Parametric Verification of Hybrid Automata Using the Inverse Method

Laurent Fribourg and Ulrich Kühne

March 2011

Research report LSV-11-04

Parametric Verification of 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 subset of parameters which guarantee a certain behavior of a system. Recently, a method has been presented for attacking this problem for Timed Automata. In this report, we show the extension of this methodology for hybrid automata with linear and affine dynamics. The method is demonstrated with a distributed temperature control system and several other hybrid system benchmarks 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 [13], practically relevant results have been obtained using this formalism [18, 10].

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[12]. 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.

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 [3]. 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 [3] is restricted to Timed Automata (TA), we present its extension to LHA in this report.

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 same time, the *robustness* of the parameter choice can be proven. Since real world systems are often subject to uncertainties wrt. to environment conditions, it is not advisable to choose a parameter instantiation that lies on the very edge to malfunction. Finally, it can be used if the parameterized *verification* of a safety property is not feasible. In this case, a pointwise verification for a set of parameter instantiations is performed. The inverse method can then 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* [4] and will be discussed in this report. 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*) [2]. The tool has originally been developed for the analysis of TA. The new version IMITATOR 3 implements the semantics of LHA as presented in Section 3. The manipulation of symbolic states is based on the polyhedral operations of the Parma Polyhedra Library [5].

The report is structured as follows. First, we will discuss related work in Section 2. In Section 3, the formal basis for the rest of the report is given. The algorithms are introduced in Section 4. Experimental results are discussed in the course of the presentation, using as a running example a distributed temperature control system. More benchmarks from the literature are treated in Section 5. The results are discussed in Section 6 and the report is concluded in Section 7.

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 [16] 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 [16], we achieve better or comparable results (see Section 5.3).

Note that both [1] and [16] only consider the coverage of the initial states, while our approach can be applied in the more general context of parameter synthesis.

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

$$\sum_{i=1}^{N} \alpha_i x_i \bowtie \beta, \tag{1}$$

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 *with parameter constraint $K \in \mathcal{L}(P)$ is a tuple $\mathcal{A}(K) = (\Sigma, Q, q_0, I, D, \to)$, 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 $D_q : \mathbb{R}^n \to \mathbb{R}^n$ 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 $\mu \in \mathcal{L}(X \cup P \cup X')$*

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 rectangular automaton *(RA) is a hybrid automaton, where in each location q, the time derivative $\dot{x}_i$ for variable x_i is contained in an interval $\dot{x}_i \in [\ell_i^q, u_i^q]$, where $\ell_i^q, u_i^q \in \mathbb{R}$.*
(2) 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.*
(3) 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.*

Rectangular automata are a superset of timed automata (TA). 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 $TA \subset RA \subset LHA \subset AHA$.

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 Section 4.4 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 in this report.

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 of the system is given by equations of the form:

$$\dot{x}_i = c_i h_i + b_i(u - x_i) + \sum_{i \neq j} a_{i,j}(x_j - x_i) \tag{2}$$

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

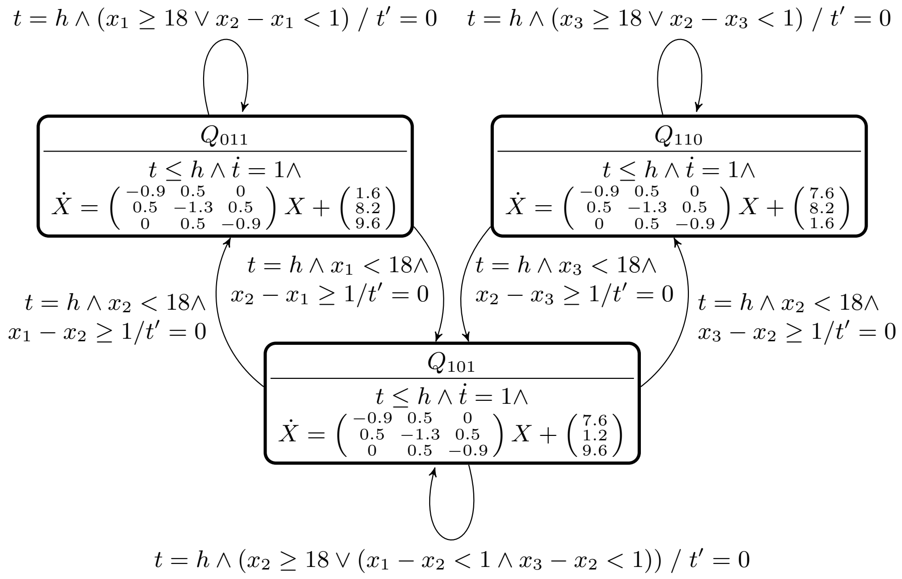

$t = h \wedge (x_1 \geq 18 \vee x_2 - x_1 < 1) \;/\; t' = 0$
$t = h \wedge (x_3 \geq 18 \vee x_2 - x_3 < 1) \;/\; t' = 0$
Q_{011}
$t \leq h \wedge \dot{t} = 1 \wedge$
$\dot{X} = \begin{pmatrix} -0.9 & 0.5 & 0 \\ 0.5 & -1.3 & 0.5 \\ 0 & 0.5 & -0.9 \end{pmatrix} X + \begin{pmatrix} 1.6 \\ 8.2 \\ 9.6 \end{pmatrix}$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

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 [17] and simulink/stateflow models [1].

3.2 Concrete semantics

In order to obtain a concrete run of a LHA $\mathcal{A}(K)$, all parameters need to be instantiated by a parameter valuation $\pi \models K$. The instantiated model is then denoted as $\mathcal{A}[\pi]$. The concrete semantics (or simulation semantics) is given by a labeled transition system (LTS).

Definition 3. *A labeled transition system over a set of symbols Σ is a triple $L = (S, S_0, \Rightarrow)$ 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 $s_0 \in S_0$. A state s_m is reachable if it is the last state of some run R.*

Definition 4. *The concrete semantics of an LHA $\mathcal{A}[\pi]$ is given by the LTS $L_H = (S, S_0, \Rightarrow)$ with*

– *states* $S = \{(q, w) \in Q \times \mathcal{V}(X) \mid \langle w, \pi \rangle \models I_q\}$
– *initial states* $S_0 = \{(q_0, w) \mid \langle w, \pi \rangle \models I_{q_0} \wedge \exists v, t : w = t \cdot v, t \in \mathbb{R}_+, v \in D_{q_0}\}$
– *discrete transitions* $(q, w) \xrightarrow{a} (q', w')$ *if*
$\exists g, \mu : q \xrightarrow{g,a,\mu} q'$ *and* $\langle w, \pi \rangle \models g$ *and* $\langle w, w', \pi \rangle \models \mu$
– *delay transitions* $(q, w) \xrightarrow{t} (q, w')$ *if*
$\exists v \in D_q : w' = w + t \cdot v$
– *transitions* $(q, w) \overset{a}{\Rightarrow} (q', w')$ *if*
$\exists t, w'' : (q, w) \xrightarrow{a} (q', w'') \xrightarrow{t} (q', w')$

The valuations of the initial states are obtained by letting time elapse from the initial valuation $\bigwedge_{i=1}^{N} x_i = 0$, while still satisfying the invariant of q_0. Finally, we define the notion of traces and their equivalence. A trace is obtained by projecting a run on the locations:

Definition 5. *For a LHA $\mathcal{A}$, the* trace associated to a concrete run $(q_0, w_0) \overset{a_0}{\Rightarrow} \dots \overset{a_{m-1}}{\Rightarrow} (q_m, w_m)$ *of $\mathcal{A}[\pi]$ is the alternating sequence of locations and actions* $q_0 \overset{a_0}{\Rightarrow} \dots \overset{a_{m-1}}{\Rightarrow} q_m$.

Definition 6. *Given a LHA $\mathcal{A}$ and two parameter instantiations π_1 and π_2, if the set of traces corresponding to the semantics of $\mathcal{A}[\pi_1]$ is equal to to the set of traces corresponding to the semantics of $\mathcal{A}[\pi_2]$, then $\mathcal{A}[\pi_1]$ and $\mathcal{A}[\pi_2]$ are said to be* trace equivalent.

3.3 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.

Definition 7. *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*

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

Note that due to the convexity of the invariants, if $C \subseteq I_q$ and $C \uparrow_q^t \subseteq I_q$, then also $\forall t' \in [0, t] : C \uparrow_q^{t'} \subseteq I_q$. To obtain any state respecting the invariant that is reached by letting some time elapse, we use the closure of the above operation.

Definition 8. *Given a symbolic state (q, C), the closure of the time-elapse operation is defined as follows*

$$w' \in C \uparrow_q \quad \textit{iff} \quad \exists t \in \mathbb{R}_+ : w' \in C \uparrow_q^t$$

Note that this operator preserves the convexity of C. Furthermore, the operator $C \downarrow_X$ denotes the projection of the constraint C on the variables in X, which is performed by existential quantification (e.g. using Fourier-Motzkin elimination) of the variables not contained in X. Based on these definitions, the symbolic semantics of a LHA $\mathcal{A}(K)$ is given by a labeled transition system (LTS) as follows.

Definition 9. *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* $q \overset{a,g,\mu}{\rightarrow} q'$ *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* $\exists t, C'' : (q, C) \xrightarrow{a} (q', C'') \xrightarrow{t} (q', C')$*, or as a closed formula:* $C' = \big([[C(X) \wedge g(X) \wedge \mu(X, X')] \downarrow_{X'} \wedge I_{q'}(X')] \uparrow_{q'} \big)_{[X'/X]}$

Analogously to the concrete semantics, the *trace of a symbolic run* $(q_0, C_0) \overset{a_0}{\Rightarrow} \ldots \overset{a_{m-1}}{\Rightarrow} (q_m, C_m)$ is obtained by projecting the symbolic states to the locations, which gives: $q_0 \overset{a_0}{\Rightarrow} \ldots \overset{a_{m-1}}{\Rightarrow} q_m$. Two runs (concrete or symbolic) 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 $Post^i_{\mathcal{A}(K)}(S)$. More formally, $Post^i_{\mathcal{A}(k)} = \{s' \mid \exists s \in S : s \overset{a_0}{\Rightarrow} \ldots \overset{a_{i-1}}{\Rightarrow} s'\}$. 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 $\mathcal{A}(K)$ are defined as $Reach_{\mathcal{A}(K)} = Post^*_{\mathcal{A}(K)}(\{s_0\})$, where s_0 is the initial state of $\mathcal{A}(K)$.

3.4 Relation between concrete and symbolic semantics

The concrete semantics captures every single run that is admissible in an automaton. While it is easy to see its relation to the automaton, it cannot be used for the formal verification of systems modeled as LHA, since the number of concrete runs is almost always infinite. Instead, the symbolic semantics allows for a more abstract view on the system by capturing sets of runs. It is based on linear constraints which can be interpreted as convex sets of valuations. For the manipulation of such constraints, there exist geometric algorithms operating on convex polyhedra [6, 11, 5].

More precisely, it can be shown that every concrete run can be simulated by a symbolic run and vice versa. The following statements are motivated by [15]. Their proofs can easily be adapted to the LHA model. First, we state again the preservation of convexity by the time-elapse operation.

Lemma 1. *For a symbolic state (q, C), the time-elapse operation $C \uparrow_q$ can be effectively computed and it preserves the linear convexity of C.*

A proof based on operations on polyhedra can be found in [11].

Lemma 2. *For the initial state (q_0, C_0) of the symbolic semantics of a LHA $\mathcal{A}(K)$, there is an initial state (q_0, w_0) in the concrete semantics of $\mathcal{A}[\pi]$ for each parameter valuation π and each valuation w_0 with $\langle w_0, \pi\rangle \models C_0$.*

Proof. According to Definition 9, we show that each valuation w with $\langle w, \pi\rangle \models K \wedge [\bigwedge_{i=1}^{N} x_i = 0] \uparrow_{q_0}$ corresponds to an initial state (q, w) according to Definition 4. By definition of the time-elapse operation, we have $w \in [\bigwedge_{i=1}^{N} x = 0] \uparrow_{q_0} \Rightarrow \exists t \in \mathbb{R}_+, v \in D_{q_0} : w = t \cdot v$. As also $\langle w, \pi\rangle \models I_{q_0}$, this completes the proof. □

Lemma 3. *For each step $(q, C) \overset{a}{\Rightarrow} (q', C')$ in the symbolic semantics of $\mathcal{A}(K)$ and each valuation w' and π with $\langle w', \pi\rangle \models C'$, there is a step $(q, w) \overset{a}{\Rightarrow} (q', w')$ in the concrete semantics of $\mathcal{A}[\pi]$ for some valuation w with $\langle w, \pi\rangle \models C$.*

Proof. We decompose the proof for the transition $(q, C) \overset{a}{\Rightarrow} (q', C')$ into the steps $(q, C) \xrightarrow{a} (q', C'') \xrightarrow{t} (q', C')$.

For the delay transition $(q', C'') \xrightarrow{t} (q', C')$, we have $C' = C'' \uparrow_{q'}$. Since (q', C'') and (q', C') are states in the symbolic semantics, we have $C'' \subseteq I_{q'}$ and $C' \subseteq I_{q'}$. Following Lemma 1, $C'' \uparrow_{q'}$ is convex and also $C'' \subseteq C'' \uparrow_{q'}$. Thus, if $\langle w', \pi\rangle \models C'$, this implies $\langle w', \pi\rangle \models I_{q'}$ and there exist $w'' \in C'', v \in D_{q'}, t \in \mathbb{R}_+$ such that w' can be reached by letting time elapse to $w' = w'' + t \cdot v$. It follows that $\langle w'', \pi\rangle \models I_{q'}$. Thus, (q', w'') is a state in the concrete semantics with $(q', w'') \xrightarrow{t} (q', w')$ and $\langle w'', \pi\rangle \models C''$.

According to Definition 9, for step $(q, C) \overset{a}{\Rightarrow} (q', C')$ there exists a discrete transition in the automaton $\mathcal{A}$ with $q \xrightarrow{g,a,\mu} q'$. We have $(q, C) \xrightarrow{a} (q', C'')$ and thus $C''(X') = [C(X) \wedge g(X) \wedge \mu(X, X')] \downarrow_{X'} \wedge I_{q'}(X')$. As shown above, there exists a valuation w'' with $\langle w'', \pi\rangle \models C''$. Since $C'' \neq$ `false`, we can conclude that $\exists w : \langle w, \pi\rangle \models C \;\wedge\; \langle w, \pi\rangle \models g \;\wedge \langle w, w'', \pi\rangle \models \mu$. Since $C \subseteq I_q$, the state (q, w) is a state in the concrete semantics. This gives us the concrete discrete transition $(q, w) \xrightarrow{a} (q', w'')$ according to Definition 4. Together with the delay transition, we have $(q, w) \xrightarrow{a} (q', w'') \xrightarrow{t} (q', w')$ and thus $(q, w) \overset{a}{\Rightarrow} (q', w')$. □

This allows us now to state the correspondence of symbolic and concrete runs:

Proposition 1. *For each symbolic run of $\mathcal{A}(K)$ reaching (q, C), for each parameter valuation π and clock valuation w with $\langle w, \pi\rangle \models C$, there is an equivalent concrete run of $\mathcal{A}[\pi]$ reaching (q, w).*

Proof. By induction over the length of the run. The base case – a run with 0 transitions – directly follows from Lemma 2.

By induction hypothesis, we have a symbolic run ending in (q, C) and a corresponding concrete run ending in (q, w) with $w \models C$. For the induction step, we assume that we have an extended symbolic run ending with a transition $(q, C) \overset{a}{\Rightarrow} (q', C')$ and $\langle w', \pi\rangle \models C'$. Following Lemma 3, there exists a concrete step $(q, w) \overset{a}{\Rightarrow} (q', w')$. The concrete run can be therefore be extended by the step $(q, w) \overset{a}{\Rightarrow} (q', w')$, resulting in the required concrete run. □

Similarly, the converse direction can be shown.

Lemma 4. *Given an initial state (q_0, w) of the concrete semantics of $\mathcal{A}[\pi]$, there exists an initial state (q_0, C) of the symbolic semantics of $\mathcal{A}(K)$ for some parameter constraint K such that $\langle w, \pi\rangle \models C$.*

Proof. For the concrete initial state (q_0, w), we have $\langle w, \pi\rangle \models I_{q_0}$ and $\exists v, t : w = t \cdot v, t \in \mathbb{R}_+, v \in D_{q_0}$. According to Definition 8, this implies that $\langle w, \pi\rangle \models [\bigwedge_{i=1}^{N} x_i = 0] \uparrow_{q_0}$. Furthermore, choosing any K with $\pi \models K$, we also have $\langle w, \pi\rangle \models K$. Finally, as $\langle w, \pi\rangle \models I_{q_0}$, this proves that (q_0, C) with $C = K \wedge [\bigwedge_{i=1}^{N} x_i = 0] \uparrow_{q_0}$ is an initial state in the symbolic semantics of $\mathcal{A}(K)$.

Proposition 2. *For each parameter valuation $\pi \models K$ and valuation w, if there is a concrete run of $\mathcal{A}[\pi]$, reaching (q, w), then there is an equivalent symbolic run of $\mathcal{A}(K)$, reaching state (q, C) for some C such that $\langle w, \pi\rangle \models C$.*

Proof. By induction over the length of the run. The base case – again a run with 0 transitions – follows from Lemma 4.

By induction hypothesis, there exists a concrete run of $\mathcal{A}[\pi]$ ending in (q, w) and a corresponding symbolic run for some $\mathcal{A}(K)$ with $\pi \models K$ ending in a state (q, C) with $\langle w, \pi\rangle \models C$. For the induction step, we assume an extended concrete run of $\mathcal{A}[\pi]$, ending with the step $(q, w) \overset{a}{\Rightarrow} (q', w')$.

The considered concrete step $(q, w) \overset{a}{\Rightarrow} (q', w')$ consists of the two steps $(q, w) \xrightarrow{a} (q', w'') \xrightarrow{t} (q', w')$. By Definition 4, there exists a transition $q \xrightarrow{g,a,\mu} q'$ in the automaton $\mathcal{A}$ such that $\langle w, \pi\rangle \models g$ and $\langle w, w'', \pi\rangle \models \mu$. Since also $\langle w, \pi\rangle \models I_q$ and $\langle w', \pi\rangle \models I_{q'}$, it follows that there exist C, C'' with $\langle w, \pi\rangle \models C$ and $\langle w'', \pi\rangle \models C''$ such that $C''(X') = [C(X) \wedge g(X) \wedge \mu(X, X')] \downarrow_{X'} \wedge I_{q'}$ and $C \subseteq I_q$. Thus, (q, C) and (q', C'') are states in the symbolic semantics and $(q, C) \xrightarrow{a} (q', C'')$.

For the concrete state (q', w'') with the delay transition $(q', w'') \xrightarrow{t} (q', w')$, following Definition 4 there exist $v \in D_{q'}, t \in \mathbb{R}_+$ such that $w' = w'' + t \cdot v$. And thus, since $\langle w'', \pi\rangle \models C''$ and $\langle w', \pi\rangle \models I_{q'}$, it follows that $\langle w', \pi\rangle \models C'' \uparrow_{q'}$. Following Definition 9, we have a symbolic delay transition $(q', C'') \xrightarrow{t} (q', C')$. Together with the discrete transition, we can finally extend the assumed path with the step $(q, C) \overset{a}{\Rightarrow} (q', C')$, ending in symbolic state (q', C') with $\langle w', \pi\rangle \models C'$. □

Note that the symbolic semantics of $\mathcal{A}(K)$ can contain more traces than any single concrete semantics of $\mathcal{A}[\pi]$ for some $\pi \models K$. In general, if there is a parameter valuation π and a parameter constraint K, such that for all reachable states (q, C) of $\mathcal{A}(K)$, it holds that $\pi \models C$, then $\mathcal{A}(K)$ and $\mathcal{A}[\pi]$ are trace equivalent.

Furthermore, 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.

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     $S \leftarrow S \cup Post_{\mathcal{A}(K)}(S)$
```

Lemma 5. *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*, which is described in the following 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 [3], and 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.

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 3. *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 $\mathcal{A}[\pi]$ are trace equivalent, i.e. the sets of traces are identical*

Proof. Based on Lemma 5 and Propositions 1 and 2, the proof in [3] can be extended for LHA in a straightforward manner. □

In other words, we obtain a (convex) constraint 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 $\pi \models K_0$.

Note that the intersection in line 8 is necessary – rather than just returning K – in order to guarantee the equivalence of the traces. Without this operation, it is possible that the symbolic semantics of $\mathcal{A}(K)$ contains additional traces caused by deadlocks that do not occur in $\mathcal{A}[\pi_0]$. For more details, refer to [3].

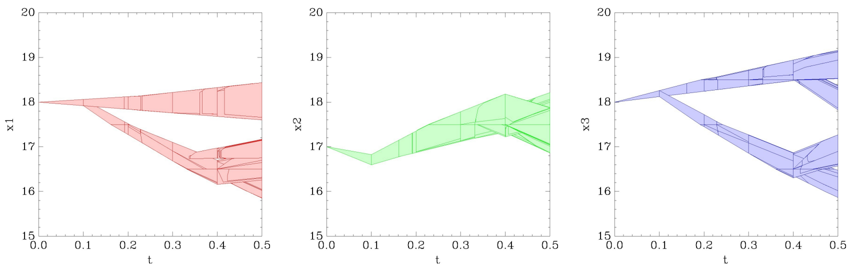
The algorithm *IM* is not guaranteed to terminate [1]. Note also that the presented algorithm involves nondeterminism. In Algorithm 1 in lines 4 and 5, one can possibly choose among several incompatible states and inequalities. This may lead to different – nevertheless correct – results. This implies in particular that the resulting constraint K_0 is not maximal in general. In order to further explore a hybrid system, the *behavioral cartography* can be applied, which is described in the next section. Before, we give a case study of the application of the inverse method.

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 *phase-portrait* approximation described in [14]. The space is partitioned into regions, and within each region, the activity field is overapproximated using linear sets of activity vectors. For each region R delimiting a portion of the partitioned state space, the activities are statically overapproximated as

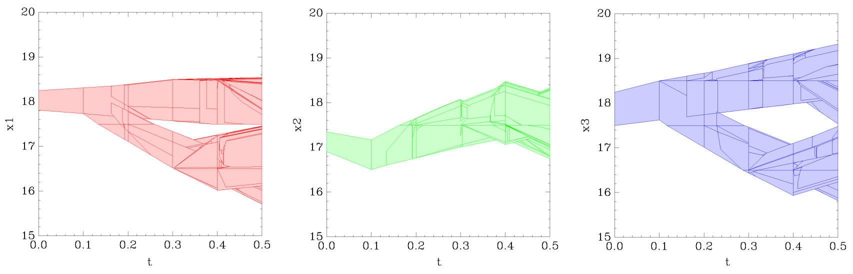
$$\dot{x}_i \in [min\{f_i(x) \mid x \in R\}, max\{f_i(x) \mid x \in R\}] \tag{3}$$

where $f_i(x)$ corresponds to the right-hand side in (2). 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 $x1, x2, x3$.

[1] Termination of such a general reachability-based procedure cannot be guaranteed due to undecidability of reachability for TA with parameters and LHA [13]



(a) Starting from a single point



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

Fig. 2. Reachable states for room heating benchmark

The inverse method considers only the discrete behavior of a system. It is thus well suited for qualitative properties that address the switching of discrete states. Such properties have been considered in [7], e.g.:

– All rooms eventually get a heater
– In all rooms there will eventually be no heater

In our model, we will use the inverse method to show a slightly weaker property, a bounded liveness property that is expressible as a safety condition as follows.

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 for varying initial conditions, we add the parameters a_1, a_2, a_3 and constrain the initial state with $x_1 = a_1 \wedge x_2 = a_2 \wedge x_3 = a3$.

The property can be checked by standard reachability analysis starting from a single point[2] (a_1, a_2, a_3). Starting from the 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

[2] In a first attempt, we checked this property using standard parametric reachability analysis. We considered the initial state wrt. the temperatures $(x1, x2, x3)$ to be within the rectangular region $[16, 18]^3$. But, no result could be obtained in this way due to the large state space.

Algorithm 2: *BC*

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

1 $Z \leftarrow \varnothing$
2 $V \leftarrow \{\pi \mid \pi_i = min_i + \ell_i \cdot \delta_i,\ \pi_i \leq max_i,\ \ell_1, \dots, \ell_M \in \mathbb{N}\}$
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 $Z$
```

(Algorithm 1), the initial point can be generalized to a larger region around the starting point $(18, 17, 18)$, resulting in the constraint

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

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* [4]. 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 $v_0 = [min_1, max_1] \times \dots \times [min_M, max_M]$. 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

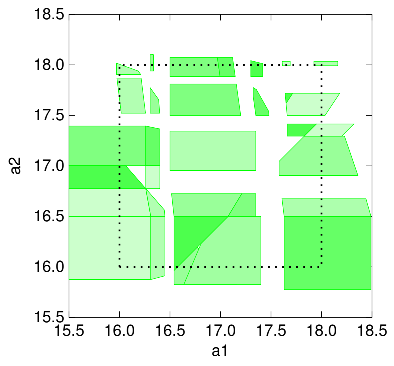


Fig. 3. Cartography of the initial states of RHB

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 the inverse method.

As such, the behavioral cartography only partitions the parameter space into tiles of distinct behavior, while it does not classify the tiles as good or bad ones. However, this classification is straightforward if a simple safety property is given as a set of bad states. Since the reachable locations under π_0 are computed during the application of the inverse method, we can easily classify each tile. Note that there may exist many tiles that differ only slightly, which may result in a large number of calls to the inverse method. However, this can be seen as a trade-off, since we may be able to classify large areas of the parameter space, while a standard parametric reachability analysis may fail.

Example 3. The cartography is illustrated by a further experiment on the RHB model from example 2. Again, we check the bounded liveness condition *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, we can even obtain a quantitative measure of the coverage of the considered region, which is 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.

Note 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. As reported in [1], their measure of coverage decreases when considering longer simulation traces (which correspond here to a bigger upper bound t_{max}). A similar effect can be observed for our method. The longer the traces, the more distinct behaviors are observed, resulting in smaller tiles and thus a smaller coverage of the parameter space. Further experimental results on LHA are reported in the appendix.

4.3 Limitations

It can be observed that for some systens there are border 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. This is e.g. the case if we repeat the above experiment fixing $a_3 = 17$ instead of $a_3 = 18$. 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 potentially 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 due to the fact that the traces contain a lot of information on the transitions between partitions that are more or less irrelevant wrt. the behavior of the modeled system.

4.4 Enhancement of the method for AHA

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 $\varDelta$ per dimension is chosen by the user. Note that cost and precision of the overall analysis may strongly depend on the chosen value for $\varDelta$. It is later discussed how one can use the methods presented in this report to perform an iterative verification, thereby refining the analysis by increasing $\varDelta$.

Given this variant of the time-elapse for affine dynamics, the computed reachable states are an over-approximation due to the piece-wise linearization of the dynamics and the convex hull operation. Thus, the equality of concrete and symbolic runs as given by Propositions 1 and 2 are no longer valid. But, as we compute an over-approximation of the possible runs, non-reachability (i.e. safety) properties are preserved.

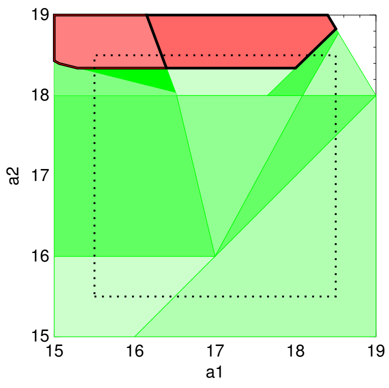


Fig. 4. Enhanced cartography for room heating benchmark

Proposition 4. *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.

As in the previous experiment, 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 Section 4.3, applying the cartography on the statically linearized RHB model does not deliver a good coverage when fixing $a_3 = 17$. 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 is shown in Fig. 4, consisting 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. Furthermore, already with this very coarse linearization, most of the tiles could be proved good. Thus, in a next step, one could concentrate a more costly analysis on the bad region.

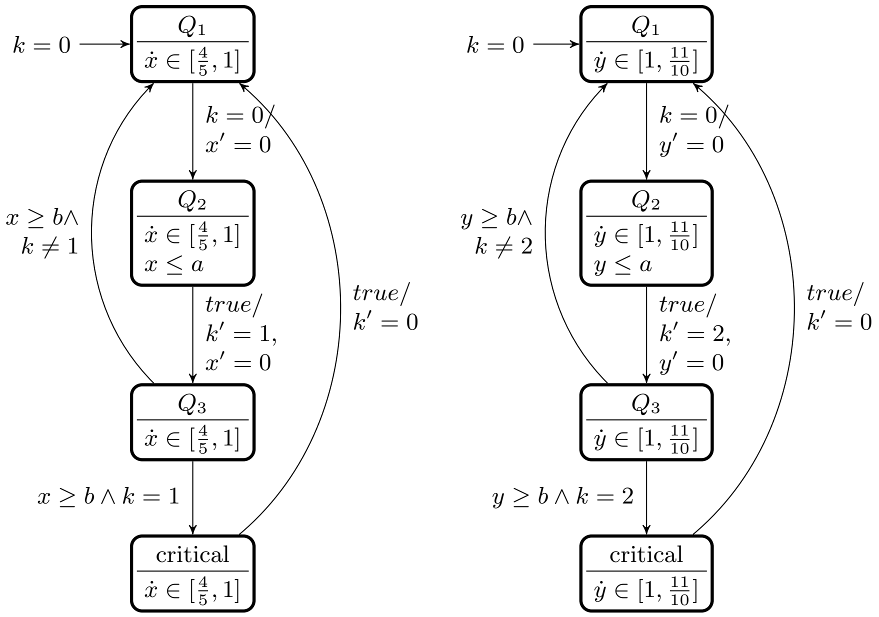


Fig. 5. Fischer mutual exclusion protocol

5 Further results

5.1 Fischer Protocol

In this section, we will examine the *Fischer mutual exclusion protocol*. The protocol provides mutual exclusion for a distributed system with skewed clocks. The processes communicate via a shared variable k. Before entering a critical section, each process reads the variable k. If k, then the process sets k to its own unique ID, which takes at most a time units. If, after waiting for b time units, k is still set to its own ID, the process is allowed to enter the critical section. Otherwise, the access cycle starts again. After leaving the critical section, k is reset to zero. The correct functionality of the protocol (there can be at most one process in a critical section) depends on the choice of the timing parameter b wrt. to the clock skew of the processes and the maximum write delay a.

Fig. 5 shows an instantiation of the Fischer protocol for two processes, where P_1 has a relative clock speed between $\frac{4}{5}$ and 1, and P_2 has a relative clock speed between 1 and $\frac{11}{10}$. In general, a high value for b will guarantee that all concurrent writes will be finished before a process checks variable k and eventually enters the critical section. For the given instantiation, it can be verified that for a maximum write delay set to the unit value $a = 1$, the choice of $b = 2$ is sufficient to guarantee a correct behavior.

For performance reasons, a value as small as possible should be chosen. Starting the Inverse Method with the above instantiation $\pi_0 = (1, 2)$, the following constraint is derived:

$$a \geq 0 \quad \wedge \quad b > \frac{11}{8}a \tag{4}$$

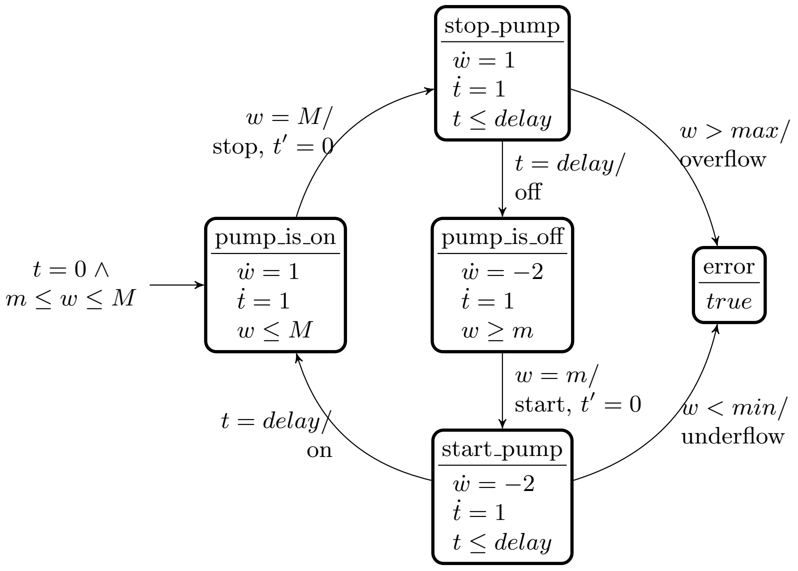


Fig. 6. LHA of a controlled water reservoir

This means that for any parameter instantiation satisfying (4), the system will show the same discrete behavior. Since the correctness of the protocol has been shown for π_0, we can conclude that (4) is sufficient for the mutual exclusion. Furthermore, the generated constraint is maximal. The same result is obtained by performing a complete parametric reachability analysis – e.g. with HyTech [12] – and projecting the reachable states on the parameters.

5.2 Water Tank Benchmark

The Model As another benchmark with a total number of five parameters, consider a monitored water reservoir [11], modeled by the LHA in Fig. 6. The state variables are the water level w and a global clock t. There is a pump attached to the reservoir, providing it with fresh water. When the pump is on, the water level increases with a constant rate of $\dot{w} = 1$. When it is off, the reservoir is drained at a rate of $\dot{w} = -2$. As parameters, max and min give the bounds on w that should be respected by the system. M and m give limits on the water level for which the pump is (de-) activated. But, there is a *delay* needed to actually change the activity of the pump. While waiting for the pump to react, an $overflow$ ($w > max$) or $underflow$ ($w < min$) can happen, leading the system to an error state.

Inverse Method The inverse method can be used to derive appropriate constraints on the parameters min, max, M, m and *delay* such that no overflow or underflow can happen. A reference valuation π_0 can easily be found for a short *delay* and by allowing for sufficiently large margins between min and m (M and max), respectively. As starting point, $\pi_0 = (min \mapsto 0, m \mapsto 10, M \mapsto 20, max \mapsto$

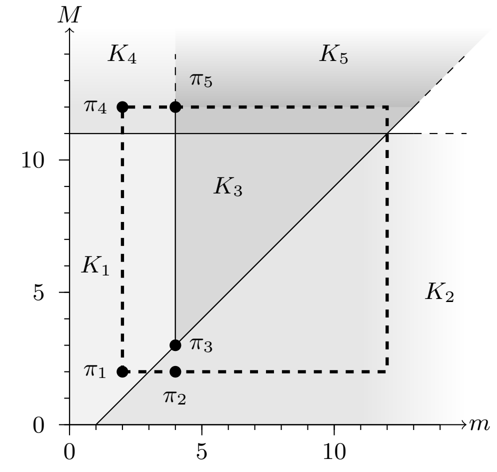


Fig. 7. Cartography for water reservoir

$30, delay \mapsto 1)$ is used. As a result, the following constraint is generated using forward reachability[3]:

$$\begin{aligned} M + delay &\geq m \wedge \\ m &\geq min + 2 \cdot delay \wedge \\ max &\geq M + delay \end{aligned} \tag{5}$$

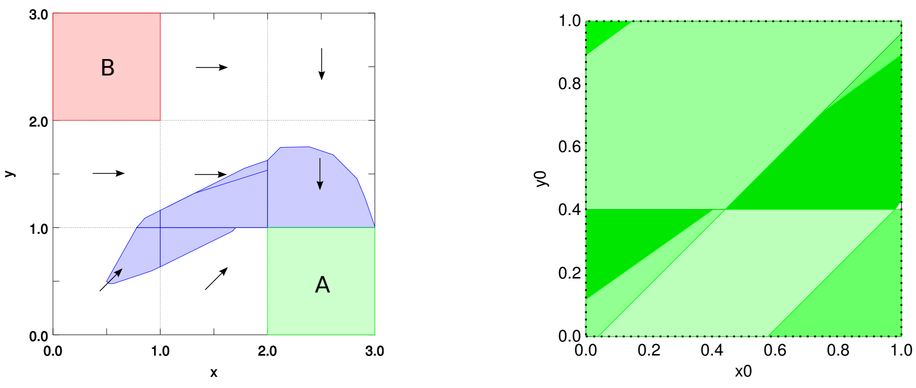
Behavioral Cartography In order to fully explore the possible behaviors of the system, the cartography algorithm is applied, with varying parameters m and M. The remaining parameters are fixed to the values $min = 2, max = 12$ and $delay = 1$. The zone to explore is given by the interval $m, M \in [min, max]$.

The cartography results in five different tiles, shown in Fig. 7. In the figure also the starting points π_1 to π_5 can be found, that were used to call the Inverse Method. As it turns out, the whole (real-valued) rectangle $[2, 12]^2$ is covered by the generated constraints. Furthermore, the trace set associated with constraint K_3 is the only one that corresponds to a good behavior of the system. Thus, the constraint in (5) is maximal.

5.3 Navigation Benchmark

The Model The *navigation benchmark* has been described in [7]. It describes an object moving in a plane. The plane is divided into square fields, where each field is associated with one of eight directions, to which the movement of the object converges. Furthermore, there are bad fields (marked B) that need to be avoided and good fields (marked A) that should eventually be reached by the object.

[3] Additionally to the shown inequalities, only non-negative parameters are considered.



(a) Reachable states from point (b) Behavioral cartography

Fig. 8. Navigation benchmark

More formally, the object has the current position $(x, y)^T$. The horizontal and vertical velocity are described by the vector $v = (v_x, v_y)^T$. For each field in the matrix F, the desired velocity is given as $v_d = (sin(i \cdot \pi/4), cos(i \cdot \pi/4))^T$, with a given $i \in \{0, \ldots, 7\}$. The convergence of the current velocities to the desired ones is determined by the differential equation $\dot{v} = A(v - v_d)$, where the matrix $A \in \mathbb{R}^{2\times2}$ is given by the benchmark instance. For example, Fig. 8(a) shows a 3×3 benchmark instance given by

$$F = \begin{pmatrix} B & 2 & 4 \\ 2 & 2 & 4 \\ 1 & 1 & A \end{pmatrix}, \; A = \begin{pmatrix} -1.2 & 0.1 \\ 0.1 & -1.2 \end{pmatrix}, \qquad (6)$$

with the initial states defined as $x \in [0, 1], y \in [0, 1], v_x \in [0.1, 0.5]$ and $v_y \in [0.05, 0.25]$.

The system can be modeled by an AHA in a straightforward way, where each field corresponds to a control location. In order to compute the reachable states, the system needs to be linearized wrt. variables v_x, v_y. The initial states are represented by parameters x_0, y_0, v_{x0}, v_{y0}.

Behavioral Cartography While a full parametric reachability analysis is costly, the system can be analyzed point-wise. This means that the parameters defining the initial state are fixed to a single value, while the behavioral cartography is used to obtain a measure of coverage of the verified behavior. As an example, the blue regions in Fig. 8(a) represent the reachable states from the initial point $(0.5, 0.5)^T$.

In an experiment, we explore the parameter space for the two parameters x_0, y_0 within the interval $[0, 1]$ with a step size $\delta = 0.1$. In this way, we obtain eight different tiles, that almost completely cover the considered rectangle, see Fig. 8(b). Only a small triangular region on the right hand side of the figure remains uncovered. All the covered tiles are classified as good tiles, since the bad state is not reached.

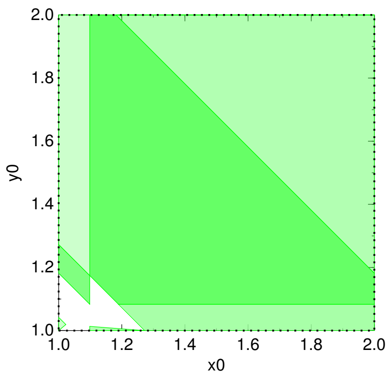


Fig. 9. Coverage of initial states

Coverage Another instance of the navigation benchmark is considered in [16], given by the map

$$F = \begin{pmatrix} B & 2 & 4 \\ 4 & 3 & 4 \\ 2 & 2 & A \end{pmatrix} \tag{7}$$

and the matrix A as in (6). There, the coverage of the initial states in the rectangle $[1,2] \times [1,2]$ with $v_{x0} = -0.2$ and $v_{y0} = 0$ is computed, starting with 25 equally distributed test points. The computed coverage is reported with 48%. In contrast, using the behavioral cartography, only a coverage of more that 97% is achieved with only 9 different tiles[4], as depicted in Fig. 9.

For the same map, the initial states $(x,y)^T \in [2.2, 2.8] \times [1.2, 1.8]$ were considered. Starting with 9 test points, an estimated coverage of 72% was computed. In constrast, choosing any single point in the given initial region, a single constraint is generated by the inverse method, which covers 100% of the rectangle.

6 Discussion

As shown in the previous sections, the inverse method – having been introduced for the analysis of timed automata – can be applied as well for hybrid systems. The extension to automata with rectangular and linear dynamics is straightforward, using the relation between concrete and symbolic semantics which extends nicely to these classes of hybrid automata. However, almost all non-trivial examples of hybrid systems from the literature have affine dynamics. The naive approach – approximating affine models statically by LHA – shows limited results, as the partitioning of locations leads to a great number of distinct trace sets.

Instead, the partitioning can be applied locally, incorporating it into the time-elapse operator and thereby grouping states that belong to the same location of the original affine model. In this way, more general constraints and thus a better coverage can be achieved. The additional convex hull operation can however

[4] Two of the tiles degenerate to a point or a line and can thus not be seen in the figure

be quite costly and strongly depends on the chosen number of partitions per dimension. This can be seen as a trade-off between precision and performance of the analysis. In practice, the method can be applied in an iterative manner, starting with a very coarse linearization, and then concentrating on small parts of the parameter space with a finer approximation.

The presented methods rely only on trace sets, abstracting from the valuations of the continuous variables. For this reason they are best suited for the verification of qualitative properties, like the (non-)reachability of a set of locations. Since many interesting properties – like safety conditions – can be expressed as reachability, this is not a serious limitation. There are also quantitative properties that can be coded as reachability, e.g. by adding a transition to a bad state when a certain deadline or threshold value is violated. Also for this class of properties, our methods can be applied.

7 Conclusions

In this report, we present a method to derive parameter constraints for LHA, that guarantee the same behavior as for a reference valuation of the parameters. This inverse 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.

The method can be used to attack the parameter synthesis problem for LHA, by generalizing a reference valuation that is known to guarantee a good behavior. 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, thus providing an alternative tool to the symbolic simulation method of [1].

The extended algorithms have been implemented in the tool IMITATOR. The tool has been demonstrated on a hybrid system benchmark, a distributed temperature control system. A good coverage of the parameter space can be achieved with our method, while standard parameterized reachability analysis fails.

References

1. R. Alur, A. Kanade, S. Ramesh, and K. Shashidhar. Symbolic analysis for improving simulation coverage of simulink/stateflow models. In *Proc. of the ACM int'l conf. on Embedded software (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. E. André, T. Chatain, E. Encrenaz, and L. Fribourg. An inverse method for parametric timed automata. *International Journal of Foundations of Computer Science*, 20(5):819–836, Oct. 2009.

4. E. André and L. Fribourg. Behavioral cartography of timed automata. In *Proc. of the Workshop on Reachability Problems in Computational Models (RP)*, volume 6227 of *LNCS*, pages 76–90. Springer, 2010.
5. 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.
6. P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *POPL*, pages 84–96, 1978.
7. A. Fehnker and F. Ivancic. Benchmarks for hybrid systems verification. In *Hybrid Systems: Computation and Control (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 *Workshop on Hybrid Systems: Computation and Control (HSCC)*, volume 4981 of *LNCS*, pages 187–200. Springer, 2008.
10. G. Frehse, B. Krogh, and R. Rutenbar. Verifying analog oscillator circuits using forward/backward abstraction refinement. In *Design, Automation and Test in Europe (DATE)*, pages 257–262, 2006.
11. N. Halbwachs, Y.-E. Proy, and P. Roumanoff. Verification of real-time systems using linear relation analysis. In *Formal Methods In System Design*, pages 157–185, 1997.
12. T. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech:A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer (STTT)*, 1:110–122, 1997.
13. T. Henzinger, P. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Journal of Computer and System Sciences*, pages 373–382. ACM Press, 1995.
14. T. Henzinger and H. Wong-Toi. Linear phase-portrait approximations for nonlinear hybrid systems. In *Hybrid Systems III*, volume 1066 of *LNCS*, pages 377–388. Springer, 1996.
15. T. Hune, J. Romijn, M. Stoelinga, and F. Vaandrager. Linear parametric model checking of timed automata. In *TACAS*, pages 189–203, 2001.
16. A. Julius, G. Fainekos, M. Anand, I. Lee, and G. Pappas. Robust test generation and coverage for hybrid systems. In *Hybrid Systems: Computation and Control*, volume 4416 of *LNCS*, pages 329–342. Springer, 2007.
17. B. Silva and B. Krogh. Modeling and verification of sampled-data hybrid systems. In *Proc. of the Int'l Conf. on Automation of Mixed Processes - Hybrid Dynamic Systems (ADPM)*, 2000.
18. T. Stauner, O. Müller, and M. Fuchs. Using HyTech to verify an automotive control system. In *In Proc. Hybrid and Real-Time Systems (HART)*, volume 1201 of *LNCS*, pages 139–153. Springer, 1997.

A Notational Conventions

a	action
g	guard
i, j	index
m, n	natural number
p	parameter
q	location
s	state
t	real number
u, v, w	valuation (point)
x, y	continuous variable
C	constraint
D	flow constraint (convex set of time derivatives)
I	invariant
K	parameter constraint
M	total number of parameters
N	total number of clocks
P	set of parameters
Q	set of locations
R	run
S	set of states
T	trace
X	set of continuous variables
Z	set of parameter constraints
α, β	integers
δ	step size for behavioral cartography
μ	jump relation
π	parameter valuation (point)
Σ	alphabet
$\mathcal{A}$	parametric linear hybrid automaton
$\mathcal{L}$	set of convex linear constraints
$\mathcal{V}$	set of valuations
$\mathbb{N}$	the natural numbers
$\mathbb{Z}$	the integers
$\mathbb{R}$	the real numbers