denoted by the substitution operation. Here, $(C)_{[X/Y]}$ denotes the constraint obtained by replacing in C the variables of X by the variables of Y.

A convex linear constraint can also be interpreted as a set of points in the space $\mathbb{R}^N$, more precisely as a convex polyhedron. We will use these notions synonymously. In this geometric context, a valuation satisfying a constraint is equivalent to the polyhedron containing the corresponding point, written as $w \in C$. Also here, for a partial valuation w (i.e. a point of a subspace of C), we write $w \in C$ iff w is contained in the projection of C on the variables of w.

Definition 1. *Given a set of continuous variables X and a set of parameters P, a* (parameterized) hybrid automaton *is a tuple $\mathcal{A} = (\Sigma, Q, q_0, I, D, \rightarrow)$, consisting of the following*

- *a finite set of actions Σ*
- *a finite set of locations Q*
- *an initial location $q_0 \in Q$*
- *a convex linear invariant $I_q \in \mathcal{L}(X \cup P)$ for each location q*
- *an activity $D_q : \mathbb{R}^n \rightarrow \mathbb{R}^n$ for each location q*
- *discrete transitions $q \xrightarrow{g,a,\mu} q'$, with guard condition $g \in \mathcal{L}(X \cup P)$, action $a \in \Sigma$ and a jump relation $\mu \in \mathcal{L}(X \cup P \cup X')$.*

Given a parameter constraint $K \in \mathcal{L}(P)$, the automaton $\mathcal{A}$ with the parameters restricted to K is denoted by $\mathcal{A}(K)$. Given a parameter valuation π, the automaton $\mathcal{A}$ with all parameters instantiated as in π is denoted by $\mathcal{A}[\pi]$.

Without loss of generality, it is assumed here that all continuous variables x are initialized with $x = 0$. Arbitrary initial values can be modeled by adding a transition with appropriate variable updates. Parameters can be seen as additional state variables which do not evolve in time (null activity).

The activities D_q describe how the continuous variables evolve within each location q. In order to obtain automata models which can be symbolically analyzed, restrictions have to be made to these activities. This leads to the following classes of hybrid automata.

Definition 2. *We define the following subclasses of hybrid automata.*

(1) A linear hybrid automaton *(LHA) is a hybrid automaton, where in each location q, the activity is given by a convex linear constraint $D_q \in \mathcal{L}(\dot{X})$ over the time derivatives of the variables.*
(2) An affine hybrid automaton *(AHA) is a hybrid automaton, where in each location q, the activity is given by a convex linear constraint $D_q \in \mathcal{L}(X \cup \dot{X})$ over the variables and the time derivatives.*

The class of timed automata can be obtained by restricting the derivatives to $\dot{x} = 1$ and limiting the jump relations to either $x' = x$ or $x' = 0$ (clock reset) for all variables $x \in X$. In total, the automata models defined above form a hierarchy $TA \subset LHA \subset AHA$.

The reachable states of LHA can be efficiently represented by convex polyhedra. Due to the more complex dynamics, this is not true for AHA. In the following, we consider linear hybrid automata with parameters. But, AHA can be approximated by LHA with arbitrary precision by partitioning the state space, as e.g. described in [8]. In Sect. 4.3 it is discussed, how these techniques can be adapted to suit our methods. In the following, we give an example of a hybrid system, that will later on be used to illustrate the approaches proposed here.

Example 1. The *room heating benchmark* (RHB) has been described in [7]. It models a distributed temperature control system. There are m movable heaters for $n > m$ rooms. The temperature x_i in each room i is a continuous variable that depends on the (constant) outside temperature u, the temperature of the adjacent rooms, and whether there is an activated heater in the room.

Depending on the relations between the temperatures measured, the heaters will be moved. If there is no heater in room i, a heater will be moved there from an adjacent room j, if the temperature has reached a threshold $x_i \leq get_i$ and there is a minimum difference of the temperatures $x_j - x_i \geq dif_i$. Note that in contrast with the RHB modeled in [1], the heater move from a room to another one is nondeterministic, since multiple guard conditions can be enabled simultaneously (in [1], the nondeterminism is resolved by moving only the heater with the smallest index). The dynamics is given by equations of the form:

$$\dot{x}_i = c_i h_i + b_i(u - x_i) + \sum_{i \neq j} a_{i,j}(x_j - x_i) \tag{1}$$

where $a_{i,j}$ are constant components of a symmetric adjacency matrix, constants b_i and c_i define the influence of the outside temperature and the effectiveness of the heater for each room i, and $h_i = 1$ if there is a heater in room i and $h_i = 0$ otherwise. Here, we will study an instantiation of RHB as given in [1] with $n = 3, m = 2$, outside temperature $u = 4$, the constants $b = (0.4, 0.3, 0.4), c = (6, 7, 8)$. The adjacency matrix $a_{i,j}$ is given as $\begin{pmatrix} 0.0 & 0.5 & 0.0 \\ 0.5 & 0.0 & 0.5 \\ 0.0 & 0.5 & 0.0 \end{pmatrix}$ and the thresholds are set to $get = 18$ and $dif = 1$ for all rooms.

The system can be modeled as an AHA, as shown in Fig. 1. There are three control modes, corresponding to the positions of the two heaters. The automaton has four variables, the temperatures $X = \{x_1, x_2, x_3\}$ and a variable t acting as clock. In this example, the temperatures are sampled at a constant rate $\frac{1}{h}$, where h is a parameter of the automaton. This sampling scheme is used in the models of sampled-data hybrid systems of [16] and simulink/stateflow models [1].

3.2 Symbolic semantics

The symbolic semantics of a LHA $\mathcal{A}(K)$ are defined at the level of constraints, a symbolic state is a pair (q, C) of a location q and a constraint C over variables and parameters. The corresponding operations are therefore performed on convex polyhedra rather than on concrete valuations. One necessary operation is the progress of time within a symbolic state, modeled by the *time-elapse* operation.

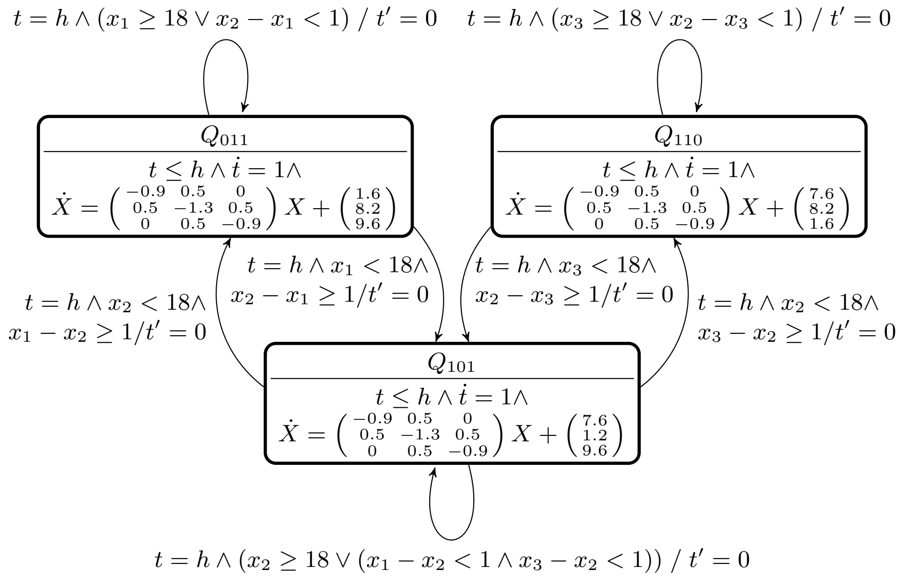

$t = h \wedge (x_1 \geq 18 \vee x_2 - x_1 < 1) \;/\; t' = 0$
$t = h \wedge (x_3 \geq 18 \vee x_2 - x_3 < 1) \;/\; t' = 0$
Q_{011}
$t \leq h \wedge \dot{t} = 1 \wedge$
$\dot{X} = \begin{pmatrix} -0.9 & 0.5 & 0 \\ 0.5 & -1.3 & 0.5 \\ 0 & 0.5 & -0.9 \end{pmatrix} X + \begin{pmatrix} 1.6 \\ 8.2 \\ 9.6 \end{pmatrix}$
Q_{110}
$t \leq h \wedge \dot{t} = 1 \wedge$
$\dot{X} = \begin{pmatrix} -0.9 & 0.5 & 0 \\ 0.5 & -1.3 & 0.5 \\ 0 & 0.5 & -0.9 \end{pmatrix} X + \begin{pmatrix} 7.6 \\ 8.2 \\ 1.6 \end{pmatrix}$
$t = h \wedge x_1 < 18 \wedge$
$x_2 - x_1 \geq 1 / t' = 0$
$t = h \wedge x_3 < 18 \wedge$
$x_2 - x_3 \geq 1 / t' = 0$
$t = h \wedge x_2 < 18 \wedge$
$x_1 - x_2 \geq 1 / t' = 0$
$t = h \wedge x_2 < 18 \wedge$
$x_3 - x_2 \geq 1 / t' = 0$
Q_{101}
$t \leq h \wedge \dot{t} = 1 \wedge$
$\dot{X} = \begin{pmatrix} -0.9 & 0.5 & 0 \\ 0.5 & -1.3 & 0.5 \\ 0 & 0.5 & -0.9 \end{pmatrix} X + \begin{pmatrix} 7.6 \\ 1.2 \\ 9.6 \end{pmatrix}$
$t = h \wedge (x_2 \geq 18 \vee (x_1 - x_2 < 1 \wedge x_3 - x_2 < 1)) \;/\; t' = 0$

Fig. 1. Automaton model for room heating benchmark

Definition 3. *Given a symbolic state (q, C), the states reached by letting t time units elapse, while respecting the invariant of q, are characterized as follows:*

$$w' \in C \uparrow_q^t \quad \text{iff} \quad \exists w \in C, v \in D_q : w' = w + t \cdot v \wedge w' \in I_q.$$

We write $w' \in C \uparrow_q$ if $w' \in C \uparrow_q^t$ for some $t \in \mathbb{R}_+$.

Note that due to the convexity of the invariants, if $C \subseteq I_q$ and $C \uparrow_q^t \subseteq I_q$, then also $\forall t' \in [0, t] : C \uparrow_q^{t'} \subseteq I_q$. The operator preserves the convexity of C. Furthermore, the operator $C \downarrow_X$ denotes the projection of the constraint C on the variables in X. Based on these definitions, the symbolic semantics of a LHA $\mathcal{A}(K)$ is given by a labeled transition system (LTS).

Definition 4. *A labeled transition system over a set of symbols Σ is a triple $(S, S_0, \Rightarrow)$ with a set of states S, a set of initial states $S_0 \subseteq S$ and a transition relation $\Rightarrow \subseteq S \times \Sigma \times S$. We write $s \overset{a}{\Rightarrow} s'$ for $(s, a, s') \in \Rightarrow$. A run of length m is a finite alternating sequence of states and symbols of the form $s_0 \overset{a_0}{\Rightarrow} s_1 \overset{a_1}{\Rightarrow} \dots \overset{a_{m-1}}{\Rightarrow} s_m$, where $s_0 \in S_0$.*

Definition 5. *The symbolic semantics of LHA $\mathcal{A}(K)$ is a LTS with*

- *states $S = \{(q, C) \in Q \times \mathcal{L}(X \cup P) \mid C \subseteq I_q\}$*
- *initial state $s_0 = (q_0, C_0)$ with $C_0 = K \wedge [\bigwedge_{i=1}^{N} x_i = 0] \uparrow_{q_0}$*
- *discrete transitions $(q, C) \xrightarrow{a} (q', C')$ if exists $q \xrightarrow{a,g,\mu} q'$ and $C' = \big([C(X) \wedge g(X) \wedge \mu(X, X')] \downarrow_{X'} \wedge I_{q'}(X') \big)_{[X'/X]}$*

– *delay transitions* $(q, C) \xrightarrow{t} (q, C')$ *with* $C' = C \uparrow_q^t$
– *transitions* $(q, C) \overset{a}{\Rightarrow} (q', C')$ *if* $\exists t, C'' : (q, C) \xrightarrow{a} (q', C'') \xrightarrow{t} (q', C')$

The *trace of a symbolic run* $(q_0, C_0) \overset{a_0}{\Rightarrow} \dots \overset{a_{m-1}}{\Rightarrow} (q_m, C_m)$ is obtained by projecting the symbolic states to the locations, which gives: $q_0 \overset{a_0}{\Rightarrow} \dots \overset{a_{m-1}}{\Rightarrow} q_m$. Two runs are said to be *equivalent*, if their corresponding traces are equal.

The set of states reachable from any state in a set S in exactly i steps is denoted as $Post^i_{\mathcal{A}(K)}(S) = \{s' \mid \exists s \in S : s \overset{a_0}{\Rightarrow} \dots \overset{a_{i-1}}{\Rightarrow} s'\}$. Likewise, the set of all reachable states from S is defined as $Post^*_{\mathcal{A}(K)}(S) = \bigcup_{i \geq 0} Post^i_{\mathcal{A}(K)}$. The reachable states of an automaton $\mathcal{A}(K)$ are defined as $Reach_{\mathcal{A}(K)} = Post^*_{\mathcal{A}(K)}(\{s_0\})$, where s_0 is the initial state of $\mathcal{A}(K)$.

Note that during a run of $\mathcal{A}(K)$, the parameter constraints associated to the reachable states can only get stronger, since the parameters do not evolve under the time elapse operation, and can only be further constrained by invariants or guard conditions. This gives rise to the following observation.

Lemma 1. *For any reachable state* $(q, C) \in Reach_{\mathcal{A}(K)}$*, it holds that* $(\exists X : C) \subseteq K$*. This implies that for each parameter valuation* $\pi \models C$*, also* $\pi \models K$*.*

The lemma follows directly from the definition of the symbolic semantics. We say that a state (q, C) is *compatible* with a parameter valuation π, or just π*-compatible*, if $\pi \models C$. Conversely, it is π*-incompatible* if $\pi \not\models C$. These observations are the basis for the *Inverse Method*, is described in next section.

4 Algorithm

4.1 Inverse Method

The Inverse Method for LHA attacks the good parameters problem by generalizing a parameter valuation π that is known to guarantee a good behavior. Thereby, the valuation π is relaxed to a constraint K such that the *discrete behavior* – i.e. the set of traces – of $\mathcal{A}[\pi]$ and $\mathcal{A}(K)$ is identical. The algorithm has first been described for parametric timed automata in [4]. It has been applied for the synthesis of timing constraints for memory circuits [2].

Algorithm 1 describes the Inverse Method for LHA. The overall structure is similar to a reachability analysis. In the main loop, the reachable states with increasing depth i are computed. In parallel, the constraint K is derived. It is initialized with `true`. Each time a π-incompatible state (q, C) is reached, K is refined such that the incompatible state is unreachable for $\mathcal{A}(K)$. If C is π-incompatible, then there must be at least one inequality J in its projection on the parameters $(\exists X : C)$, which is incompatible with π. The algorithm selects one such inequality and adds its negation $\neg J$ to the constraint K. Before continuing with the search, the reachable states found so far are updated to comply with the new constraint K (line 7). If there are no more π-incompatible states, then i is increased and the loop continues.

Algorithm 1: *IM*$(\mathcal{A}, \pi)$

```
input  : Parametric linear hybrid automaton $\mathcal{A}$
input  : Valuation $\pi$ of the parameters
output: Constraint $K_0$ on the parameters
1  $i \leftarrow 0$ ;  $K \leftarrow$ true ;  $S \leftarrow \{s_0\}$
2  while true do
3      while there are $\pi$-incompatible states in $S$ do
4          Select a $\pi$-incompatible state $(q, C)$ of $S$ (i.e., s.t. $\pi \not\models C$) ;
5          Select a $\pi$-incompatible inequality $J$ in $(\exists X : C)$ (i.e., s.t. $\pi \not\models J$) ;
6          $K \leftarrow K \wedge \neg J$ ;
7          $S \leftarrow \bigcup_{j=0}^{i} Post^{j}_{\mathcal{A}(K)}(\{s_0\})$ ;
8      if $Post_{\mathcal{A}(K)}(S) \sqsubseteq S$ then return $K_0 \leftarrow \bigcap_{(q,C)\in S}(\exists X : C)$
9      $i \leftarrow i + 1$ ;
10     $S \leftarrow S \cup Post_{\mathcal{A}(K)}(S)$
```

The algorithm stops as soon as no new states are found (line 8). The output of the algorithm is then a parameter constraint K_0, obtained as the intersection of the constraints associated with the reachable states. The resulting constraint can be characterized as follows.

Proposition 1. *Suppose that the algorithm* $IM(\mathcal{A}, \pi_0, k)$ *terminates with the output* K_0*. Then the following holds:*

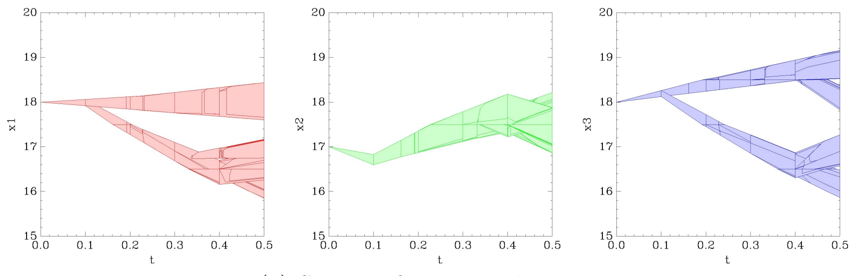
- $\pi_0 \models K_0$
- *For all* $\pi \models K_0$*,* $\mathcal{A}[\pi_0]$ *and* $\mathcal{A}[\pi]$ *have the same sets of traces.*

A proof along the lines of [13] can be found in [10]. We obtain a (convex) constraint K_0 including the initial point π_0, that describes a set of parameter valuations for which the same set of traces is observable. In particular, if $\mathcal{A}[\pi_0]$ is known to avoid a set of (bad) locations for π_0, so will $\mathcal{A}[\pi]$ for any $\pi \models K_0$.

The algorithm *IM* is not guaranteed to terminate[1]. Note also that the presented algorithm involves nondeterminism. In Algorithm 1 in lines 4 and 5, one can possibly choose among several incompatible states and inequalities. This may lead to different – nevertheless correct – results. This implies in particular that the resulting constraint K_0 is not maximal in general. (In order to overcome this limitation, the *behavioral cartography* method will be proposed in Section 4.2).

Example 2. In order to enable the application of the inverse method as described above to the RHB from example 1, the AHA automaton is converted to a LHA. This is done using the method described in [8]. The space is partitioned into regions, and within each region, the activity field is overapproximated using

[1] Termination of such a general reachability-based procedure cannot be guaranteed due to undecidability of reachability for TA with parameters and LHA [12]


(a) Starting from a single point

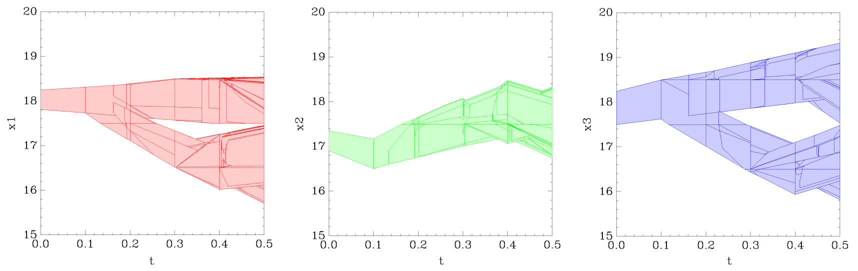

(b) Starting from a tile synthesized by the Inverse Method

Fig. 2. Reachable states for room heating benchmark

linear sets of activity vectors. For each region R delimiting a portion of the partitioned state space, the activities are statically overapproximated as

$$\dot{x}_i \in [min\{f_i(x) \mid x \in R\}, max\{f_i(x) \mid x \in R\}] ,$$

where $f_i(x)$ corresponds to the right-hand side in (1). The approximation can be made arbitrarily accurate by approximating over suitably small regions of the state space. Here, each region R corresponds to a unit cube (of size 1 degree Celsius) in the dimensions x_1, x_2, x_3.

We now consider the following (bounded liveness) property:

Prop1: At least one of the heaters will be moved within a given time interval $[0, t_{max}]$ with $t_{max} = \frac{1}{2}$ and a sampling time $h = \frac{1}{10}$.

The upper bound t_{max} plays here the role of the maximal number of discrete transitions that are used in the method of [1]. In the automaton model, a violation of the property is modeled by a transition to a location q_{bad}. To check the property *Prop1* for varying initial conditions, we add the parameters a_1, a_2, a_3 and constrain the initial state with $x_1 = a_1 \wedge x_2 = a_2 \wedge x_3 = a3$. For initial point $(a_1, a_2, a_3) = (18, 17, 18)$, the reachable states for the variables x_1, x_2 and x_3 are shown in Fig. 2(a). The bad location is not reached from this point. Using the Inverse Method (Algorithm 1), the initial point can be generalized to a larger region around the starting point $(18, 17, 18)$, resulting in the constraint

$$a_1 \geq a_2 + \tfrac{181}{200} \wedge a_1 < \tfrac{a_3}{2} + \tfrac{37}{4} \wedge a_2 > \tfrac{3381}{200} \wedge a_2 < \tfrac{35}{2} \wedge a_3 > \tfrac{35}{2} \wedge a_3 < \tfrac{456}{25} .$$

Algorithm 2: *BC*

```
input : Parametric linear hybrid automaton $\mathcal{A}$
input : Parameter bounds $min_1 \ldots min_M$ and $max_1 \ldots max_M$
input : Step sizes $\delta_1 \ldots \delta_M$
output: Set of constraints $Z$ on the parameters

1 $Z \leftarrow \varnothing$
2 $V \leftarrow \{\pi \mid \pi_i = min_i + \ell_i \cdot \delta_i,\ \pi_i \leq max_i,\ \ell_1, \ldots, \ell_M \in \mathbb{N}\}$
3 while true do
4     Select point $\pi \in V$ with $\forall K \in Z : \pi \not\models K$
5     $K \leftarrow IM(\mathcal{A}, \pi)$
6     $Z \leftarrow Z \cup \{K\}$
7     if $\forall \pi \in V : \exists K \in Z : \pi \models K$ then
8         return $Z$
```

The symbolic runs starting from this enlarged initial region are depicted in Fig. 2(b). The sets of traces of the two figures coincide, i.e. the sequence of discrete transitions of every run represented in Fig. 2(b) is identical to the sequence of discrete transitions of some run in Fig. 2(a).

4.2 Behavioral Cartography

The inverse method works efficiently in many cases, since large parts of the state space can effectively be pruned by refining the parameter constraint K. In this way, many bad states never have to be computed, in contrast to the traditional approach to parameter synthesis. A drawback of the inverse method is that the notion of equivalence of the traces may be too strict for some cases. If e.g. one is interested in the non-reachability of a certain bad state, then there may exist several admissible regions in the parameter space that differ in terms of the discrete behavior or trace-sets. In order to discover these regions, the inverse method needs to be applied iteratively with different starting points.

The systematic exploration of the parameter space using the inverse method is called *behavioral cartography* [5]. It works as shown in Algorithm 2. For each parameter p_i, the interval $[min_i, max_i]$, possibly containing a single point, specifies the region of interest. This results in a rectangular zone $v_0 = [min_1, max_1] \times \cdots \times [min_M, max_M]$. Furthermore, step sizes $\delta_i \in \mathbb{R}$ are given. The algorithm selects (yet uncovered) points defined by the region v_0 and the step sizes and calls the inverse method on them. The set Z contains the tiles (i.e. parameter constraints) computed so far. The algorithm proceeds until all starting points are covered by some tile $K \in Z$.

By testing the inclusion in some computed tile, repeated computations are avoided for already covered points. The result of the cartography is a set of tiles of the parameter space, each representing a distinct behavior of the LHA $\mathcal{A}$. Note that the computed tiles do not necessarily cover the complete region v_0. On the other hand, it is possible that v_0 be covered by very few calls to

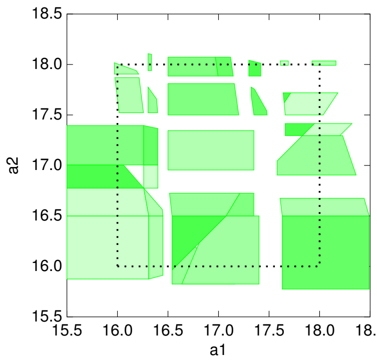


Fig. 3. Cartography of the initial states of RHB

the inverse method. Note also that, compared to the algorithm in [1], this is a stronger result, as each tile corresponds to a *set* of traces that exploits all possible behavior for the covered parameter valuations, including nondeterminism.

Example 3. The cartography is illustrated by a further experiment on the RHB model from example 2. Again, we check *Prop1*. The initial point is varied for the initial values a_1 and a_2, while fixing $a_3 = 18$. Therefore, the cartography procedure is used, iterating the initial point within the rectangle $[16, 18]^2$ (i.e, $min_1 = min_2 = 16$ and $max_1 = max_2 = 18$) with a step size of $\delta_1 = \delta_2 = \frac{1}{3}$. This leads to a total of 32 tiles, shown in Fig. 3. By analyzing the cartography, one gets a quantitative measure of the coverage of the considered region (shown as a dashed rectangle in the figure). In this case, the computed tiles cover 56% of the rectangle. All tiles in the figure have been classified as good tiles.

4.3 Enhancement of the method for affine dynamics

It can be observed that for some systems there are areas in the parameter space, where slight variations of the initial conditions lead to many different traces. In this case, a good coverage based the cartography approach will be very costly, since many points have to be considered. In general, the inverse method and the behavioral cartography is quite limited when applied to LHA models that were obtained from AHA by static partitioning.

As described in [8], AHA can be approximated by LHA with arbitrary precision. This is done by partitioning the invariant of a location, usually into a set of small rectangular regions. For each region R, the affine dynamics are over-approximated by linear dynamics. In this way, the locations are split up until the desired precision is obtained.

Due to this partitioning, the resulting LHA will have more locations than the original AHA, leading also to more different traces for each parameter instantiation. This renders the inverse method ineffective for AHA, as the region around a parameter valuation π that corresponds to the same trace set, will generally be very small. This is because the traces contain a lot of information on the transitions between partitions that are irrelevant wrt. the system's behavior.

These limitations can be overcome by grouping reachable states that only represent different partitions of the same invariant of a location q. In our algorithm, this is done as an extension of the time-elapse operator. Each time that the time-elapse $C \uparrow_q$ needs to be computed for a location with affine dynamics D_q, the following steps are performed:

1. Build local partitions P of the invariant I_q
2. Compute a linear over-approximation $\hat{D}_P$ of D_q for each partition P
3. Compute the locally reachable states S wrt. partitions P and dynamics $\hat{D}_P$
4. Compute the convex hull of the states S

Here, the number of partitions Δ per dimension is chosen by the user. Note that cost and precision of the overall analysis may strongly depend on the chosen value for Δ. In practice, one would iterate the methods presented in this paper in order to refine the analysis by increasing Δ.

Given this variant of the time-elapse for affine dynamics, the computed reachable states are an over-approximation due to the piecewise linearization of the dynamics and the convex hull operation. Thus, the trace equivalence is no longer valid. But, as we compute an over-approximation of the possible runs, non-reachability is preserved.

Proposition 2. *Given an AHA $\mathcal{A}$, suppose that the algorithm $IM(\mathcal{A}, \pi_0, k)$ terminates with the output K_0. Then the following holds:*

– $\pi_0 \models K_0$
– *If for $\mathcal{A}[\pi_0]$, a location q_{bad} is unreachable, then it is also unreachable for all $\mathcal{A}[\pi]$ with $\pi \models K_0$*

Example 4. The adapted algorithm is applied to the RHB. With the discussed techniques, we can apply the inverse method and thus the cartography directly on the AHA model, without statically partitioning the state space in order to obtain a LHA. Again, by repeating the inverse method, a large part of the system's initial state space is decomposed into tiles of distinct discrete behavior. The reachability analysis for the AHA model is quite costly. Therefore, we will try to cover large parts of the parameter space using a very coarse linearization, given by a small number Δ of partitions. This is illustrated in the following. As reported in Example 3, applying the cartography on the statically linearized RHB model delivers a coverage of only 56% when fixing $a_3 = 18$. Instead, we apply the enhanced method directly on the AHA model, again regarding property *Prop1*. Here, the initial values a_1 and a_2 are varied within the rectangle $[15.5, 18.5]^2$ (i.e, $min_1 = min_2 = 15.5$ and $max_1 = max_2 = 18.5$) with a step size of $\delta_1 = \delta_2 = \frac{1}{2}$. In the first step, the invariants will be uniformly linearized, i.e. we set $\Delta = 1$. The resulting cartography in Fig. 4 consists of 12 tiles, where the good ones are shown in green, while the tiles corresponding to a bad behavior are shown in red (and outlined in bold). Note that the whole rectangular region is covered and that already with a coarse linearization, most of the tiles could be proved good.

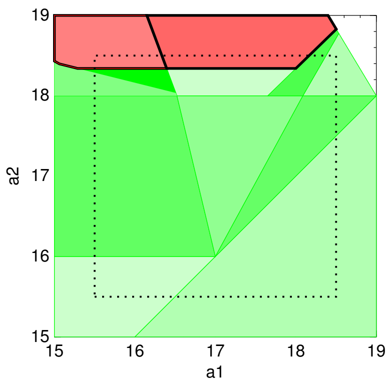


Fig. 4. Enhanced cartography for room heating benchmark

5 Final Remarks

In this paper, we present a method to derive parameter constraints for LHA, that guarantee the same behavior as for a reference valuation of the parameters. This method has been recently introduced for deriving timing constraints for timed automata. Here, we provide the extension of the method to LHA. Furthermore, it is shown how the reachability procedure can be adapted to enable the analysis of systems with affine dynamics. By early pruning of invalid states, the method is more efficient than the parameter synthesis based on standard reachability analysis. Repeated analysis for different starting points yields a "behavioral cartography". This allows to cover large parts of the initial state space of nondeterministic hybrid systems, and provides an alternative tool to the symbolic simulation method of [1], which gives sometimes better results.

References

1. R. Alur, A. Kanade, S. Ramesh, and K. Shashidhar. Symbolic analysis for improving simulation coverage of simulink/stateflow models. In *EMSOFT*, pages 89–98, 2008.
2. E. André. IMITATOR: A tool for synthesizing constraints on timing bounds of timed automata. In *Proc. of the Int'l Colloquium on Theoretical Aspects of Computing (ICTAC)*, volume 5684 of *LNCS*, pages 336–342. Springer, 2009.
3. É. André. IMITATOR II: A tool for solving the good parameters problem in timed automata. In *INFINITY*, volume 39 of *EPTCS*, pages 91–99, Sept. 2010.
4. E. André, T. Chatain, E. Encrenaz, and L. Fribourg. An inverse method for parametric timed automata. *IJFCS*, 20(5):819–836, Oct. 2009.
5. E. André and L. Fribourg. Behavioral cartography of timed automata. In *RP*, volume 6227 of *LNCS*, pages 76–90. Springer, 2010.
6. R. Bagnara, P. Hill, and E. Zaffanella. Applications of polyhedral computations to the analysis and verification of hardware and software systems. *Theoretical Computer Science*, 410(46):4672–4691, 2009.

7. A. Fehnker and F. Ivancic. Benchmarks for hybrid systems verification. In *HSCC*, pages 326–341. Springer, 2004.
8. G. Frehse. PHAVer: algorithmic verification of hybrid systems past HyTech. *STTT*, 10(3):263–279, 2008.
9. G. Frehse, S. Jha, and B. Krogh. A counterexample-guided approach to parameter synthesis for linear hybrid automata. In *HSCC*, volume 4981 of *LNCS*, 2008.
10. L. Fribourg and U. Kühne. Parametric verification of hybrid automata using the inverse method. Research Report LSV-11-04, LSV, ENS Cachan, France, 2011.
11. T. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A model checker for hybrid systems. *STTT*, 1:110–122, 1997.
12. T. Henzinger, P. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *JCSS*, pages 373–382, 1995.
13. T. Hune, J. Romijn, M. Stoelinga, and F. Vaandrager. Linear parametric model checking of timed automata. *JLAP*, 52-53:183 – 220, 2002.
14. S. Jha, B. Krogh, J. Weimer, and E. Clarke. Reachability for linear hybrid automata using iterative relaxation abstraction. In *HSCC*, volume 4416 of *LNCS*, pages 287–300, 2007.
15. A. Julius, G. Fainekos, M. Anand, I. Lee, and G. Pappas. Robust test generation and coverage for hybrid systems. In *HSCC*, volume 4416 of *LNCS*. 2007.
16. B. Silva and B. Krogh. Modeling and verification of sampled-data hybrid systems. In *ADPM*, 2000.