Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 223 (2008) 47–60
www.elsevier.com/locate/entcs

Approximated Reachability on Hybrid Automata: Falsification meets Certification

K. Bauer [a,1] R. Gentilini [a,2] K. Schneider [a,3]

[a] *Department of Computer Science, Embedded Systems Group, University of Kaiserslautern (Germany)*

Abstract

Undecidability of the reachability problem is ubiquitous in the context of hybrid automata. Being mostly based on either bounded reachability or on the notion of simulation preorder, current techniques for the approximated reachability analysis force to choose between under- and over-approximations.
In this paper, we introduce a novel method for the reachability analysis of hybrid automata featuring (1) the ability of combining the certification and the falsification of reachability properties, and (2) the applicability to highly expressive families of hybrid automata, whose dynamics are not amenable to an exact representation.

Keywords: Hybrid Automata, Reachability Analysis, Abstraction-Refinement.

1 Introduction

Hybrid automata [10,2] provide an appropriate modeling paradigm for systems where continuous variables interact with discrete modes. Such models are frequently used in complex engineering fields like embedded systems, robotics, avionics, and aeronautics [1,3,20,9]. In hybrid automata, the interaction between discrete and continuous dynamics is naturally expressed by associating a set of differential equations to every location of a finite automaton.

Finite automata and differential equations are well established formalisms in mathematics and computer science. Despite of their long-standing tradition, their combination in form of hybrid automata leads to surprisingly difficult problems that are often undecidable. In particular, the *reachability* problem is undecidable for most families of hybrid automata [16,15,17,11,2,5], and the few decidability results are built upon strong restrictions of the dynamics [4,12]. The reachability

[1] Email: k_bauer@informatik.uni-kl.de
[2] Email: gentilin@informatik.uni-kl.de
[3] Email: schneider@informatik.uni-kl.de

doi:10.1016/j.entcs.2008.12.030

analysis of hybrid automata is a fundamental task, since checking *safety* properties of the underlying system can be reduced to a reachability problem for the set of bad configurations [10].

For this reason, a growing body of research is being developed on the issue of dealing with approximated reachability on undecidable – yet reasonably expressive – hybrid automata [6,21,8,19,20]. To this end, most of the techniques proposed so far either rely on bounded state-reachability or on the definition of finite abstractions. While the first approach suffers inherently of incompleteness, the quest for *soundness* is a key issue in the context of methods based on abstractions. In fact, abstractions can introduce unrealistic behaviors that may yield to spurious errors being reported in the safety analysis. Usually, a simulation preorder is required to relate the abstraction to the concrete dynamics of the hybrid system under consideration, ensuring at least the correctness of each response of (abstract) *non* reachability. In [7], a novel abstraction-refinement framework has been proposed, based on the encoding of both over- and under-approximations of the reachable configurations. Hence, both sound proofs *and* sound counter-examples can be reported for reachability (i.e. safety).

Unfortunately, the abstraction-refinement framework in [7] is based on the *exact* analysis of the continuous dynamics in the underlying hybrid automata. Namely, it requires (1) the ability of solving the set of differential equations in each location ℓ, obtaining the corresponding flow functions f_ℓ, and (2) the ability of recovering an exact symbolic representation of the image-set $f_\ell(r)$, for an arbitrary initial region r. These requirements strongly limit the applicability of the method in [7]. In particular, though (possibly) undecidable, the classes of hybrid automata amenable to [7] have relatively simple continuous dynamics, and thus they still do not allow the faithful modeling of real dynamical systems.

In this paper, we propose a new sequence of abstractions that still is adequate *both* to certify and to falsify safety properties on the underlying hybrid automata. However, the computation of our abstractions does not require to recover *exactly* the target set of the flow's evolution in each location, $f_\ell(r)$. Rather, we assume that the latter can be only over-approximated and under-approximated. Hence, the method applies to a family of hybrid automata significatively more expressive than the one covered by [7]. As an example, we can now consider the evolution of continuous variables according to flows involving trigonometric terms, that can not be dealt with by means of symbolic computation, but are easily amenable to approximation methods (e.g by means of interval arithmetic).

The paper is organized as follows. Following the preliminaries (in Section 2), Section 3 introduces our new sequence of abstractions, so called *ABB abstractions*. Section 4 proves that ABB abstractions can be used to both certify and falsify safety properties on hybrid automata: Related efficient algorithms (linear in the size of the ABB abstractions) are provided in Subsection 4.1. Section 5, finally sketches a sound three-valued semantics for more general branching properties (namely CTL formulas) on ABB abstractions.

2 Preliminaries

In this section, we introduce the basic definitions and the notations used in the remainder of the paper.

Definition 2.1 [Hybrid Automata [4]] A *Hybrid Automaton* is a tuple $H = (L, E, X, Init, Inv, F, G, R)$ with the following components:

- a finite set of *locations* L
- a finite set of *discrete transitions* (or *jumps*) $E \subseteq L \times L$
- a finite set of *continuous variables* $X = \{x_1, \dots x_n\}$ that take values in $\mathbb{R}$
- an initial set of conditions: $Init \subseteq L \times \mathbb{R}^n$
- Inv: $L \to 2^{\mathbb{R}^n}$, the *invariant location labeling*
- $F : L \times \mathbb{R}^n \to \mathbb{R}^n$, assigning to each location $\ell \in L$ a vector field $F(\ell, \cdot)$ that defines the evolution of continuous variables within ℓ
- $G : E \to 2^{\mathbb{R}^n}$, the *guard edge labeling*
- $R : E \times \mathbb{R}^n \to 2^{\mathbb{R}^n}$, the *reset edge labeling.*

We write $\mathbf{v}$ to represent a valuation $(v_1, \dots, v_n) \in \mathbb{R}^n$ of the variables' vector $\mathbf{x} = (x_1, \dots, x_n)$, whereas $\dot{\mathbf{x}}$ denotes the first derivatives of the variables in $\mathbf{x}$ (they all depend on the time, and are therefore rather functions than variables). A *state* in H is a pair $s = (\ell, \mathbf{v})$, where $\ell \in L$ is called the *discrete component* of s and $\mathbf{v}$ is called the *continuous component* of s. A *run* of $H = (L, E, X, Init, Inv, F, G, R)$, starts at any $(\ell, \mathbf{v}) \in Init$ and consists of continuous evolutions (within a location) and discrete transitions (between two locations). Formally, a run of H is a path with alternating continuous and discrete transitions in the *time abstract transition system* of H, defined below:

Definition 2.2 The *time abstract transition system* of the hybrid automaton $H = (L, E, X, Init, Inv, F, G, R)$ is the transition system $T_H = (Q, Q_0, \to)$, where:

- $Q \subseteq L \times \mathbb{R}^n$ and $(\ell, \mathbf{v}) \in Q$ if and only if $\mathbf{v} \in Inv(\ell)$
- $Q_0 \subseteq Q$ and $(\ell, \mathbf{v}) \in Q_0$ if and only if $\mathbf{v} \in Init(\ell) \cap Inv(\ell)$
- The transition relation $\to$ is defined as follows:
  - there is a *continuous transition* $(\ell, \mathbf{v}) \to (\ell, \mathbf{v}')$, if and only if there is a differentiable function $f : [0, t] \to \mathbb{R}^n$, with $\dot{f} : [0, t] \to \mathbb{R}^n$ such that:
    1. $f(0) = \mathbf{v}$ and $f(t) = \mathbf{v}'$
    2. for all $\varepsilon \in (0, t)$, $f(\varepsilon) \in Inv(\ell)$, and $\dot{f}(\varepsilon) = F(\ell, f(\varepsilon))$.
  - there is a *discrete transition* $(\ell, \mathbf{v}) \to (\ell', \mathbf{v}')$ if and only if there exists an edge $e = (\ell, \ell') \in E$, $\mathbf{v} \in G(\ell)$ and $\mathbf{v}' \in R((\ell, \ell'), \mathbf{v})$.

A region is a subset of the states Q of $T_H = (Q, Q_0, \to)$. Given a region α, the predecessor region $Pre(\alpha)$ is defined as the region $\{q \in Q \mid \exists q' \in \alpha . q \to q'\}$.

The relations of *simulation* and *bisimulation*, recalled below, are two fundamental tools in the context of hybrid automata abstraction.

Definition 2.3 [Simulation] Let $T^1 = (Q^1, Q_0^1, \rightarrow^1)$, $T^2 = (Q^2, Q_0^2, \rightarrow^2)$ be two transition systems, and consider a partition $\mathcal{P}$ over $Q^1 \cup Q^2$. A *simulation* from T^1 to T^2 is a nonempty relation $\leq_S \subseteq Q^1 \times Q^2$ such that, for all $p \leq_S q$ it holds:

- $p \in Q_0^1$ iff $q \in Q_0^2$, and $[p]_{\mathcal{P}} = [q]_{\mathcal{P}}$, where $[p]_{\mathcal{P}}$ denotes the class of p in $\mathcal{P}$.
- For each node p' such that $p \rightarrow^1 p'$, there exists a node q' such that $p' \leq_S q'$ and $q \rightarrow^2 q'$.

If there exists a simulation from T^1 to T^2, then we say that T^2 *simulates* T^1, denoted by $T^1 \leq_S T^2$. If $T^1 \leq_S T^2$ and $T^2 \leq_S T^1$, then the transition systems T^1 and T^2 are said *simulation-equivalent* (or *similar*), denoted by $T^1 \equiv_S T^2$.

Definition 2.4 [Bisimulation] Let $T^1 = (Q^1, Q_0^1, \rightarrow^1)$, $T^2 = (Q^2, Q_0^2, \rightarrow^2)$ be two transition systems and consider a partition $\mathcal{P}$ over $Q^1 \cup Q^2$. A *bisimulation* for T^1, T^2 is a nonempty relation $\equiv_B \subseteq Q^1 \times Q^2$ such that, for all $p \equiv_B q$ it holds:

- $p \in Q_0^1$ iff $q \in Q_0^2$, and $[p]_{\mathcal{P}} = [q]_{\mathcal{P}}$.
- For each node p' such that $p \rightarrow^1 p'$, there exists a node q' such that $p' \equiv_B q'$ and $q \rightarrow^2 q'$.
- For each node q' such that $q \rightarrow^2 q'$, there exists a node p' such that $p' \equiv_B q'$ and $p \rightarrow^1 p'$.

If there exists a bisimulation relation for the transition systems T^1, T^2, then T^1 and T^2 are said *bisimilation-equivalent* (or *bisimilar*), denoted by $T^1 \equiv_B T^2$.

From an algorithmic point of view, the characterization of the (maximum) bisimulation equivalence in terms of a coarsest partition problem [18,13] leads to the bisimulation algorithm for hybrid automata [10] depicted in Figure 1. Such an algorithm refines an initial partition over the states of an hybrid automaton H, until each class is *stable* w.r.t. the transition relation of T_H, i.e. until for each pair of classes (α, β), it holds $\beta \cap Pre(\alpha) \in \{\emptyset, \beta\}$.

3 Approximated Bounded Bisimulation Abstractions (ABB) of Hybrid Automata

In this section, we build up the basement of our safety analysis method for hybrid automata, featuring the ability of both certifying and falsifying safety.

Such a feature distinguishes our approach from traditional techniques based on the simulation preorder, that allow only to over-approximate the set of configurations admitting an evolution toward the target (unsafe) states. As in [20], we assume to deal with highly expressive hybrid automata, whose dynamics could be in principle not amenable to an exact symbolic computation. This means that, given a region α of the hybrid automaton H, the set of states

$$Pre(\alpha) = \{x \mid x \text{ can evolve to } \alpha \text{ by means of a continuous/discrete transition}\}$$

can not be exactly computed (and represented). Rather, we assume that the prede-

BISIM $(H, \mathcal{P})$

Input: Hybrid automaton H, Initial Partition $\mathcal{P}$
Output: Bisimulation w.r.t. $\mathcal{P}$, $\mathcal{B} = \langle Q, E \subseteq (Q \times Q)\rangle$

(1) $Q := \mathcal{P}, E := \emptyset;$

/*———-*Partition refinement via 2-way split*———-*/
(2) **while** $(\exists \alpha, \beta \in Q : Pre(\alpha) \cap \beta \notin \{\emptyset, \beta\}$ **do**
(3) $\beta_1 = Pre(\alpha) \cap \beta, \beta_2 = \beta \setminus \beta_1$
(4) $Q := (Q \setminus \{\beta\}) \cup \{\beta_1, \beta_2\}$

/*———*Definition of edges over the classes* ———*/
(5) **for each** $(\delta, \gamma \in Q)$ **do**
(6) **if** $(\delta \cap Pre(\gamma) = \delta)$ **then** $E = E \cup (\delta, \gamma)$
(7) **return** $B = \langle Q, E\rangle$

Fig. 1. Bisimulation on hybrid automata.

cessor region $Pre(\alpha)$ can be only over- and under-approximated. In the remainder of this work, we will use the notation $Pre^{\uparrow}(\alpha)$ (resp. $Pre^{\downarrow}(\alpha)$) to refer to an arbitrary over- (resp. under-) approximation of $Pre(\alpha)$, i.e. $Pre^{\downarrow}(\alpha) \subseteq Pre(\alpha) \subseteq Pre^{\uparrow}(\alpha)$.

APPROXBOUNDEDBISIM $(n, H, \mathcal{P})$

Input: $n \in \mathbb{N}$, H: Hybrid automaton, $\mathcal{P}$: Initial Partition
Output: n-ABB abstraction of H w.r.t. $\mathcal{P}$, $\mathcal{A}^n = \langle Q^n_{\mathcal{A}}, E^n_{\mathcal{A}} \subseteq (Q^n_{\mathcal{A}} \times Q^n_{\mathcal{A}} \times \mathbb{N} \cup \{\infty\})\rangle$

(1) $Q^0_{\mathcal{A}} := \mathcal{P}_0, E^0_{\mathcal{A}} := \emptyset;$
(2) **while** $(n > 0)$ **do**
(3) $n := n - 1; Q^n_{\mathcal{A}} := Q^{n-1}_{\mathcal{A}}, E^n_{\mathcal{A}} := \emptyset$

/*———————*Partition refinement via 3-way split*———————*/
(4) **for each** $\alpha \in Q^{n-1}_{\mathcal{A}}$
(5) **for each** $(\beta \in Q^n_{\mathcal{A}})$ **do**
(6) $\beta_1 := \beta \cap Pre^{\downarrow}(\alpha); \beta_2 := \beta \setminus Pre^{\uparrow}(\alpha) \cap (\beta \setminus \beta_1); \beta_3 := \beta \setminus (\beta_1 \cup \beta_2);$
(7) $Q^n_{\mathcal{A}} := Q^n_{\mathcal{A}} \setminus \beta;$
(8) **if** $\beta_{i=1,2,3} \neq \emptyset$ **then** $Q^n_{\mathcal{A}} := Q^n_{\mathcal{A}} \cup \beta_i;$

/*———*Definition of weighted-edges over the classes of the partition* ———-*/
(9) **for each** $(\gamma, \delta) \in Q^n_{\mathcal{A}}$ **do**
(10) **if** $Pre^{\downarrow}(\delta^{n-1}) \supseteq \gamma$ **then**
(11) **if** $(\gamma^{n-1}, \delta^{n-1}, m) \in E^{n-1}_{\mathcal{A}}$ **then** $\omega := min(n, m)$ **else** $\omega := n$
(12) $E^n_{\mathcal{A}} := E^n_{\mathcal{A}} \cup \{(\delta, \gamma, \omega)\}$
(13) **else if** $Pre^{\uparrow}(\delta^{n-1}) \supseteq \gamma$ **then** $E^n_{\mathcal{A}} := E^n_{\mathcal{A}} \cup \{(\delta, \gamma, \infty)\}$
(14) **return** $\mathcal{A}^n = \langle Q^n_{\mathcal{A}}, E^n_{\mathcal{A}}\rangle$

Fig. 2. Construction of ABB abstractions.

Note, that in our context, the construction of a bisimulation quotient is not amenable to the classic partition refinement approach, independently from its size. More precisely, the traditional partition-refinement bisimulation algorithm (cfr. Figure 1, in the previous section) is subject not only to problems of non-termination (that naturally occur for undecidable hybrid automata) but also of *non computability*! Such a procedure relies in fact on the availability of the operation $Pre(\alpha)$, for the predecessor region.

The development of our method departs exactly from attempting at recovering a sort of approximation of bisimulation for our families of hybrid automata. In more detail, we will build a *succession* of abstractions, that we call *approximated bounded bisimulation* (ABB) abstractions: Such abstractions retain enough information to provide *both* an over-approximation and an under-approximation of the set of states fulfilling a given reachability property.

The construction of the n-th element in our succession of abstractions (n-ABB abstraction) is given in Figure 2, and can be better understood if compared with the related classic bisimulation algorithm. There are three main differences between the procedure in Figure 1 (for bisimulation), and the construction of an n-ABB abstraction, given in Figure 2.

First, the fix-point is replaced by a bounded number of n iterations (as in classic *bounded* bisimulation).

Second, and more important, the fundamental step of *two-way split* for classic bisimulation, is replaced by a *three-way split* in an n-ABB abstraction.

As depicted in Figure 3, on the right, the classic two way-split induced by the class α on the class β, leads to the two new regions:

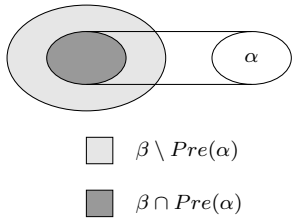

Fig. 3: Two way split

$$\beta_1 = Pre(\alpha) \cap \beta \quad \wedge \quad \beta_2 = \beta \setminus Pre(\alpha)$$

Here, β_1 must evolve to α, while β_2 does not reach α. In a three way split, the approximations $Pre^{\downarrow}(\alpha), Pre^{\uparrow}(\alpha)$ are employed in place of $Pre(\alpha)$. This leads to the three way split of β via α into:

$$\beta_1 = Pre^{\downarrow}(\alpha) \cap \beta \quad \wedge \quad \beta_2 = \beta \cap (Pre^{\uparrow}(\alpha) \setminus Pre^{\downarrow}(\alpha)) \quad \wedge \quad \beta_3 = \beta \setminus Pre^{\uparrow}(\alpha)$$

Now, β_1 *must* evolve to α, β_2 *may* evolve to α, while β_3 does not lead to α, as illustrated in Figure 4, below.

The last difference between a bounded bisimulation and an n-ABB abstraction involves the definition of the edges over the two abstractions. In fact, in ABB abstractions the control graphs are designed so that the information relative to a *must*/*may* evolution over regions is encoded into an edge, as soon as such an information is made explicit via a three-way split. More precisely, the edges of an n-ABB-abstraction are labeled by weights $w \in \{1 \dots n\} \cup \{\infty\}$. Infinite weights

among classes denote simply that the corresponding edge over-approximates the trajectory linking the states in such classes (i.e. has a *may* nature). To introduce the meaning of integer weights, denote by γ^i the (unique) region in the i-ABB abstraction $\mathcal{A}^i$ which contains γ, where γ is a class in the finer n-ABB abstraction $\mathcal{A}^{n\geq i}$. Then, an integer weight-label $m \neq \infty$ on the transition (δ, γ), encodes that the corresponding edge under-approximates the trajectories linking δ to γ^{n-1}. The different values of integer-weights allows to further recognize whether the under-approximation operator $Pre^{\downarrow}(\gamma)$ has been recurrently used to refine δ. The meaning of integer weights is more precisely outlined by Lemma 3.1, below, that concludes this section.

Lemma 3.1 (Interpretation of the Weights) *Let $\alpha \xrightarrow{m} \beta$ be a weighted transition in the n-ABB abstraction A^n, with $m \leq n$. For all $n \geq m' \geq m$ it holds: $\alpha^{m'} \rightarrow \beta^{m'-1}$ is a* must-transition*, i.e. $\forall x \in \alpha^{m'} \exists y \in \beta^{m'-1}$ such that H admits a trajectory departing from x and leading to y.*

Proof. We proceed by induction on $k = n-m$. For the base case, assume $n-m = 0$. Then, $\alpha \xrightarrow{n=m} \beta$, and we need to prove that $\alpha \rightarrow \beta^{n-1}$ is a must transition i.e. $\forall x \in \alpha$, $\exists y \in \beta^{n-1}$ such that H admits a trajectory from x to y. By Line 11 of Algorithm 2, $Pre^{\downarrow}(\beta^{n-1}) \supseteq \alpha$ is a necessary condition for the definition of the weighted edge $\alpha \xrightarrow{n} \beta$. Such a condition yields our claim. For the inductive step, assume $k = n - m > 0$ and let $\alpha \xrightarrow{m} \beta$. By Lines 11–12 in Algorithm 2 we have that (1) $Pre^{\downarrow}(\beta^{n-1}) \supseteq \alpha$, and (2) $\alpha^{n-1} \xrightarrow{m} \beta^{n-1}$ is a weighted edge in $\mathcal{A}^{n-1}$. Condition (1) ensures that $\alpha \rightarrow \beta^{n-1}$ is a must transition. Our inductive hypothesis applied to $\alpha^{n-1} \xrightarrow{m} \beta^{n-1}$ finally yields that, $\forall m \leq m' \leq n-1$, $\alpha^{m'} \rightarrow \beta^{m'-1}$ is also a must transition, i.e. our thesis. □

4 ABB Abstractions and Reachability: Safety Falsification meets Certification

In this section, we prove that the abstraction-refinement framework introduced in Section 5 can be used to both prove and falsify safety properties on the under-

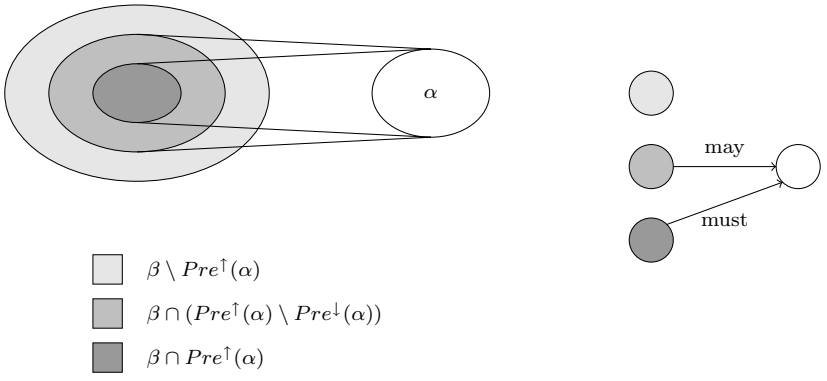

Fig. 4. The three-way split employed in the construction of ABB abstractions.

lying hybrid automata. To obtain such a result, we show that ABB abstractions provide enough information to both under-approximate, and over-approximate the possible global evolutions of the related dynamical system. In turn, safety can be established on the ground of overapproximations (of reachable 'bad' states), while underapproximations are sufficient to deal with safety falsification.

We start by establishing, in Theorem 4.1, a sufficient condition to determine whether a path in a n-ABB abstraction provides a faithful representation of a run-prefix. Thus, such a path can be used in the context of underapproximated reachability analysis.

Theorem 4.1 (Underapproximated Reachability) *Consider a weighted path $p : \alpha = \gamma_0 \overset{m_0}{\to} \dots \overset{m_{k-1}}{\to} \gamma_k = \beta$ in the n-ABB-abstraction $\mathcal{A}^n$ for the hybrid automaton H. Assume that, for all $0 \leq i \leq k$, it holds $m_i \leq n - i$. Then, for each $0 \leq i \leq k$, H admits an evolution from $\alpha = \gamma_0$ to γ_i^{n-i}.*

Proof. According to Lemma 3.1 and by our condition $m_i \leq n - i$, each transition $\gamma_i^{n-i} \to \gamma_{i+1}^{n-i-1}$ is a must-transition, i.e. each $x \in \gamma_i^{n-i}$ admits an evolution to a point $x' \in \gamma_{i+1}^{n-i-1}$ in H. Thus, starting in $\alpha = \gamma_0^n$ and successively following these must-transitions, it is possible to construct for each $x \in \alpha = \gamma_0$ a feasible trajectory of H, $x = x_0 \rightsquigarrow x_1 \rightsquigarrow \dots \rightsquigarrow x_i$, with $x_i \in \gamma_i^{n-i}$. □

Theorem 4.2 establishes that ABB abstractions are amenable also to overapproximated reachability analysis. This result is proved upon the existence of a simulation preorder relating each ABB abstraction to the time abstract transition system of the corresponding hybrid automaton.

Theorem 4.2 (Overapproximated Reachability) *Consider an ABB abstraction $\mathcal{A}^n$ for the hybrid automaton H, let s, q be two states of H, and denote by $\alpha = [s], \beta = [q]$ the corresponding classes in $\mathcal{A}^n$. If H admits a trajectory from s to q, then $\mathcal{A}_n$ admits a path from α to β.*

Proof. Assume that H admits a trajectory from s to q, and consider the classes of $\mathcal{A}^n$ traversed along such a trajectory, $\alpha_1, \dots, \alpha_k$. Then, for all $i = 1 \dots k$, $\alpha_i \subseteq Pre^{\uparrow}(\alpha_{i+1}^{n-1})$. By definition of $E^n_{\mathcal{A}}$ we conclude that $(\alpha_1, \alpha_2, \dots, \alpha_k)$ is a path in $\mathcal{A}^n$. □

Corollary 4.3 combines the above results on over- and under-approximated reachability into a three-valued approach to the safety analysis of hybrid automata.

Corollary 4.3 (Safety) *Let $\mathcal{A}^n = \langle Q^n, E^n \rangle$ be an n-ABB abstraction of the hybrid automaton H computed w.r.t. the initial partition $\mathcal{P}$. Assume that $\mathcal{P}$ is compatible*[4] *w.r.t. the initial states of H, I_H, and w.r.t. a set of bad configurations for H, B_H. Then, denoted by I (resp. B) the set of classes in $\mathcal{A}_n$ refining I_H (resp. B_H):*

(i) *H is* safe *w.r.t. B_H, if $\forall \alpha \in I, \forall\ \beta \in B : \beta$ is not reachable from α in $\mathcal{A}^n$.*

[4] A partition $\mathcal{P}$ on Q is compatible w.r.t. $r \subseteq Q$ iff for each class α in $\mathcal{P}$, $\alpha \cap r \in \{r, \emptyset\}$.

(ii) *H is* not safe *w.r.t. B_H, if $\mathcal{A}^n$ admits a path $p : I \ni \alpha = \gamma_0 \overset{m_0}{\rightarrow} \ldots \overset{m_{k-1}}{\rightarrow} \gamma_k = \beta \in B$ satisfying $m_i \leq n - i$, for all $0 \leq i \leq k$.*

We are now ready to provide an applicative character to Corollary 4.3, i.e. to define an efficient safety analysis algorithm on ABB abstractions. Such an algorithm is discussed in the following subsection.

4.1 Efficient Algorithmic Safety Analysis on ABB Abstractions

The algorithmic 'positive' safety analysis on simulation-based abstractions of hybrid automata, relies in general on the visit of the corresponding finite graph structure, by means of traditional *linear* graph exploration algorithms. Here, 'positive' refers to the fact that the only *certification* of safety can be stated, since the solely *non*-reachability results (for a region of bad states) can be trusted, due to overapproximation. The procedure depicted in Figure 5 shows that the algorithmic exploration of ABB abstractions leads instead to a *three*-valued safety analysis, using again only a linear number of steps. In our three-valued approach, boolean values refer to either a *positive* or a *negative* response to the safety inquiry, while the third value $\perp$ models the fact that the given abstraction is too coarse to decide the problem.

The algorithm in Figure 5 is indeed a variant of the classic breadth first search exploration, where the weights on the edges of the abstraction are also taken into account, according to Corollary 4.3. More precisely, the procedure takes as input an n-ABB-abstraction $\mathcal{A}^n = \langle Q^n, E^n \rangle$, a set of initial regions $I \subseteq Q^n$ and a set of (bad) regions [5], $B \subseteq Q^n$.

The first loop at lines (4)–(10) uses a queue Q to collect the states of $\mathcal{A}^n$ relevant to falsify the safety of the underlying hybrid automaton, according to Theorem 4.1 and Corollary 4.3. Namely, this loop collects the states of $\mathcal{A}^n$ reachable via a path p in which each edge e_i is labeled by a weight m_i satisfying the relation $m_i \leq n - i$. This is done by:

(i) First, maintaining for each node inserted into Q, say α, the length of the 'feasible' path leading to its discovery. The latter is stored in the field $\ell(\alpha)$.

(ii) Second, selecting the edges having a weight $0 \leq m \leq n - \ell(\alpha)$, in order to discover new nodes from α, using again 'feasible' paths (cfr. line (9)).

Here, 'feasible' naturally refers to our constraint on the disposition of weights along the paths, given in Theorem 4.1. If any of the abstract states so collected belongs to the set B given as input, then the algorithm terminates returning the value **'not safe'**, meaning that the underlying hybrid automaton is unsafe.

The second loop completes the visit to recover the states of $\mathcal{A}^n$ relevant for certifying the safety of the abstracted dynamical system, according to Theorem 4.2 and Corollary 4.3. The latter are given by the rest of reachable nodes in $\mathcal{A}^n$ (via an arbitrary path). If also none of these regions belongs to B, the procedure returns

[5] We naturally assume that $\mathcal{A}^n = \langle Q^n, E^n \rangle$ has been computed w.r.t. an initial partition for the hybrid automaton H compatible w.r.t. to the initial and the bad configurations of H, I_H, B_H. Hence, $I \subseteq Q^n$ and $B \subseteq Q^n$ refine I_H and B_H, respectively.

SafetyAnalysis $(\mathcal{A}^n, I, B)$

Input: ABB abstraction A^n of H, Initial regions $I \subseteq Q_A^n$
*Output: $a \in \{$**'safe', 'not safe', 'perhaps not safe'**$\}$*

```
(1)  Q, Q' := ∅; (FIFO − queues)
(2)  for each  α ∈ Q^n do :  reach(α) := 0; ℓ(α) := +∞;
(3)  for each  α ∈ I do :  ℓ(α) := 0; Q.append((r, n));

/*————————Collection of states relevant for safety falsification————————*/
(4)  while Q.notEmpty do
(5)    α := Q.getfront;
(6)    if reach(α) = 0 then
(7)      reach(α) := 1;
(8)      for each  e = (α, β, m) do
(9)        if m ≤ n − ℓ(α) then Q.append(β);  l(β) := l(α) + 1;
(10)         else Q'.append(β);

/*————————Collection of states relevant for safety certification————————*/
(11) while Q'.notEmpty do
(12)   α := Q'.getfront;
(13)   if reach(α) = 0 then
(14)     reach(α) :=⊥;
(15)     for each  e = (α, β, m) do : Q'.append(β);

/*————————Three valued safety analysis————————————————————————————*/
(16) if ∃α ∈ B : reach(α) = 1 then return 'not safe';
(17) if ∃α ∈ B : reach(α) =⊥ then return 'perhaps not safe';
(18) return 'safe';
```

Fig. 5. Linear Three-valued safety analysis algorithm on ABB-abstractions.

'safe'. Theorem 4.4 below states that each boolean answer **'safe'** (resp. **'not safe'**) returned by Algorithm 5 is sound.

Theorem 4.4 (Soundness) *If the algorithm* SafetyAnalysis *$(\mathcal{A}^n, I, B)$ returns **'not safe'**, then the hybrid automaton H abstracted by $\mathcal{A}^n$ admits a run to a bad configuration $x \in \alpha \in B$. Conversely, if* SafetyAnalysis *$(\mathcal{A}^n, I, B)$ returns **'safe'**, then H does not admit any run to a bad configuration $x \in \alpha \in B$.*

Proof. By Lines (16)–(18) in Algorithm 5, the procedure returns **'not safe'** iff upon the termination of the two loops there exists a state $\alpha \in B$, whose corresponding field $reach(\alpha)$ is 1. By Lines (6)–(7) $reach(\alpha) = 1$ iff α has been inserted into the queue Q within the first loop. Thus, by Corollary 4.3, to establish our claim for each negative boolean answer **'not safe'** it is sufficient to show that a state β is inserted into the queue Q iff it $\mathcal{A}^n$ contains a path $p : \alpha = \gamma_0 \overset{m_0}{\rightarrow} \dots \overset{m_{k-1}}{\rightarrow} \gamma_k = \beta \in B$ satisfying $m_i \leq n - i$, for all $0 \leq i \leq k$. Such a statement can be easily proved by induction on the number of insertions into Q.

Similarly, the soundness of each positive boolean answer **' safe'**, can be estab-

lished on the ground of Corollary 4.3, and by induction on the number of insertions into either the queue Q or the queue Q'. □

If the algorithm terminates with output **'perhaps not safe'**, then the abstraction is too coarse to make a decision. Theorem 4.5, below, finally establishes the claimed linear complexity of our algorithm, and concludes this section.

Theorem 4.5 (Complexity) *The algorithm* SafetyAnalysis *$(\mathcal{A}^n, I, B)$ terminates in time $\mathcal{O}(|Q^n + E^n|)$, where $\mathcal{A}^n = \langle Q^n, E^n \rangle$.*

Proof. Overall the execution of the algorithm, the adjacency list of each node α can be inspected at most once (either at lines (8)–(10), or at line (15)). In fact, such inspections are guarded by the condition $reach(\alpha) = 0$, and the field $reach(.)$ is updated either to 1 or to $\perp$ as soon as it is discovered to be 0. It follows that the global cost of executing the innermost for-loops is $\mathcal{O}(\sum_{\alpha \in Q^n} E^n(\alpha)) = \mathcal{O}(E^n)$. Such a cost determine the entire complexity of the external while-loops at lines (4)–(15). The cost of the rest of the code is clearly $\mathcal{O}(Q^n)$. □

5 A 3-Valued Semantics for CTL on ABB Abstractions

After having introduced in Section 4.1 a framework for the three-valued safety analysis of hybrid automata, in this section we consider stronger properties expressed by means of the temporal logic CTL. Definition 5.1 and Definition 5.2 recapitulate syntax and semantics of CTL formulas on hybrid automata. Note that due to the density of the underlying time framework the neXt operator is omitted [2,10].

Definition 5.1 [CTL for Hybrid Automta] Let AP be a finite set of propositional letters and $p \in \mathsf{AP}$. CTL is the set of formulas defined by the following syntax:

$$\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid E(\phi_1 \underline{\mathrm{U}} \phi_2) \mid \mathsf{A}(\phi_1 \underline{\mathrm{U}} \phi_2)$$

Definition 5.2 [CTL Semantics] Let $H = (L, E, X, Init, Inv, F, G, R)$ be a hybrid automaton, and let AP be a set of propositional letters. Consider $\ell_{\mathsf{AP}} : L \times X \to 2^{\mathsf{AP}}$. Given $\phi \in CTL$ and $q \in Q$, $[\![\phi]\!](s)$ is inductively defined:

- $[\![p]\!](q) = 1$ if and only if $p \in \ell_{\mathsf{AP}}(s)$
- $[\![\neg\phi]\!] = \neg[\![\phi]\!]$
- $[\![\phi \diamond \psi]\!] = [\![\phi]\!] \diamond [\![\psi]\!]$ for $\diamond \in \{\wedge, \vee\}$
- $[\![E(\phi \underline{\mathrm{U}} \psi)]\!](q) = 1$ iff there exists a run ρ departing from q that admits a prefix $\rho^* := q_1 \to \ldots \to q_n$, where $q = q_1$, $q_i = (l, v_i)$, satisfying:
  - $[\![\psi]\!](q_n) = 1$ and for $1 \leq i < n$: $[\![\phi]\!](q_i) = 1$
  - for each $1 \leq i < n$, if $q_i \to q_{i+1}$ is a continuous transition, then there exists a differentiable function $f : [0, t] \to \mathbb{R}^n$ for which:
    1. $f(0) = \mathbf{v_i}$ and $f(t) = \mathbf{v_{i+1}}$
    2. for all $\varepsilon \in (0, t)$, $f(\varepsilon) \in Inv(\ell)$, and $\dot{f}(\varepsilon) = F(\ell, f(\varepsilon))$
    3. for all $\varepsilon \in (0, t)$, $q' = (l_i, f(\varepsilon))$ satisfies $[\![\phi \vee \psi]\!](q') = 1$

- $[\![A(\phi \underline{\mathrm{U}} \psi)]\!](q_1) = 1$ iff for all runs ρ departing from q there exists a prefix $\rho^* := q_1 \to \ldots \to q_n$, where $q = q_1$, $q_i = (l, v_i)$, satisfying:
  - $[\![\psi]\!](q_n) = 1$ and for $1 \leq i < n$: $[\![\phi]\!](q_i) = 1$
  - for each $1 \leq i < n$, if $q_i \to q_{i+1}$ is a continuous transition, then there exists a differentiable function $f : [0, t] \to \mathbb{R}^n$ for which:
    1. $f(0) = \mathbf{v_i}$ and $f(t) = \mathbf{v_{i+1}}$
    2. for all $\varepsilon \in (0, t)$, $f(\varepsilon) \in Inv(\ell)$, and $\dot{f}(\varepsilon) = F(\ell, f(\varepsilon))$
    3. for all $\varepsilon \in (0, t)$, $q' = (l_i, f(\varepsilon))$ satisfies $[\![\phi \vee \psi]\!](q') = 1$

$H \vDash \phi$ iff $\forall q \in Q_0 : [\![\phi]\!](q) = 1$.

Since ABB abstractions encode both an over- and an under-approximation of the dynamics in the underlying hybrid automaton H, a preservative three valued semantics for the logic CTL can be derived thereof. Intuitively, the simulation preorder $H \leq_S A^n$ allows us to use unbounded runs for the falsification (resp. certification) of CTL-formulas of the form $E\phi \underline{\mathrm{U}} \psi$ (resp. $A\phi \underline{\mathrm{U}} \psi$), whereas the must-transitions induced by integer weights allow to deal with the other formulas. However, in contrast to the pure safety analysis, for CTL properties it is important to consider not only the target state of a given evolution, but also the intermediate configurations. In order to follow such intermediate configurations, we assume here to dispose of an additional edge-label: The label *dir*. Such a label simply distinguishes whether an integer weighted transition from α to β encodes a must-trajectory that passes *directly* from α to β^{n-1} (i.e. without never leaving these two regions).

Given the above premises, Definition 5.3 formalizes our three valued semantics for CTL on ABB abstractions.

Definition 5.3 [Three Valued CTL Semantics on ABBs] Let H be a hybrid automaton, let AP be a finite set of atomic propositions, and let $\mathcal{P}$ be a partition of the state space of H consistent w.r.t. the labeling function $l_{AP} : Q \to 2^{|AP|}$. Consider the succession of ABB abstractions, $\{A^i\}_{i=1,\ldots,n}$, computed w.r.t. the initial partition $\mathcal{P}$. Then the semantics of CTL on A^n is recursively defined as follows:

- If ϕ is an atomic proposition, then $[\![\phi]\!](\alpha) = \begin{cases} 1 & \phi \in l_{AP}(\alpha) \\ 0 & \textit{otherwise} \end{cases}$
- $[\![\neg\phi]\!] := \neg_3 [\![\phi]\!]$
- For $\diamond \in \{\vee, \wedge\}$: $[\![\phi \diamond \psi]\!] := [\![\phi]\!] \diamond_3 [\![\psi]\!]$
- Let $\{\alpha_i\}_{i \in \mathbb{N}}$ denote an infinite path of $\mathcal{A}^n$ starting in $\alpha_1 = \alpha$. Then:

$$[\![E(\varphi \underline{\mathrm{U}} \psi)]\!](\alpha) = \begin{cases} 1 & [\![\psi]\!](\alpha) = 1 \ \textit{or} \\ & \exists \alpha \overset{dir,m}{\to} \beta : \ [\![\varphi]\!](\alpha) = 1 \wedge [\![E(\varphi \underline{\mathrm{U}} \psi)]\!](\beta^{n-1}) = 1 \\ 0 & \forall \{\alpha_i\}_{i \in \mathbb{N}} \forall k \in \mathbb{N} : [\![\psi]\!](\alpha_k) \neq 0 \Rightarrow \exists j < k : [\![\varphi]\!](\alpha_j) = 0 \\ \bot & \textit{otherwise} \end{cases}$$

$$[\![A(\varphi \underline{\mathrm{U}} \psi)]\!](\alpha) = \begin{cases} 1 & \forall \{\alpha_i\}_{i \in \mathbb{N}} \exists k \in \mathbb{N} : [\![\psi]\!](\alpha_k) = 1 \wedge \forall j < k : [\![\varphi]\!](\alpha_j) = 1 \\ 0 & [\![\psi \vee \varphi]\!](\alpha) = 0 \vee \exists \alpha \overset{dir,m}{\rightarrow} \beta : [\![A(\varphi \underline{\mathrm{U}} \psi)]\!](\beta^{n-1}) = 0 \\ \perp & \textit{otherwise} \end{cases}$$

Theorem 5.4, below, states the desired result of preservation for our three valued semantics. Note that for a CTL formula ϕ, we distinguish between the semantics $[\![\phi]\!]_H$ on a hybrid automaton H (as given in Definition 5.1) and the semantics $[\![\phi]\!](r)$ on the region α of an ABB abstraction for H (as given in Definition 5.3).

Theorem 5.4 (Preservation Theorem) *Let H be a hybrid automaton and let A be an n-ABB abstraction of H. Then, for any CTL formula ϕ,*

$$[\![\phi]\!](\alpha) = 1 \Rightarrow \ \forall\, x \in \alpha : \ [\![\phi]\!]_H(x) = 1$$
$$[\![\phi]\!](\alpha) = 0 \Rightarrow \ \forall\, x \in \alpha : \ [\![\phi]\!]_H(x) = 0$$

Proof. We proceed by induction on the bound n for the ABB-abstraction $\mathcal{A}^n$. The base case is immediate. To prove our claim for $n > 0$ we use a structural inductive argument. Atomic propositions and boolean operations can be easily dealt with. For the temporal operators, let us start to consider the case $[\![E(\varphi \underline{\mathrm{U}} \psi)]\!](\alpha) = 1$. By Definition 5.3 $[\![E(\varphi \underline{\mathrm{U}} \psi)]\!](\alpha) = 1$ iff either $[\![\psi]\!](\alpha) = 1$ or $\exists \alpha \overset{m,dir}{\rightarrow} \beta : [\![\varphi]\!](\alpha) = 1 \wedge [\![E(\varphi \underline{\mathrm{U}} \psi)]\!](\beta^{n-1}) = 1$. In the first case the claim holds by our structural inductive hypothesis. In the second case, the edge $\alpha \overset{m,dir}{\rightarrow} \beta$ ensures that for any $x \in \alpha$, x admits an evolution to β^{n-1} in H, which traverses the only regions α, β^{n-1}. Moreover, α satisfies φ and β^{n-1} satisfies $E(\varphi \underline{\mathrm{U}} \psi)$. Thus, using our induction on n as well as our structural induction, we can conclude that for each $x \in \alpha$, $[\![E(\varphi \underline{\mathrm{U}} \psi)]\!]_H(x) = 1$. The case for which $[\![E(\varphi \underline{\mathrm{U}} \psi)]\!](\alpha) = 0$ can be easily established on the ground of the simulation preorder relating $\mathcal{A}^n$ to H, stated in Theorem 4.2. The boolean evaluations of the operator $A\underline{\mathrm{U}}$ can be simmetrically dealt with. □

6 Conclusions

We proposed a novel framework for the reachability analysis of hybrid automata. Our method is based on the definition of a succession of abstractions, whose construction does not relies on the exact representation of the predecessor (resp. successor) regions. Nevertheless, our ABB abstractions encode enough information to both prove and falsify safety properties, as well as more general CTL formulas. In particular, we provide efficient (linear) algorithms for the three valued safety analysis on ABB abstractions.

References

[1] R. Alur, C. Courcoubetis, and Henzinger, T. A., "Computing Accumulated Delays in Real-Time Systems" Formal Methods in System Design **11**, 1997, pp. 137–156.

[2] R. Alur and D. L. Dill "A theory of timed automata" Theoretical Computer Science **126**, 1994, pp. 183–235.

[3] R. Alur and T. A. Henzinger and P. Ho, "Automatic Symbolic Verification of Embedded Systems" IEEE Real-Time Systems Symposium, 1993, pp. 2–11.

[4] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems" Proceedings of the IEEE **88**, 2000, pp. 971–984.

[5] T. Brihaye, C. Michaux, C. Rivier, and C. Troestler, "On O-Minimal Hybrid Systems" Hybrid Systems: Computation and Control HSCC, LNCS **2993**, 2004.

[6] M. Fränzle, "What Will Be Eventually True of Polynomial Hybrid Automata?" Proceedings of 4th International Symposium on Theoretical Aspects of Computer Software, LNCS **2215**, 2001, pp. 340–359.

[7] R. Gentilini, K. Schneider, and B. Mishra, "Successive Abstractions of Hybrid Automata for Monotonic CTL Model Checking" , Logical Foundations of Computer Science, International Symposium, LFCS 2007, LNCS **4514**, pp. 224–240.

[8] R. Ghosh, A. Tiwari, and C. Tomlin, "Automated Symbolic Reachability Analysis with Application to Delta-Notch Signaling Automata" Hybrid Systems: Computation and Control HSCC, LNCS **2623**, 2003.

[9] R. Ghosh and C. J. Tomlin, "Lateral Inhibition through Delta-Notch Signaling: A Piecewise Affine Hybrid Model" LNCS **2034**, 2001, pp. 232–245.

[10] T. A. Henzinger, "The Theory of Hybrid Automata." Proceedings of the 11th IEEE Symposium on Logic in Computer Science, IEEE Computer Society, 1996, pp. 278–292.

[11] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke, "Computing simulations on finite and infinite graphs" Proceedings of the 36th Annual Symposium on Foundations of Computer Science, IEEE, 1995, pp. 453–462.

[12] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" Proceedings of the 27th Symposium on Theory of Computing, ACM Press, 1995, pp. 373–382.

[13] P. C. Kannellakis, and S. A. Smolka, "CCS Expressions, Finite State Processes, and Three Problems of Equivalence" Information and Computation **86**, 1990, pp. 43–68.

[14] S. C. Kleene, "Introduction to Metamathematics" Wolters-Noordhoff 1971.

[15] G. Lafferriere, J. G. Pappas, and S. Yovine "A New Class of Decidable Hybrid Systems" Proceedings of the 2nd International Workshop on Hybrid Systems, LNCS, 1999, pp. 137–151.

[16] G. Lafferriere, G. Pappas, and S. Sastry, "O-minimal hybrid systems" Mathematics of Control, Signals, and Systems, Springer-Verlag **13**, 2000, pp. 1–21.

[17] J. S. Miller, "Decidability and Complexity Results for Timed Automata and Semi-linear Hybrid Automata." Proceedings of the 3th International Workshop on Hybrid Systems, LNCS, 2000, pp. 296–309.

[18] R. Paige, and R. E. Tarjan, "Three partition refinement algorithms" SIAM Journal on Computing **16**, 1987, pp. 973–989.

[19] C. Piazza, M. Antoniotti, V. Mysore, A. Policriti, F. Winkler, and B. Mishra, "Algorithmic Algebraic Model Checking I: Challenges from Systems Biology." Proceedings of the 17th International Conference on Computer Aided Verification, LNCS **3576**, pp. 5–19.

[20] S. Ratschan and Z. She. "Safety Verification of Hybrid Systems by Constraint Propagation Based Abstraction Refinement." Proceedings of the 8th International Workshop on Hybrid Systems: Computation and Control, LNCS **3414**, pp. 573-589.

[21] A. Tiwari and G. Khanna, "Series of Abstractions for Hybrid Automata" Proceedings of the 5th International Workshop on Hybrid Systems, LNCS, 2002, pp.465–478.