Successive Abstractions of Hybrid Automata for Monotonic CTL Model Checking

R. Gentilini[2], K. Schneider[2] and B. Mishra[1,3]

[1] Courant Institute, New York University, New York, NY, U.S.A.
[2] University of Kaiserslautern, Department of Computer Science, Germany
[3] NYU School of Medicine, New York University, New York, NY, U.S.A.
{gentilin, Klaus.Schneider}@informatik.uni-kl.de, mishra@nyu.edu

Abstract. Current symbolic techniques for the automated reasoning over undecidable hybrid automata, force one to choose between the refinement of either an overapproximation or an underapproximation of the set of reachable states. When the analysis of branching time temporal properties is considered, the literature has developed a number of abstractions techniques based on the simulation preorder, that allow the preservation of only true universally quantified formulæ.
This paper suggests a way to surmount these difficulties by defining a succession of abstractions of hybrid automata, which not only (1) allow the detection and the refinement of both over- and under-approximated reachable sets symmetrically, but also (2) preserves the full set of branching time temporal properties (when interpreted on a dense time domain). Moreover, our approach imposes on the corresponding set of abstractions a desirable monotonicity property with respect to the set of model-checked formulaæ.

1 Introduction

Over the past few years, questions related to the analysis of *hybrid automata* [10] have occupied a considerable amount of attention and interest within the automatic verification research community, since the consequent models provide a high fidelity representation of real world (embedded) systems, and yet the nontrivial computational problems they raise do not yield to the classical techniques either of applied mathematics or of theoretical computer science.

As originally envisioned in [10, 9], hybrid automata have aspired to combine the traditional *automata* tools from logic and computer science with differential equation systems, and their long tradition in mathematics. In this respect, the enormous potentials of hybrid automata in challenging applications fields—namely, the analysis of embedded, real time, and biological systems, to cite only a few of them—were immediately recognized. However, the trade-off between the representational fidelity of hybrid automata and the solvability of related decidability problems addressing properties such as *reachability*, was also immediately apparent. Hence, the major effort of the hybrid automata research community, to date, has been devoted to the study of *decidable* classes of hybrid automata, for which at least the reachability problem remains decidable [10, 9, 13, 14, 2]. Listed in their chronological order, the (main) decidable families in the literature are the ones corresponding to *timed automata* [1], *singular automata* [10, 9], *rectangular automata* [9], and *o-minimal automata* [13]. Unfortunately, for each one of the above families, the sacrifice in the expressiveness of either the discrete or the continuous dynamics [2] that has to be exacted in exchange for the decidability result, strongly casts doubt on the possibility of faithfully capturing complex hybrid dynamics arising, for example, in the system biology area [16, 8].

Motivated by the reasons listed above, many authors have recently focused on developing techniques for the symbolic analysis of undecidable—and yet reasonably expressive—hybrid automata [16, 8, 19, 6, 17]. However, any method developed so far relies either on the definition of abstractions simulating the underlying hybrid automata [8, 19, 17] or on symbolic bounded reachability techniques [16, 6]. In the first case, only an overapproximation of the reachable state-space is possible. Usually, those techniques target the proof of *safety property*, stating that something undesirable should never happen on any reachable state of the system. In general, the simulation preorder from the abstraction to the hybrid automaton allows for preservation of only *true* formulæ in the *universal fragment* of a branching time temporal logic. In the second case, only an underapproximation of the reachable state-space can be explored and used for generating counterexamples to the reactive system properties of interest (e.g. safety).

In this paper we develop a framework to *both prove and disprove* reactive system properties expressed by means of CTL logic [4, 18] on (undecidable) hybrid automata. To the best of authors' knowledge, no other symbolic technique for the analysis of undecidable hybrid automata can be claimed to preserve both true and false reactive systems properties simultaneously. Our framework is based on the design of a succession of abstraction and a corresponding three valued semantics for the logic CTL, allowing for the monotonic preservation of true/false formulæ along the succession of abstractions. Given a structure $\mathcal{A}$ in our succession, we finally show that the three valued CTL model checking problem on $\mathcal{A}$ is linear in the length of the formula and in the size of the abstraction. Because of the space constraints, we omit the proofs of the results shown here, but collect them in [7].

2 Preliminaries

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

Definition 1 (Hybrid Automata [2]). *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 $X = \{x_1, \ldots x_n\}$ *that take values in* $\mathbb{R}$
- *an initial set of conditions: Init* $\subseteq L \times \mathbb{R}^n$
- *Inv:* $L \mapsto 2^{\mathbb{R}^n}$*, the* invariant location labeling
- $F : L \times \mathbb{R}^n \mapsto \mathbb{R}^n$*, assigning to each location* $\ell \in L$ *a vector field* $F(\ell, \cdot)$ *that defines the evolution of continuous variables within* ℓ
- $G : E \mapsto 2^{\mathbb{R}^n}$*, the* guard edge labeling
- $R : E \times \mathbb{R}^n \mapsto 2^{\mathbb{R}^n}$*, the* reset edge labeling.

We write $\mathbf{v}$ to represent a valuation $(v_1, \ldots, v_n) \in \mathbb{R}^n$ of the variables' vector $\mathbf{x} = (x_1, \ldots, x_n)$, 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 $H = (L, E, X, \mathit{Init}, \mathit{Inv}, F, G, R)$, starts at any $(\ell, \mathbf{v}) \in \mathit{Init}$ 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 steps in the *time abstract transition system* of H, defined below:

Definition 2. *The* time abstract transition system *of the hybrid automaton* $H = (L, E, X, \mathit{Init}, \mathit{Inv}, F, G, R)$ *is the transition system* $T_H = (Q, Q_0, \ell_{\rightarrow}, \rightarrow)$, *where:*

- $Q \subseteq L \times \mathbb{R}^n$ *and* $(\ell, \mathbf{v}) \in Q$ *if and only if* $\mathbf{v} \in \mathit{Inv}(\ell)$
- $Q_0 \subseteq Q$ *and* $(\ell, \mathbf{v}) \in Q_0$ *if and only if* $\mathbf{v} \in \mathit{Init}(\ell) \cap \mathit{Inv}(\ell)$
- $E \cup \{\delta\}$ *is the set of edge labels, that are determined as follows:*
  - *there is a* continuous transition $(\ell, \mathbf{v}) \xrightarrow{\delta} (\ell, \mathbf{v}')$, *if and only if there is a differentiable function* $f : [0, t] \rightarrow \mathbb{R}^n$, *with* $\dot{f} : [0, t] \rightarrow \mathbb{R}^n$ *such that:*
    1. $f(0) = \mathbf{v}$ *and* $f(t) = \mathbf{v}'$
    2. *for all* $\varepsilon \in (0, t)$, $f(\varepsilon) \in \mathit{Inv}(\ell)$, *and* $\dot{f}(\varepsilon) = F(\ell, f(\varepsilon))$.
  - *there is a* discrete transition $(\ell, \mathbf{v}) \xrightarrow{e} (\ell', \mathbf{v}')$ *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 $T_H = (Q, Q_0, \ell_{\rightarrow}, \rightarrow)$. Given a region B and a transition label $a \in \ell_{\rightarrow}$, the predecessor region $Pre_a(B)$ is defined as the region $\{q \in Q \mid \exists q' \in B. q \xrightarrow{a} q'\}$. The *bisimulation* and the *simulation* relations are two fundamental tools in the context of hybrid automata abstraction.

Definition 3 (Bisimulation). *Let* $T^1 = (Q^1, Q^1_0, \ell^1_{\rightarrow}, \rightarrow^1)$, $T^2 = (Q^2, Q^2_0, \ell^2_{\rightarrow}, \rightarrow^2)$ *be two edge-labeled transition systems and let* $\mathcal{P}$ *be a partition on* $Q_1 \cup Q_2$. *A* bisimulation *for* T_1, T_2 *is a nonempty relation on* $\equiv_B \subseteq Q_1 \times Q_2$ *such that, for all* $p \equiv_B q$ *it holds:*

- $p \in Q^1_0$ *iff* $q \in Q^2_0$ *and* $[p]_{\mathcal{P}} = [q]_{\mathcal{P}}$, *where* $[p]_{\mathcal{P}}$ *denotes the class of* q *in* $\mathcal{P}$.
- *for each label* $a \in \ell_{\rightarrow}$, *if there exists* p' *such that* $p \xrightarrow{a} p'$, *then there exists* q' *such that* $p' \equiv_B q'$ *and* $q \xrightarrow{a} q'$.
- *for each label* $a \in \ell_{\rightarrow}$, *if there exists* q' *such that* $q \xrightarrow{a} q'$, *then there exists* p' *such that* $p' \equiv_B q'$ *and* $p \xrightarrow{a} p'$.

If there exists a bisimulation relation for T_1, T_2, *then* T_1 *and* T_2 *are* bisimilation equivalent *(or* bisimilar*), denoted* $T_1 \equiv_B T_2$.

Definition 4 (Simulation). *Let* $T^1 = (Q^1, Q^1_0, \ell^1_{\rightarrow}, \rightarrow^1)$, $T^2 = (Q^2, Q^2_0, \ell^2_{\rightarrow}, \rightarrow^2)$ *be two edge-labeled transition systems and let* $\mathcal{P}$ *be a partition on* $Q_1 \cup Q_2$. *A* simulation *from* T_1 *to* T_2 *is a nonempty relation on* $\leq_S \subseteq Q_1 \times Q_2$ *such that, for all* $p \leq_S q$:

- $p \in Q^1_0$ *iff* $q \in Q^2_0$ *and* $[p]_{\mathcal{P}} = [q]_{\mathcal{P}}$.
- *for each label* $a \in \ell_{\rightarrow}$, *if there exists* p' *such that* $p \xrightarrow{a} p'$, *then there exists* q' *such that* $p' \leq_S q'$ *and* $q \xrightarrow{a} q'$.

If there exists a simulation from T_1 to T_2, then we say that T_2 simulates *T_1, denoted $T_1 \leq_S T_2$. If $T_1 \leq_S T_2$ and $T_2 \leq_S T_1$, then T_1 and T_2 are said to be* similation equivalent *(or* similar*) and we write $T_1 \equiv_S T_2$.*

Definition 6 recapitulates the semantics of the temporal logic CTL (where the neXt temporal operator is omitted because of the density of the underlying time framework) on hybrid automata [1, 10].

Definition 5 **(CTL for Hybrid Automta).** *Let AP be a finite set of propositional letters and $p \in \mathsf{AP}$. CTL is the set of formulæ defined by the following syntax:*

$$\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \mathsf{E}\phi_1\mathsf{U}\phi_2 \mid \mathsf{A}\phi_1\mathsf{U}\phi_2 \mid \mathsf{E}\phi_1\mathsf{R}\phi_2 \mid \mathsf{A}\phi_1\mathsf{R}\phi_2$$

Definition 6 **(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 $\ell_{\mathsf{AP}} : L \times X \mapsto 2^{\mathsf{AP}}$. Given $\phi \in \mathsf{CTL}$ and $s \in Q$, $s \models \phi$ is inductively defined as follows:*

- $s \models \mathsf{p}$ *if and only if* $\mathsf{p} \in \ell_{\mathsf{AP}}(s)$
- $s \models \neg\phi$ *if and only if not* $s \models \phi$
- $s \models \phi_1 \vee \phi_2$ *if and only if* $s \models \phi_1$ *or* $s \models \phi_2$
- $s \models \mathsf{E}\phi_1\mathsf{U}\phi_2$ *if and only if there exists a run ρ and a time t such that:*
  - $\cdot$ $\rho(t) \models \phi_2$
  - $\cdot$ $\forall t' \leq t\ (\rho(t') \models \phi_1 \vee \phi_2)$
- $s \models \mathsf{A}\phi_1\mathsf{U}\phi_2$ *if and only if for each run ρ there exists a time t such that:*
  - $\cdot$ $\rho(t) \models \phi_2$
  - $\cdot$ $\forall t' \leq t\ (\rho(t') \models \phi_1 \vee \phi_2)$
- $s \models \mathsf{E}\phi_1\mathsf{R}\phi_2$ *iff* $s \models \neg(\mathsf{A}\neg\phi_1\mathsf{U}\neg\phi_2)$
- $s \models \mathsf{A}\phi_1\mathsf{R}\phi_2$ *iff* $s \models \neg(\mathsf{E}\neg\phi_1\mathsf{U}\neg\phi_2)$

$H \models \phi$ iff for each $s \in Q_0$, $s \models \phi$.

2.1 O-Minimal Theories and O-Minimal Hybrid Automata

In this subsection, we give a brief introduction to *order minimality* (o-minimality) which is used to define o-minimal hybrid automata. We refer to [21, 20, 22] for a more comprehensive introduction to o-minimality.

Consider a structure over the reals, $\mathcal{M} = \langle \mathbb{R}, <, \ldots \rangle$, where the underlying language includes at least a binary relation interpreted as the usual total order over $\mathbb{R}$. The *theory* $\mathrm{Th}(\mathcal{M})$ associated to $\mathcal{M}$ is the set of first order sentences that hold in $\mathcal{M}$. A set $Y \subseteq \mathbb{R}^n$ is *definable* in $\mathcal{M}$ if and only if there exists a first order formula $\psi(x_1, \ldots, x_n)$ such that $Y = \{(a_1, \ldots, a_n) | \mathcal{M} \models \psi(a_1, \ldots, a_n)\}$. A map $f : A \mapsto \mathbb{R}^n$ with $A \subseteq \mathbb{R}^m$ is definable in $\mathcal{M}$ if and only if its graph $\Gamma(f) \subseteq \mathbb{R}^m \times \mathbb{R}^n$ is definable in $\mathcal{M}$.

Definition 7 **(O-Minimal Structure).** *The structure $\mathcal{M} = \langle \mathbb{R}, <, \ldots \rangle$ is o-minimal if and only if every definable subset of $\mathbb{R}$ is a finite union of points and (possibly unbounded) intervals. In this case, the theory $\mathrm{Th}(\mathcal{M})$ is also said to be o-minimal.*

Given an o-minimal structure $\mathcal{M} = \langle \mathbb{R}, <, \ldots \rangle$, the notion of set definability is closed under each boolean set composition operation, cartesian product, and projection. The notion of map definability is closed under composition, cartesian product, and projection. In the following, we will use the same symbol to denote both a given o-minimal structure and the corresponding theory, omitting the term $\mathrm{Th}(\cdot)$.

The class of o-minimal structures over the reals is quite rich: both the structure $\mathsf{Li}(\mathbb{R}) = (\mathbb{R}, <, +, -, 0, 1)$, used to express linear constraints over the reals, and the ordered real field $\mathsf{OF}(\mathbb{R}) = (\mathbb{R}, <, +, -, *, 0, 1)$ are o-minimal. The extensions of the above structures by the exponential function are also o-minimal. Another important extension is obtained by restricted analytic functions. Further extensions are discussed in [13]. The variety of o-minimal theories over the reals ensures that the family of *o-minimal hybrid automata* as introduced in [13, 14] (cf. Definition 9, below) constitutes a large and important family of hybrid automata, admitting powerful continuous evolutions. In the following definitions, we will adopt the notation used in [13].

Definition 8. *Let $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ be a smooth vector field on $\mathbb{R}^n$. For each $\mathbf{v} \in \mathbb{R}^n$, let $\gamma_{\mathbf{v}}(t)$ denote the integral curve of F which passes through $\mathbf{v}$ at $t = 0$, that is $\dot{\gamma}_{\mathbf{v}}(t) = F(\gamma_{\mathbf{v}}(t))$ and $\gamma_{\mathbf{v}}(0) = \mathbf{v}$. We say that F is* complete *if, for each $\mathbf{v} \in \mathbb{R}^n$, $\gamma_{\mathbf{v}}(t)$ is defined for all t. For such an F, the* flow *of F is the function $\phi : \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}^n$ given by $\phi(\mathbf{v}, t) = \gamma_{\mathbf{v}}(t)$.*

Definition 9 (O-Minimal Hybrid Automata [13]). *The hybrid automaton $H = (L, E, X,$* Init, Inv, *$F, G, R)$ is* o-minimal *if the following holds:*

a) for each $\ell \in L$ the smooth vector field $F(\ell, \cdot)$ is complete
*b) for each $(\ell, \ell') \in E$, the reset function $R : E \mapsto \mathbb{R}^n$ does not depend on continuous variables (*constant resets*)*
c) for each $\ell \in L$ and $(\ell, \ell') \in E$, the sets $Inv(\ell)$, $R(\ell, \ell')$, $G(\ell)$, $Init(\ell)$, and the flow of $F(\ell, \cdot)$ are definable in the same o-minimal structure.

Given an o-minimal structure $\mathcal{M}$, the o-minimal hybrid automata induced by $\mathcal{M}$ are called o-minimal($\mathcal{M}$) hybrid automata. O-minimal hybrid automata admit a finite bisimulation quotient [13]. Computability of such a bisimulation quotient (and hence decidability) depends on the underlying o-minimal structure: in [14], the class of o-minimal ($\mathsf{OF}(\mathbb{R})$) hybrid automata and various subclasses of o-minimal ($\mathsf{OF}_{\mathsf{exp}}(\mathbb{R})$) automata were proven to be decidable with respect to reachability.

3 A Succession of Abstractions for Monotonic CTL Model Checking on Hybrid Automata

Throughout this section and Section 4, we solve the problem of defining a succession of abstractions for hybrid automata and a corresponding three valued semantics for CTL formulæ with the following property: whenever a CTL formula is true (false) on a given abstraction, its value is preserved on the hybrid automaton. Moreover, we require that the set of formulæ evaluating to $\perp$ (according to the three valued semantics over the abstractions) decreases monotonically in its size along the succession of abstractions.

Such a requirement is reminiscent of the usual *regularity* property for Kleene three valued logics [12] (and its many variants [5]), according to which the behavior of the third value is compatible with any increase of information[1].

The first idea we exploit is based on the classical notion of *n-bounded bisimulation*. In fact, a simulation preorder relates successively finer bounded bisimulations, thus enabling the establishment of a "monotonic truth-preservation result" for the set of true universally quantified formulæ along this succession of abstractions. Since the n-bounded bisimulation is characterized by the n-bounded modal logic (where at most n neXt operators are admitted in the formulæ), it is possible to recover preservation for non universal CTL formulæ, by evaluating them on n-bounded paths. The preceding ideas can be effectively developed for the analysis of infinite *discrete* transition systems. However, in our context infinite transition systems represent mixed continuous/discrete systems. Hence, first the neXt temporal operator is meaningless in the corresponding logics. Second, a query such as $s \models \mathsf{E}\phi_1\mathsf{U}\phi_2$ can never be checked by considering only a finite path departing from s in the time abstract transition system T_H of a hybrid automaton H. In fact, due to the dense nature of the underlying time framework, each finite path of the kind $z \xrightarrow{\delta} z'$ subsumes an uncountable number of continuous transitions' infinite paths in T_H, on which $\mathsf{E}\phi_1\mathsf{U}\phi_2$ needs to be established. In other words, while for Kripke structures modelling (infinite) discrete dynamical systems, there is a nice correspondence between the index of the bounded bisimulation and the length of path that can be trusted, in the case of hybrid automata this is lost. More precisely, runs that can be trusted in successive finer bounded bisimulations could be never allowed to traverse more than two locations, since they are abstracted by paths containing more and more continuous transitions.

3.1 Discrete Bounded Bisimulation Abstraction

Motivated by the discussion above, we develop here a new succession of hybrid automata abstractions suitable for our purposes, and refer to them as *discrete bounded bisimulation* abstractions. It is well known that the classic bisimulation equivalence can be characterized as a coarsest partition stable with respect to a given transition relation [11]. Bounded bisimulation imposes a bound on the number of times each edge can be used for partition refinement purposes. For discrete bounded bisimulation, the latter bound applies only to discrete edges. Formally, our discrete bounded bisimulation abstractions are inductively defined in Definition 10.

Definition 10 (Discrete Bounded Bisimulation (DBB)). *Consider the time abstract transition system $T_H = (Q, Q_0, \ell_{\rightarrow}, \rightarrow)$ of a hybrid automaton H, and let $\mathcal{P}$ be a partition on Q:*

1. *$\equiv_0 \in Q \times Q$ is the maximum relation on Q such that for all $p, q \in Q$:*
   if $p \equiv_0 q$ then (a) $[p]_{\mathcal{P}} = [q]_{\mathcal{P}}$ *and* $p \in Q_0$ *iff* $q \in Q_0$
   (b) $\forall p'(p \xrightarrow{\delta} p' \Rightarrow \exists q'(p' \equiv_0 q' \wedge q \xrightarrow{\delta} q')$
   (c) $\forall q'(q \xrightarrow{\delta} q' \Rightarrow \exists p'(p' \equiv_0 q' \wedge p \xrightarrow{\delta} p')$

[1] Note that, in our framework, the lack of information is inherent in the *abstraction* of the hybrid automaton model, rather than in the (indefinite) value of some propositional letter.

2. *Given $n \in \mathbb{N}^+$, $\equiv_n$ is the maximum relation on Q such that for all $p, q \in Q$:*
   if $p \equiv_n q$ then
$$\begin{aligned}
&(a)\ p \equiv_{n-1} q\\
&(b)\ \forall p'(p \xrightarrow{\delta} p' \Rightarrow \exists q'(p' \equiv_n q' \wedge q \xrightarrow{\delta} q')\\
&(c)\ \forall q'(q \xrightarrow{\delta} q' \Rightarrow \exists p'(p' \equiv_n q' \wedge p \xrightarrow{\delta} p')\\
&(d)\ \forall p'(p \xrightarrow{e} p' \Rightarrow \exists q'(p' \equiv_{n-1} q' \wedge q \xrightarrow{e} q')\\
&(e)\ \forall q'(q \xrightarrow{e} q' \Rightarrow \exists p'(p' \equiv_{n-1} q' \wedge p \xrightarrow{e} p')
\end{aligned}$$

Given $n \in \mathbb{N}$, the relation $\equiv_n$ will be referred to as n-DBB equivalence.

Definition 11 (Succession of DBB Abstractions). *Let $T_H = (Q, Q_0, \ell_{\rightarrow}, \rightarrow)$ be the time abstract transition system of the hybrid automaton H, let $\mathcal{P}$ be a partition on Q, and consider the n-DBB equivalence $\equiv_n$. The n-DBB abstraction structure $H_{/\equiv_n} = (Q', Q'_0, \ell'_{\rightarrow}, \rightarrow)$ is defined as:*

- *$Q' = Q_{/\equiv_n}$, $Q'_0 = Q_{0/\equiv_n}$ and $\ell'_{\rightarrow} = \ell_{\rightarrow}$.*
- *$\forall \alpha, \beta \in Q'$:*
  - *$\alpha \xrightarrow{e} \beta$ iff $\exists s \in \alpha, \exists q \in \beta(s \xrightarrow{e} q))$*
  - *$\alpha \xrightarrow{\delta} \beta$ iff $\exists s \in \alpha, \exists q \in \beta(s \xrightarrow{\delta} q$ by traversing the only regions α and $\beta)$*

Lemma 1 establishes some folk theorems describing few properties of discrete bounded bisimulation, and can be easily proved using an inductive argument. Among them, we remark the existence of a simulation preorder relating successive elements in our succession of DBB abstractions. The latter property allows one to use the succession of DBB structures to refine an overapproximation of the underlying hybrid automaton reachable set.

Lemma 1. *Let H be a hybrid automaton, and consider the succession of n-DBB abstractions $\langle H_{/\equiv_n} \rangle_{n \in \mathbb{N}}$. For all $n \in \mathbb{N}$:*

- *$T_H \leq_S H_{/\equiv_n}$ and $H_{/\equiv_{n+1}} \leq_S H_{/\equiv_n}$.*
- *If $H_{/\equiv_n}$ coincides with $H_{/\equiv_{n+1}}$, then $T_H \equiv_B H_{/\equiv_n}$.*

As a consequence of Lemma 2, it is also possible to use the succession of DBB abstractions to obtain $\subseteq$-monotonic underapproximations of the underlying hybrid automaton reachable set. More precisely, $H_{/\equiv_n}$ preserves the reachability of a given region of interest (in the initial partition), whenever the latter can be established on H following a path that traverses at most n locations. Given two states in a hybrid automaton H, we use the notation $q \overset{n}{\rightsquigarrow} q'$ to state that q' is reachble from q following a run that contains at most n discrete edges (i.e. traverses at most n locations of H).

Lemma 2. *Let p and q be two states in a hybrid automaton H and let $\equiv_n$ be the n-DBB equivalence on T_H with respect to a partition $\mathcal{P}$. If $p \equiv_n q$, then for all $m \leq n$ it holds that:*

- *For all p' such that $p \overset{m}{\rightsquigarrow} p'$, there exists q' such that $p' \equiv_{n-m} q'$ and $q \overset{m}{\rightsquigarrow} q'$.*
- *For all q' such that $q \overset{m}{\rightsquigarrow} q'$, there exists p' such that $p' \equiv_{n-m} q'$ and $p \overset{m}{\rightsquigarrow} p'$.*

3.2 Finiteness and Computability of DBB Abstractions

Figure 1 presents a semi-decision procedure to obtain the n-DBB equivalence on the time-abstract transition system of a hybrid automaton H. Such a semi-decision procedure takes as input the hybrid automaton H, the bound n, and an initial (finite) partition $\mathcal{P}_0$ over the state-space of H. As stated in Lemma 3, it constitutes an effective algorithm for n-DBB equivalence whenever it is computable and gets to termination. Clearly, while computability depends on disposing of opportune symbolic techniques to represent and manipulate sets of states, termination is related to the n-DBB quotient finiteness.

Lemma 3. *Let $n \in \mathbb{N}$; let H be a hybrid automaton; and let $\mathcal{P}_0$ be a finite partition over the state-space of H. If* $\mathrm{DBB}(n, H, \mathcal{P}_0)$ *terminates, then algorithm* $\mathrm{DBB}(n, H, \mathcal{P}_0)$ *computes the quotient of the n-DBB equivalence with respect to $\mathcal{P}_0$ on T_H.*

Using a number of techniques developed in [13, 14], it is rather easy to obtain n-DBB finiteness and computability results for the broad *undecidable* family of *fully o-minimal hybrid automata* (cfr. Definition 12). The latter extends the o-minimal based systems in [13] by admitting arbitrary o-minimal functions as resets, in place of constant functions. Such a relaxation in the formulation of the discrete dynamics allows the family to encompass several classes of hybrid automata for which the reachability problem has been proven undecidable (e.g. the class of uninitialized rectangular automata [10, 9], or the undecidable classes studied in [3, 15]).

Definition 12 (Fully O-Minimal Hybrid Automata). *The hybrid automaton $H = (L, E, X$, Init, Inv, $F, G, R)$ is* fully o-minimal *iff:*

a) for each $\ell \in L$ the smooth vector field $F(\ell, \cdot)$ is complete
b) for each $\ell \in L$ and $(\ell, \ell') \in E$, the sets $Inv(\ell)$, $G(\ell)$, $Init(\ell)$, the reset function $R(\ell, \ell') : \mathbb{R}^{|X|} \mapsto \mathbb{R}^{|X|}$ and the flow of $F(\ell, \cdot)$ are definable in the same o-minimal structure.

Theorem 1. *Let H be a fully o-minimal($\mathcal{M}$) hybrid automaton, and let $\mathcal{P}$ be an initial finite partition over the state-space of H definable in $\mathcal{M}$. Then, the algorithm* $\mathrm{DBB}(n, H, \ell_Q)$ *terminates for any $n \in \mathbb{N}$.*

By Theorem 1, the whole family of fully o-minimal automata have for all $n \in \mathbb{N}$ a finite n-DBB abstraction structure. Computability of such a finite abstraction is instead parameterized with respect to the theory underlying fully o-minimal automata. In particular, as stated in Corollary 1, the class of fully o-minimal ($\mathsf{OF}(\mathbb{R})$) hybrid automata has a finite and effectively computable n-DBB abstraction. The result depends on the fact that $\mathsf{OF}(\mathbb{R})$ is a decidable theory admitting quantifier elimination. Therefore, the theory $\mathsf{OF}(\mathbb{R})$ provides the means for representing sets, computing post-images and boolean compositions, as well as checking for set emptiness. Techniques similar to the ones adopted in [13] can be used to obtain further computability results on subclasses of o-minimal($\mathsf{OF}_{\mathsf{exp}}(\mathbb{R})$) hybrid automata.

Corollary 1. *Given $n \in \mathbb{N}$, the n-DBB abstraction on fully o-minimal ($\mathsf{OF}(\mathbb{R})$) hybrid automata is finite and computable.*

```
DBB$(n, H, \mathcal{P}_0)$
(1)  Let $\mathcal{P}$ be the coarsest partition refining $\mathcal{P}_0$ compatible with $Q_0$
/*————Compute 0-DBB equivalence quotient————————————*/
(2)  while $(\exists B, B' \in \mathcal{P}$ such that $\emptyset \neq B \cap Pre_\tau(B') \neq B)$
(3)      $B_1 \leftarrow B \cap Pre_\tau(B'); B_2 \leftarrow B \setminus Pre_\tau(B')$;
(4)      $\mathcal{P} \leftarrow (\mathcal{P} \setminus \{B\}) \cup \{B_1, B_2\}$;
/*————Perform $n$ refinement steps to obtain $n$-DBB equivalence quotient—*/
(5)  while $(n > 0)$
(6)      $n \leftarrow n - 1$; $\mathcal{P}_{old} \leftarrow \mathcal{P}$;
(7)      for each $(e = (\ell, \ell') \in E)$
(8)          for each $(B' \in \mathcal{P}_{old}, B \in \mathcal{P}$ such that $\emptyset \neq B \cap Pre_e(B') \neq B)$
(9)              $B_1 \leftarrow B \cap Pre_e(B'); B_2 \leftarrow B \setminus Pre_e(B')$;
(10)             $\mathcal{P} \leftarrow (\mathcal{P} \setminus \{B\}) \cup \{B_1, B_2\}$;
(11)     while $(\exists B, B' \in \mathcal{P}$ such that $\emptyset \neq B \cap Pre_\tau(B') \neq B)$
(12)         $B_1 \leftarrow B \cap Pre_\tau(B'); B_2 \leftarrow B \setminus Pre_\tau(B')$
(13)         $\mathcal{P} \leftarrow (\mathcal{P} \setminus \{B\}) \cup \{B_1, B_2\}$;
(14) return $\mathcal{P}$
```

Fig. 1. Partition refinement algorithm for n-DBB equivalence.

Corollary 2. *Let H be a fully o-minimal$(\mathsf{OF}_{\mathsf{exp}}(\mathbb{R}))$ hybrid automaton in which:*

- *for each $\ell \in L$, the vector field is of the form $F(\ell, \mathbf{x}) = A\mathbf{x}$, where:*
  1. *$A \in \mathbb{Q} \times \mathbb{Q}$ is nilpotent or*
  2. *$A \in \mathbb{Q} \times \mathbb{Q}$ is diagonalizable with rational eigenvalues or*
  3. *$A \in \mathbb{Q} \times \mathbb{Q}$ has purely imaginary eigenvalues of the form $\mathrm{i}r$, $r \in \mathbb{Q}$, with diagonal real Jordan form.*
- *for each $\ell \in L$ and $(\ell, \ell') \in E$, the sets $Inv(\ell)$, $G(\ell)$, $Init(\ell)$, and the reset function $R(\ell, \ell') : X \mapsto X$ are definable inside $\mathsf{OF}(\mathbb{R})$.*

Then, for all $n \in \mathbb{N}$, $\equiv_n$ is finite and computable on H.

4 3-Valued CTL Semantics over DBB Abstractions

In this section we introduce a 3-valued semantics for the logic CTL on n-DBB abstractions. Such a three valued semantics exploits, besides the inductive definition of DBB abstractions, the simulation preorder relating successive DBB abstractions in the succession $\langle H_{/\equiv_n} \rangle_{n \in \mathbb{N}}$. The latter allows us to use unbounded runs in the evaluation of (not purely existential) CTL properties. In other words, each CTL\ECTL formula is not constrained to be evaluated by looking exclusively at paths in $H_{/\equiv_n}$ abstracting bounded runs of H, to obtain a value in $\{\mathsf{tt}, \mathsf{ff}, \perp\} \setminus \{\perp\}$. This key point needs to be emphasized, since it endows our framework with the abilities to handle both refutations as well as *proof* of safety or liveness properties over a hybrid automaton H. In fact, such properties intrinsically model some conditions that need to be maintained along the whole evolution of any run in H.

Definition 13. *Let H be a hybrid automaton having state space Q, let* AP *be a finite set of atomic propositions, and let $\mathcal{P}$ be the partition on Q induced by the labelling* *function $\ell_{\mathsf{AP}} : Q \mapsto 2^{|\mathsf{AP}|}$. Consider the n-DBB abstraction of H with respect to $\mathcal{P}$, $H_{/\equiv_n}$. Given the node $[s]_{\equiv_n}$ in $H_{/\equiv_n}$, the value $[[s]_{\equiv_n} \models_3 \phi] \in \{\mathsf{tt}, \mathsf{ff}, \bot\}$ is inductively defined on n as follows:*

– *If $\phi = \mathsf{p}$, then $[[s]_{\equiv_n} \models_3 \phi]$ is defined as:*

$$\begin{cases} \mathsf{tt} & \textit{iff} \quad \mathsf{p} \in \ell_{\mathsf{AP}}([s]_{\equiv_n})) \\ \mathsf{ff} & \textit{iff} \quad \mathsf{p} \notin \ell_{\mathsf{AP}}([s]_{\equiv_n})) \end{cases}$$

– *If $\phi = \neg\phi_1$, then $[[s]_{\equiv_n} \models_3 \phi]$ is defined as:*

$$\begin{cases} \mathsf{tt} & \textit{iff} \quad [[s]_{\equiv_n} \models_3 \phi_1] = \mathsf{ff}; \\ \mathsf{ff} & \textit{iff} \quad [[s]_{\equiv_n} \models_3 \phi_1] = \mathsf{tt}; \\ \bot & \textit{otherwise.} \end{cases}$$

– *If $\phi = \phi_1 \wedge \phi_2$, then $[[s]_{\equiv_n} \models_3 \phi]$ is defined as:*

$$\begin{cases} \mathsf{tt} & \textit{iff} \quad [[s]_{\equiv_n} \models_3 \phi_1] = \mathsf{tt} \wedge [[s]_{\equiv_n} \models_3 \phi_2] = \mathsf{tt}; \\ \mathsf{ff} & \textit{iff} \quad [[s]_{\equiv_n} \models_3 \phi_1] = \mathsf{ff} \vee [[s]_{\equiv_n} \models_3 \phi_2] = \mathsf{ff}; \\ \bot & \textit{otherwise} \end{cases}$$

– *If $\phi = \mathsf{E}\phi_1\mathsf{U}\phi_2$, then $[[s]_{\equiv_n} \models_3 \phi]$ is defined as:*

$$\begin{cases} \mathsf{tt} & \textit{iff there exists a path } \langle [s_i]_{\equiv_n} \rangle_{0 \le i \le k} \textit{ such that:} \\ & (1)\ \forall\, i < k\ ([s_i]_{\equiv_n} \xrightarrow{\delta} [s_{i+1}]_{\equiv_n}\ \wedge\ [[s_i]_{\equiv_n} \models_3 \phi_1 \vee \phi_2] = \mathsf{tt}) \\ & (2)\ [[s_k]_{\equiv_n} \models_3 \phi_2] = \mathsf{tt} \vee ([s_k]_{\equiv_n} \xrightarrow{e} [s_{k+1}]_{\equiv_n} \wedge [[s_{k+1}]_{\equiv_{n-1}} \models_3 \phi] = \mathsf{tt}) \\ \mathsf{ff} & \textit{iff for each path } \langle [s_i]_{\equiv_n} \rangle_{i \in \mathbb{N}} \textit{ it holds:} \\ & \forall\, i \in \mathbb{N}\ ([[s_i]_{\equiv_n} \models_3 \phi_2] = \mathsf{tt} \rightarrow (\exists\, j < i([[s_j]_{\equiv_n} \models_3 \neg\phi_1 \wedge \neg\phi_2] = \mathsf{tt}))) \\ \bot & otherwise \end{cases}$$

– *If $\phi = \mathsf{A}\phi_1\mathsf{U}\phi_2$, then $[[s]_{\equiv_n} \models_3 \phi]$ is defined as:*

$$\begin{cases} \mathsf{tt} & \textit{iff for each path } \langle [s_i]_{\equiv_n} \rangle_{i \in \mathbb{N}} \textit{ there exists an index } k \textit{ such that:} \\ & (1)\ \forall\, i < k\ ([[s_i]_{\equiv_n} \models_3 \phi_1 \vee \phi_2] = \mathsf{tt}) \\ & (2)\ [[s_k]_{\equiv_n} \models_3 \phi_2] = \mathsf{tt} \\ \mathsf{ff} & \textit{iff there exists a path } \langle [s_i]_{\equiv_n} \rangle_{i \le k} \textit{ such that:} \\ & (1)\ \forall\, i < k\ ([s_i]_{\equiv_n} \xrightarrow{\delta} [s_{i+1}]_{\equiv_n} \wedge [[s_i]_{\equiv_n} \models_3 \neg\phi_1 \vee \neg\phi_2] = \mathsf{tt}) \\ & (2)\ [[s_k]_{\equiv_n} \models_3 \neg\phi_1] = \mathsf{tt} \vee ([s_k]_{\equiv_n} \xrightarrow{e} [s_{k+1}]_{\equiv_n} \wedge [[s_{k+1}]_{\equiv_{n-1}} \models_3 \phi] = \mathsf{ff}) \\ \bot & otherwise \end{cases}$$

– *If $\phi = \mathsf{E}\phi_1\mathsf{R}\phi_2$, then $[[s]_{\equiv_n} \models_3 \phi] = [[s]_{\equiv_n} \models_3 \neg(\mathsf{A}\neg\phi_1\mathsf{U}\neg\phi_2)]$.*
– *If $\phi = \mathsf{A}\phi_1\mathsf{R}\phi_2$, then $[[s]_{\equiv_n} \models_3 \phi] = [[s]_{\equiv_n} \models_3 \neg(\mathsf{E}\neg\phi_1\mathsf{U}\neg\phi_2)]$.*

Finally, $H_{/\equiv_n} \models_3 \phi$ is defined as:

$$\begin{cases} \mathsf{tt} & \textit{iff} \quad \forall\, [s]_{\equiv_n} \in Q^0_{/\equiv_n} ([[s]_{\equiv_n} \models_3 \phi_1] = \mathsf{tt}); \\ \mathsf{ff} & \textit{iff} \quad \exists\, [s]_{\equiv_n} \in Q^0_{/\equiv_n} ([[s]_{\equiv_n} \models_3 \phi_1] = \mathsf{ff}); \\ \bot & \textit{otherwise} \end{cases}$$

Theorem 2 (Preservation). *Let $H_{/\equiv_n}$ be the n-DBB abstraction for a hybrid automaton H, and let ϕ be a* CTL *formula. If $[H_{/\equiv_n} \models_3 \phi] = \mathsf{tt}$, then $H \models \phi$; If $[H_{/\equiv_n} \models_3 \phi] = \mathsf{ff}$, then $\neg(H \models \phi)$.*

Theorem 3 (Monotonicity). *Let $\langle H_{/\equiv_n} \rangle_{n \in \mathbb{N}}$ be the succession of DBB abstractions for a hybrid automaton H. For any* CTL *formula ϕ, for any $n \in \mathbb{N}$ it holds:*

$$([H_{/\equiv_n} \models_3 \phi] = b \wedge b \in \{\mathsf{tt}, \mathsf{ff}\}) \rightarrow \forall\, m > n([H_{/\equiv_m} \models_3 \phi] = b)$$

5 A Linear Algorithm for 3-Valued CTL Model Checking on Discrete Bounded Bisimulation Abstractions

In this Section we define an efficient algorithm for the three valued CTL model checking on bounded bisimulation abstractions, assuming the latter to be finite. Classical CTL model checking over Kripke structures is known to be linear in the size of the structure and in the length of the formula; analogously the complexity of our three valued model checking procedure is linear in the size of the abstraction and in the length of the formula.

5.1 The case of 3-Valued ECTL∪ACTL Model Checking

We start solving a simpler problem: Namely, the definition of a procedure for the evaluation of $[H_{/\equiv_n} \models_3 \phi]$, where ϕ is either a universal or an existential formula of CTL.
Let ϕ be an ACTL formula, and let α be a node in $H_{/\equiv_n}$. According to Definition 13, $[\alpha \models_3 \phi] = \mathsf{tt}$ iff $\alpha \models \phi$ (with respect to the classical 2-valued semantics for CTL on the finite transition system $H_{/\equiv_n}$). Hence, it is sufficient to use a classical (2-valued) CTL model checking algorithm on $H_{/\equiv_n}$ to detect those nodes in $H_{/\equiv_n}$ for which $[\alpha \models_3 \phi] = \mathsf{tt}$ in time $\mathcal{O}(|H_{/\equiv_n}| * |\phi|)$.
The problem of collecting the nodes α in $H_{/\equiv_n}$ for which $[\alpha \models_3 \phi] = \mathsf{ff}$ reduces to the problem of (1) rewriting $\neg\phi$ as a formula γ in ECTL (2) determining the set of states α in $H_{/\equiv_n}$ for which $[\alpha \models_3 \gamma] = \mathsf{tt}$.

Summarizing the above observations, we can derive an overall $\mathcal{O}(|H_{/\equiv_n}| * |\phi|)$ algorithm for computing $H_{/\equiv_n} \models_3 \phi$, if we prove that it is possible to recognize all those nodes α for which $[\alpha \models_3 \gamma] = \mathsf{tt}$, $\gamma \in \mathsf{ECTL}$, in time $\mathcal{O}(|H_{/\equiv_n}| * |\gamma|)$.

Let γ be an ECTL formula: we solve the above subproblem by employing an efficient ($\mathcal{O}(|H_{/\equiv_n}| * |\gamma|)$) strategy to distribute on the nodes in $H_{/\equiv_n}$ the set of labels $\langle \psi, \mathsf{tt}, m \rangle$, where:

- ψ is a subformula of γ and $m \leq n$.
- $\alpha = [s]_n$ receives the label $\langle \psi, \mathsf{tt}, m \rangle$ if and only if

$$[[s]_m \models_3 \psi] = \mathsf{tt} \wedge \forall\, m' < m([[s]_{m'} \models_3 \psi] = \bot)$$

Our strategy uses a structural induction on γ. The cases in which γ is either a propositional letter or a boolean composition of subformulæ are easily dealt with (cfr. lines (1.1)–(2.4) of the procedure PROCESSE in Figure 2).

```
ALGO1($H_{/\equiv_n}, n, \phi$)
Input: $n \in \mathbb{N}$, the discrete $n$-bounded bisimulation abstraction $H_{/\equiv_n}$ for a hybrid automaton $H$,
       a formula $\phi \in \mathsf{ECTL} \cup \mathsf{ACTL}$
Output: $[H_{/\equiv_n} \models_3 \phi] \in \{\mathsf{tt}, \mathsf{ff}, \bot\}$

(1) Let $\gamma \equiv \neg\phi$, where $\gamma$ is in negation normal form
(2) if $(\phi \in \mathsf{ACTL})$ then PROCESSA($H_{/\equiv_n}, n, \phi$); PROCESSE($H_{/\equiv_n}, n, \gamma$)
(3)                   else PROCESSE($H_{/\equiv_n}, n, \phi$); PROCESSA($H_{/\equiv_n}, n, \gamma$)
(4) If $(\forall \alpha \in Init(H_{/\equiv_n})\ (\langle\phi, \mathsf{tt}, -\rangle \in \ell(\alpha)))$ then return tt
(5) If $(\exists \alpha \in Init(H_{/\equiv_n})\ (\langle\phi, \mathsf{ff}, -\rangle \in \ell(\alpha)))$ then return ff else return $\bot$
```

```
PROCESSE($H_{/\equiv_n}, n, \phi$)
Input: $n \in \mathbb{N}$, the discrete $n$-bounded bisimulation abstraction $H_{/\equiv_n}$ for a hybrid automaton $H$,
       a formula $\phi \in \mathsf{ECTL}$
Output: $\forall\ [s]_n in H_{/\equiv_n}$, $[s]_n$ is labeled $\langle\phi, \mathsf{tt}, m\rangle$ iff $[[s]_m \models_3 \phi] = \mathsf{tt} \wedge \forall k < m\ [[s]_k \models_3 \phi] = \bot$

        case $\phi$ of
(1.1)   $\mathsf{p}$     : for each $\alpha \in Q_{/\equiv_n}$ such that $lab_{\mathsf{p}}(\alpha) = \mathsf{tt}$ do $\ell(\alpha) \leftarrow \ell(\alpha) \cup \langle\mathsf{p}, \mathsf{tt}, 0\rangle$
(1.2)   $\neg\mathsf{p}$    : for each $\alpha \in Q_{/\equiv_n}$ such that $lab_{\mathsf{p}}(\alpha) = \mathsf{ff}$ do $\ell(\alpha) \leftarrow \ell(\alpha) \cup \langle\neg\mathsf{p}, \mathsf{tt}, 0\rangle$

(2.1)   $\phi_1 \diamondsuit \phi_2$ : for each $\alpha \in Q_{/\equiv_n}$ do
(2.2) $\diamondsuit \in \{\vee\wedge\}$       if $[\langle\phi_1, \mathsf{tt}, m_1\rangle \in \ell(\alpha)] \diamondsuit [\langle\phi_2, \mathsf{tt}, m_2\rangle \in \ell(\alpha)]$ then
(2.4)                        $\ell(\alpha) \leftarrow \ell(\alpha) \cup \{\langle\phi, \mathsf{tt}, \max(m_1, m_2)\rangle\}$

(3.1)   $\mathsf{E}\phi_1\mathsf{U}\phi_2$ : / * Initialization * /
(3.2)          $N \leftarrow |H_{/\equiv_n}|; S_0 \leftarrow \emptyset; \ldots; S_n \leftarrow \emptyset$;  for each $(\alpha \in Q_{/\equiv_n})$ $color(\alpha) \leftarrow green$
(3.3)          Let $A_1[0 \ldots N], A_2[0 \ldots N]$ be two array of lists of nodes in $Q_{/\equiv_n}$, such that $\alpha$
               belongs to the list $A_j[i]$ iff $\langle\phi_j, \mathsf{tt}, i\rangle \in \ell(\alpha)$
             / * For $i = 0 \ldots n$, build the set $S_i$ of nodes $\alpha = [s]_{\equiv_n}$ such that $[[s]_{\equiv_i} \models_3 \phi] = \mathsf{tt}$ and * /
             / * $\forall j < i[[s]_{\equiv_j} \models_3 \phi] = \bot$ * /
(3.4)          for each (node $\alpha$ in the list $A_2[0]$) do $S_0 \leftarrow S_0 \cup \{\alpha\}$
(3.5)          Use a breadth-first search like algorithm to discover and color red each node $\alpha$ such that
               $\alpha \overset{0}{\rightsquigarrow} S_0 \wedge \langle\phi_1, \mathsf{tt}, 0\rangle$ and augment $S_0$ with the nodes discovered.
(3.6)          for each $(\alpha \in pre(S_0))$ if $(color(\alpha) = green)$ then $color(\alpha) \leftarrow yellow$
(3.7)          for $(i = 1 \ldots n)$ do
(3.8)              for each (not red $\alpha$ in the list $A_2[0]$, yellow $\beta$ in the list $A_1[i]$) do $S_i \leftarrow S_i \cup \alpha$;
(3.9)              Use a breadth-first search like algorithm to discover and color red each $\alpha$ such that
                   $\alpha \overset{1}{\rightsquigarrow} (S_i \cup S_{i-1}) \wedge \langle\phi_1, \mathsf{tt}, -\rangle \in \ell(\alpha)$. Augment $S_i$ with the nodes discovered.
(3.10)             for each $(\alpha \in pre(S_i))$ if $(color(\alpha) = green)$ then $color(\alpha) \leftarrow yellow$
             / * Assign the labels * /
(3.11)          for $(i = 1 \ldots n)$ do
(3.12)              Assign the label $\langle\phi, \mathsf{tt}, i\rangle$ to each node in $S_i$

(4.1)   $\mathsf{E}\phi_1\mathsf{R}\phi_2$ : Rewrite $\phi$ as $\mathsf{E}\phi_2\mathsf{U}\phi_1$ and use the rules for case (2)
```

```
PROCESSA($H_{/\equiv_n}, n, \phi$)
Input: $n \in \mathbb{N}$, the discrete $n$-bounded bisimulation abstraction $H_{/\equiv_n}$ for a hybrid automaton $H$,
       a formula $\phi \in \mathsf{ACTL}$
Output: $\forall\ [s]_n in H_{/\equiv_n}$, $[s]_n$ is labeled $\langle\phi, \mathsf{tt}, \bot\rangle$ iff $[[s]_m \models_3 \phi] = \mathsf{tt}$

/ * Use a 2-valued model checking procedure to process the formula $\phi$ on $H_{/\equiv_n}$ * /
(1.1)   for each $(\alpha \in H_{/\equiv_n})$ do
(1.3)       if $\alpha \models \mathsf{A}\phi_1\mathsf{U}\phi_2$ then $\ell(\alpha) \leftarrow \ell(\alpha) \cup \{\langle\phi, \mathsf{tt}, \bot\rangle\}$
```

Fig. 2. The linear algorithm for $\mathsf{ACTL} \cup \mathsf{ECTL}$ 3-valued Model Checking on $H_{/\equiv_n}$.

The case for which $\gamma = \mathsf{E}\gamma_1\mathsf{U}\gamma_2$ requires instead more attention. As illustrated in Figure 2 (cfr. lines (3.1)–(3.12) of the procedure PROCESSE), given $N = |H_{/\equiv_n}|$, we first build the two vectors $A_1[1,\dots,N], A_2[1,\dots,N]$ of lists of nodes in $H_{/\equiv_n}$, where $\alpha \in A_j[i]$ iff the label $\langle\gamma_j, \mathsf{tt}, i\rangle$ has been inductively associated to α. Note that, $A_1[1\dots N]$ (resp. $A_2[1\dots N]$) requires space $\mathcal{O}(|H_{/\equiv_n}|)$, since the lists in each slot of the array A are disjoint. Then, by induction on $i = 0\dots n$, we build the sets $S_0,\dots,S_n$, where S_i contains all the nodes of $H_{/\equiv_n}$ that need to be labeled with $\langle\gamma, \mathsf{tt}, i\rangle$. More precisely, such a building process is supported by a coloring marking in which a node is red if it has been already assigned to some $S_{j<i}$; A node is yellow if it admits a transition to a red node; A node is green otherwise. Given the above coloring we let S_i to contain:

- Each not red node in the list $A_2[i]$.
- Each yellow node in the list $A_1[i]$.
- Each node $\beta \in H_{/\equiv_n}$ admitting a path p to S_{i-1} such that:
  1. p contains at most one edge labeled e
  2. the label $\langle\gamma_1 \vee \gamma_2, \mathsf{tt}, i\rangle$ is associated to each node of p.

Finally, if $\gamma = \mathsf{E}\gamma_1\mathsf{R}\gamma_2$, we can reduce to process the formula $\gamma' = \mathsf{E}\gamma_2\mathsf{U}\gamma_1$ according to the three-valued CTL semantics in Definition 13. The pseudocode of the overall algorithm above sketched for determining $[H_{/\equiv_n} \models_3 \phi]$, $\phi \in \mathsf{ECTL} \cup \mathsf{ACTL}$, is reported in Figure 2. Given the formula $\phi \in \mathsf{ECTL} \cup \mathsf{ACTL}$ in negation normal form, Theorem 4 states that our procedure ALGO1$(H_{/\equiv_n}, \phi)$ computes the value $[H_{/\equiv_n} \models_3 \phi]$ in time $\mathcal{O}(|H_{/\equiv_n}| * |\phi|)$ and space $\mathcal{O}(|H_{/\equiv_n}|)$.

Theorem 4. *The algorithm* ALGO1$(H_{/\equiv_n}, n, \phi)$ *computes* $[H_{/\equiv_n} \models_3 \phi] \in \{\mathsf{tt}, \mathsf{ff}\bot\}$ *in time* $\mathcal{O}(|H_{/\equiv_n}| * |\phi|)$ *and space* $\mathcal{O}(|H_{/\equiv_n}|)$.

5.2 3-Valued CTL Model Checking

In this Subsection we extend the techniques outlined in Subsection 5.1 to design a $\mathcal{O}(|H_{/\equiv_n}| * |\phi|)$ algorithm for computing $H_{/\equiv_n} \models_3 \phi$, where ϕ is a general CTL formula. We start with some preliminary observations to illustrate the bottlenecks related to the above extension. Let $\phi_1 = \mathsf{EpUq}$, let $\phi_2 = \mathsf{ApUq}$, and consider the two formulæ belonging to $\mathsf{CTL} \setminus (\mathsf{ECTL} \cup \mathsf{ACTL})$: $\psi_1 = \mathsf{ArU}\phi_1$ and $\psi_2 = \mathsf{ErU}\phi_2$.
Since ϕ_1, ϕ_2, and r belong to $\mathsf{ECTL} \cup \mathsf{ACTL}$, we can assume to have determined with ALGO1 the set of nodes α in $H_{/\equiv_n}$ for which $[\alpha \models_3 \varsigma] = \mathsf{tt}$, $\varsigma \in \{\phi_1, \phi_2, \mathsf{r}\}$. Then, the task of processing the formula $\psi_1 = \mathsf{ArU}\phi_1$ according to Definition 13 does not present any problem. In fact, determining each node β for which $[\beta \models_3 \phi] = \mathsf{tt}$ boils down to applying a *classical model checking* subprocedure targeting the operator AU (where the precomputed labelling is interpreted in a 2-valued fashion and $\bot$ corresponds to ff, according to a 'pessimistic view').
Instead, we face the following problem in processing the formula $\psi_2 = \mathsf{ErU}\phi_2$ above, according to the 3-valued semantics in Definition 13. Namely, for each $\alpha = [s]_{\equiv_n}$ such that $[\alpha \models_3 \phi_2] = \mathsf{tt}$, we *need to know the least* $m \leq n$ *for which* $[[s]_{\equiv_m} \models_3 \phi_2] = \mathsf{tt}$. If ϕ_2 were a formula having an *existential* main path quantifier, say $\phi_2 = \exists\phi_3\mathsf{U}\phi_4$, then such minimum indexes would have been dependent on the number of discrete edges

```
ALGO2($H_{/\equiv_n}, T_n, n, \phi, \gamma$)
Input: $n \in \mathbb{N}$, the discrete $n$-bounded bisimulation abstraction $H_{/\equiv_n}$, the compact partition tree $T_n$
        associated to $\equiv_n$, $\phi, \gamma \equiv \neg\phi \in \mathsf{CTL}$ , where $\phi$ and $\gamma$ are in negation normal form
Output: $[H_{/\equiv_n} \models_3 \phi] \in \{\mathsf{tt}, \mathsf{ff} \bot\}$

(1.1)  if ($\phi \in \{\mathsf{p}, \neg\mathsf{p}\}$ for some $\mathsf{p} \in \mathsf{AP}$) then
(1.2)     for each $\alpha \in Q_{/\equiv_n}$ do
(1.3)        if $\ell_{\mathsf{p}}(\alpha) = \mathsf{tt}$ then $\ell(\alpha) \leftarrow \ell(\alpha) \cup \{\langle \mathsf{p}, \mathsf{tt}, 0\rangle\}$ else $\ell(\alpha) \leftarrow \ell(\alpha) \cup \{\langle \neg\mathsf{p}, \mathsf{tt}, 0\rangle\}$

(2.1)  if ($\phi = \phi_1 \diamondsuit \phi_2, \diamondsuit \in \{\vee \wedge\}$) then
(2.2)     for each $\alpha \in Q_{/\equiv_n}$ do
(2.3)        if $[\langle \phi_1, \mathsf{tt}, m_1\rangle \in \ell(\alpha)] \diamondsuit [\langle \phi_2, \mathsf{tt}, m_2\rangle \in \ell(\alpha)]$ then
(2.4)                          $\ell(\alpha) \leftarrow \ell(\alpha) \cup \{\langle \phi, \mathsf{tt}, \max(m_1, m_2)\rangle\}$

(3.1)  if ($\phi = \mathsf{E}\phi_1 \mathsf{U} \phi_2 \wedge \gamma = \mathsf{A}\gamma_1 \mathsf{R} \gamma_2$) then
(3.2)     ALGO2($H_{/\equiv_n}, T_n, n, \phi_1, \gamma_1$); ALGO2($H_{/\equiv_n}, T_n, n, \phi_2, \gamma_2$)
(3.3)     PROCESSEU($H_{/\equiv_n}, T_n, n, \phi_1, \phi_2$); PROCESSAR($H_{/\equiv_n}, T_n, n, \gamma_1, \gamma_2$)

(4.1)  if ($\phi = \mathsf{A}\phi_1 \mathsf{U} \phi_2 \wedge \gamma = \mathsf{E}\gamma_1 \mathsf{R} \gamma_2$) then
(4.2)     ALGO2($H_{/\equiv_n}, T_n, n, \phi_1, \gamma_1$); ALGO2($H_{/\equiv_n}, T_n, n, \phi_2, \gamma_2$)
(4.3)     PROCESSAU($H_{/\equiv_n}, T_n, n, \phi_1, \phi_2$); PROCESSER($H_{/\equiv_n}, T_n, n, \gamma_1, \gamma_2$)

(5.1)  if ($\phi = \mathsf{E}\phi_1 \mathsf{R} \phi_2 \wedge \gamma = \mathsf{A}\gamma_1 \mathsf{E} \gamma_2$) then
(5.2)     ALGO2($H_{/\equiv_n}, T_n, n, \phi_1, \gamma_1$); ALGO2($H_{/\equiv_n}, T_n, n, \phi_2, \gamma_2$)
(5.3)     PROCESSER($H_{/\equiv_n}, T_n, n, \phi_1, \phi_2$); PROCESSAU($H_{/\equiv_n}, T_n, n, \gamma_1, \gamma_2$)

(6.1)  if ($\phi = \mathsf{A}\phi_1 \mathsf{R} \phi_2 \wedge \gamma = \mathsf{E}\gamma_1 \mathsf{U} \gamma_2$) then
(6.2)     ALGO2($H_{/\equiv_n}, T_n, n, \phi_1, \gamma_1$); ALGO2($H_{/\equiv_n}, T_n, n, \phi_2, \gamma_2$)
(6.3)     PROCESSAR($H_{/\equiv_n}, T_n, n, \phi_1, \phi_2$); PROCESSEU($H_{/\equiv_n}, T_n, n, \gamma_1, \gamma_2$)

(7.1)  If ($\forall \alpha \in Init(Q_{/\equiv_n})$ ($\langle \phi, \mathsf{tt}, -\rangle \in \ell(\alpha)$)) then return tt
(7.2)  If ($\exists \alpha \in Init(Q_{/\equiv_n})$ ($\langle \gamma, \mathsf{tt}, -\rangle \in \ell(\alpha)$)) then return ff else return $\bot$
```

Fig. 3. The linear CTL 3-valued Model Checking on $H_{/\equiv_n}$.

that one needs to traverse to evaluate ϕ_2 (assuming a previous labeling relative to ϕ_3 and ϕ_4[2]). However, formulæ having a main universal path quantifier are evaluated by considering any path in $H_{/\equiv_n}$, regardless of the number of discrete edges traversed. Our solution to recover the required minimum indexes for formulæ having a main universal path quantifier, is that of disposing of a data structure that we call *compact partition tree*, requiring space $\mathcal{O}(|H_{/\equiv_n}|)$. The formal description of a compact partition tree is given in Definition 14.

Definition 14 (Compact Partition Tree associated to $H_{/\equiv_n}$). *Let H be a hybrid automaton. The compact partition tree T_n associated to $H_{/\equiv_n}$ is inductively defined as:*

- *T_0 is a tree of depth one in which each leaf f is associated to a distinct node $\alpha \in H_{/\equiv_n}$ and is labeled with the triple $\ell(f) = \langle 0, 0, \alpha\rangle$.*

[2] In fact, the indices m are determined 'on the fly' in our procedure PROCESSE for ECTL.

– *T_{n+1} is obtained from T_n processing each leaf f in T_n according to the following procedure:*
  - *If f is labeled with the triple $\langle m, n-1, \alpha\rangle$, and α is a class of $H_{/\equiv_n}$, then let $\ell(f) = \langle m, n, \alpha\rangle$.*
  - *Otherwise, substitute the label of f with the pair $\ell(f) = \langle m, n-1\rangle$. For each $\alpha_i \subseteq \alpha, \alpha_i \in H_{/\equiv_n}$, create a new successor of f, f_i, and associate to f_i the label $\langle n, n, \alpha_i\rangle$.*

Lemma 4. *Let H be a hybrid automaton. The space complexity of the compact partition tree associated to $H_{/\equiv_n}$ is $\mathcal{O}(|H_{/\equiv_n}|)$.*

Compact partition trees can be easily computed along discrete bounded bisimulations. Let $\phi \in \mathsf{CTL}$, and assume that the main path quantifier in ϕ is universal. Suppose having determined the set of nodes $S = \{[s]_{\equiv_n} \in H_{/\equiv_n} \,|\, [[s]_{\equiv_n} \models_3 \phi] = \mathsf{tt}\}$. Then, the compact partition tree T_n can be used as follows to associate to each node $\alpha = [[s]_{\equiv_n} \in S$ (in time $\mathcal{O}(|H_{/\equiv_n}|)$) the required label $\langle \phi, \mathsf{tt}, m\rangle$, where m is the minimum index such that $[[s]_{\equiv_m} \models_3 \phi] = \mathsf{tt}$. First we use a depth first search-like algorithm to discover each node k of T_n satisfying the two conditions listed below:

1. any leaf of T_n which is a descendant of the node k is associated to a class $\alpha \in H_{/\equiv_n}$ for which $\alpha \models_3 \phi] = \mathsf{tt}$
2. k is a node of minimal depth having property 1.

If $[m, m']$ is the interval labeling k, then each class α associated to a leaf-descendant of k needs to be labeled by the triple $\langle \phi, \mathsf{tt}, m\rangle$.

The ideas sketched above lead to the final algorithm[3] reported in Figure 3, which computes $[H_{/\equiv_n} \models_3 \phi]$, where $\phi \in \mathsf{CTL}$, in time $\mathcal{O}(|H_{/\equiv_n}| * |\phi|)$ and space $\mathcal{O}(|H_{/\equiv_n}|)$.

Theorem 5. *The algorithm* ALGO2$(H_{/\equiv_n}, T_n, n, \phi, \gamma)$ *computes* $[H_{/\equiv_n} \models_3 \phi] \in \{\mathsf{tt}, \mathsf{ff} \bot\}$ *in time* $\mathcal{O}(|H_{/\equiv_n}| * |\phi|)$ *and space* $\mathcal{O}(|H_{/\equiv_n}|)$.

6 Conclusions

This paper has proposed a novel framework to extend the power of automated reasoning over hybrid automata, even when those automata are undecidable in classical CTL. In this framework, it is possible to *both prove and disprove* reactive system properties expressed by means of CTL logic on (undecidable) hybrid automata. To the best of authors' knowledge, this research is novel and points to a fresh approach, as no other currently available symbolic technique analyzing undecidable hybrid automata can cope with both proofs and refutations of such general reactive systems properties as safety or liveness. The key ingredients of this innovative framework consist of proof systems, built upon a succession of abstractions and a corresponding three valued semantics for the CTL logic, which in turn allows for the monotonic preservation of true and false properties along these successive abstractions. This paper further proves that the new three valued model checking problem is not only decidable on our DBB abstractions, but is as efficient as the classical model checking of discrete Kripke structures, as it is linear in the length of the formula and in the size of the abstraction.

[3] The subprocedures used in the main algorithm ALGO2 are illustrated in the Appendix.

References

1. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88:971–984, 2000.
3. T. Brihaye, C. Michaux, C. Rivie, and C. Troestler. On o-minimal hybrid systems. In *Proc. of the 7th Int. Workshop on Hybrid Systems*, pages 219–233, 2004.
4. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
5. M. Fitting. Kleene's three valued logics and their children. *Fundam. Inf.*, 20(1-3):113–131, 1994.
6. M. Fränzle. What will be eventually true of polynomial hybrid automata? In *Proc. of Int. Symp. on Theoretical Aspects of Computer Software*, volume 2215 of *LNCS*, pages 340–359. Springer, 2001.
7. R. Gentilini, B. Mishra, and K. Schneider. Successive abstractions of hybrid automata for monotonic CTL model checking. Technical report, Kaiserslautern University, 2007.
8. R. Ghosh, A. Tiwari, and C. Tomlin. Automated symbolic reachability analysis with application to delta-notch signaling automata. In *Hybrid Systems: Computation and Control HSCC*, volume 2623 of *LNCS*, pages 233–248. Springer, 2003.
9. T. Henzinger, P. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Proc. of the 27th Symposium on Theory of Computing*, pages 373–382. ACM Press, 1995.
10. T. A. Henzinger. The theory of hybrid automata. In *Proc. of the 11th IEEE Symp. on Logic in Computer Science*, pages 278–292. IEEE Computer Society, 1996.
11. P. C. Kannellakis and S. A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, 1990.
12. S. C. Kleene. *Introduction to Metamathematics*. Wolters-Noordhoff, Groningen, 1971.
13. G. Lafferriere, G. Pappas, and S. Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13:1–21, 2000.
14. G. Lafferriere, J. Pappas, and S. Yovine. A new class of decidable hybrid systems. In *Proc. of the 2nd Int. Workshop on Hybrid Systems*, pages 137–151. Springer, 1999.
15. J. Miller. Decidability and complexity results for timed automata and semi-linear hybrid automata. In *Proc. of the 3th Int. Workshop on Hybrid Systems*, pages 296–309, 2000.
16. C. Piazza, M. Antoniotti, V. Mysore, A. Policriti, F. Winkler, and B. Mishra. Algorithmic algebraic model checking i: Challenges from systems biology. In *Proc. of the 17th Int. Conf. on Computer Aided Verification*, volume 3576 of *LNCS*, pages 5–19. Springer, 2005.
17. S. Ratschan and Z. She. Safety verification of hybrid systems by constraint propagation based abstraction refinement. In *Proc. of the 8th Int. Workshop on Hybrid Systems: Computation and Control*, volume 3414 of *LNCS*, pages 573–589. Springer, 2005.
18. K. Schneider. *Verification of Reactive Systems*. Springer Verlag, 2004.
19. A. Tiwari and G. Khanna. Series of abstractions for hybrid automata. In *Proc. of the 5th International Workshop on Hybrid Systems*, pages 465–478. Springer-Verlag, 2002.
20. L. van den Dries. O-minimal structures. In W. Hodges, editor, *Logic: From Foundations to Applications*, pages 99–108. Clarendon Press, 1996.
21. L. van den Dries. *Tame topology and o-minimal structures*, volume 248 of *London Math. Soc. Lecture Note Ser.* Cambridge University Press, 1998.
22. A. J. Wilkie. Schanuel conjecture and the decidability of the real exponential field. *Algebraic Model Theory*, pages 223–230, 1997.

7 Appendix

```
PROCESSAU($H_{/\equiv_n}, T_n, n, \phi_1, \phi_2$)
Input: For $\psi \in \{\phi_1, \phi_2\}$, each node $\alpha = [s]_n$ in $H_{/\equiv_n}$ is assumed to be labeled with $\langle \psi, \mathsf{tt}, m\rangle$
        if $[[s]_m \models_3 \psi] = \mathsf{tt} \wedge \forall k < m\ [[s]_k \models_3 \psi] = \perp$
Out: $\forall\ [s]_n in H_{/\equiv_n}$, $[s]_n$ is labeled $\langle \phi = \mathsf{A}\phi_1\mathsf{U}\phi_2, \mathsf{tt}, m\rangle$ iff $[[s]_m \models_3 \phi] = \mathsf{tt} \wedge \forall k < m\ [[s]_k \models_3 \phi] = \perp$

/ * Use a 2-valued model checking procedure targeting the AU operator to process the formula $\mathsf{A}\phi_1\mathsf{U}\phi_2$ * /
/ * assuming, for $\psi \in \{\phi_1, \phi_2\}$, that $\alpha \models \psi$ iff $\langle \psi, \mathsf{tt}, -\rangle \in \ell(\alpha)$ * /
(1.1)  for each ($\alpha \in H_{/\equiv_n}$) do
(1.3)      if $\alpha \models \mathsf{A}\phi_1\mathsf{U}\phi_2$ then $\ell(\alpha) \leftarrow \ell(\alpha) \cup \{\langle \mathsf{A}\phi_1\mathsf{U}\phi_2, \mathsf{tt}, -\rangle\}$
/ * Use a depth first search like algorithm on $T_n$ to determine the set of $T_n$ nodes $S$ * /
(1.4)  $S \leftarrow \{k \in T_n \mid$ any $\alpha \in H_{/\equiv_n}$ associated to a leaf descending from $k$ is labeled
           $\langle \mathsf{A}\phi_1\mathsf{U}\phi_2, \mathsf{tt}, -\rangle$, and $k$ is a node of minimal depth having the above property$\}$
(1.5)  for each ($\langle k \in S$, leaf $f$ descending from $k\rangle$) do
(1.6)    if ($\ell(f) = \langle -, -, \alpha\rangle \wedge \ell(k) = \langle m, -\rangle$)
(1.7)      then $\ell(\alpha) \leftarrow (\ell(\alpha) \setminus \{\langle \mathsf{A}\phi_1\mathsf{U}\phi_2, \mathsf{tt}, n\rangle\}) \cup \{\langle \mathsf{A}\phi_1\mathsf{U}\phi_2, \mathsf{tt}, m\rangle\}$
```

```
PROCESSEU($H_{/\equiv_n}, T_n, n, \phi_1, \phi_2$)
Input: For $\psi \in \{\phi_1, \phi_2\}$, each node $\alpha = [s]_n$ in $H_{/\equiv_n}$ is assumed to be labeled with $\langle \psi, \mathsf{tt}, m\rangle$
        if $[[s]_m \models_3 \psi] = \mathsf{tt} \wedge \forall k < m\ [[s]_k \models_3 \psi] = \perp$
Out: $\forall\ [s]_n in H_{/\equiv_n}$, $[s]_n$ is labeled $\langle \phi = \mathsf{E}\phi_1\mathsf{U}\phi_2, \mathsf{tt}, m\rangle$ iff $[[s]_m \models_3 \phi] = \mathsf{tt} \wedge \forall k < m\ [[s]_k \models_3 \phi] = \perp$

/ * Initialization * /
(1.1)  $N \leftarrow |H_{/\equiv_n}|; S_0 \leftarrow \emptyset; \ldots; S_n \leftarrow \emptyset;$  for each ($\alpha \in Q_{/\equiv_n}$) $color(\alpha) \leftarrow green$
(1.2)  Let $A_1[0 \ldots N]$, $A_2[0 \ldots N]$ be two array of lists of nodes in $Q_{/\equiv_n}$, such that $\alpha$ belongs to the list
       $A_j[i]$ iff $\langle \phi_j, \mathsf{tt}, i\rangle \in \ell(\alpha)$
/ * For $i = 0 \ldots n$, build the set $S_i$ of nodes $[s]_{\equiv_n}$ such that $[[s]_{\equiv_i} \models_3 \phi] = \mathsf{tt}$ and $\forall j < i[[s]_{\equiv_j} \models_3 \phi] = \perp$ * /
(1.3)  for each ($\alpha$ in the list $A_2[0]$) do $S_0 \leftarrow S_0 \cup \alpha$ endfor
(1.4)  Use a breadth-first search like algorithm to discover and mark each node $\alpha$ such that
       $\alpha \overset{0}{\rightsquigarrow} S_0 \wedge \langle \phi_1, \mathsf{tt}, 0\rangle \in \ell(\alpha)$ and augment $S_0$ with the nodes discovered.
(1.5)  for each ($\alpha \in pre(S_0)$) if ($color(\alpha) = green$) then $color(\alpha) \leftarrow yellow$
(1.6)  for ($i = 1 \ldots n$) do
(1.7)      for each (not red $\alpha$ in $A_2[i]$, yellow $\beta$ in $A_1[i]$) do $S_i \leftarrow S_i \cup \{\alpha, \beta\}$ endfor
(1.8)      Use a breadth-first search like algorithm to discover and color red each $\alpha$ such that
             $\alpha \overset{1}{\rightsquigarrow} (S_i \cup S_{i-1}) \wedge \langle \phi_1, \mathsf{tt}, i\rangle \in \ell(\alpha)$. Augment $S_i$ with the nodes discovered.
(1.9)      for each ($\alpha \in pre(S_i)$) if ($color(\alpha) = green$) then $color(\alpha) \leftarrow yellow$
/ * Assign the labels * /
(1.10)  for ($i = 1 \ldots n$) do  Assign the label $\langle \phi, \mathsf{tt}, i\rangle$ to each node in $S_i$ endfor
```

```
PROCESSAR($H_{/\equiv_n}, T_n, n, \phi_1, \phi_2$)
Input: For $\psi \in \{\phi_1, \phi_2\}$, each node $\alpha = [s]_n$ in $H_{/\equiv_n}$ is assumed to be labeled with $\langle \psi, \mathsf{tt}, m\rangle$
        if $[[s]_m \models_3 \psi] = \mathsf{tt} \wedge \forall k < m\ [[s]_k \models_3 \psi] = \perp$
Out: $\forall\ [s]_n in H_{/\equiv_n}$, $[s]_n$ is labeled $\langle \phi = \mathsf{A}\phi_1\mathsf{R}\phi_2, \mathsf{tt}, m\rangle$ iff $[[s]_m \models_3 \phi] = \mathsf{tt} \wedge \forall k < m\ [[s]_k \models_3 \phi] = \perp$

/ * Use a 2-valued model checking procedure targeting the AR operator to process the formula $\mathsf{A}\phi_1\mathsf{R}\phi_2$ * /
/ * assuming, for $\psi \in \{\phi_1, \phi_2\}$, that $\alpha \models \psi$ iff $\langle \psi, \mathsf{tt}, -\rangle \in \ell(\alpha)$ * /
(1.1)  for each ($\alpha \in H_{/\equiv_n}$) do
(1.3)      if $\alpha \models \mathsf{A}\phi_1\mathsf{R}\phi_2$ then $\ell(\alpha) \leftarrow \ell(\alpha) \cup \{\langle \mathsf{A}\phi_1\mathsf{R}\phi_2, \mathsf{tt}, n\rangle\}$
/ * Use a depth first search like algorithm on $T_n$ to determine the set of $T_n$ nodes $S$ * /
(1.4)  $S \leftarrow \{k \in T_n \mid$ any $\alpha \in H_{/\equiv_n}$ associated to a leaf descending from $k$ is labeled
           $\langle \mathsf{A}\phi_1\mathsf{R}\phi_2, \mathsf{tt}, -\rangle$, and $k$ is a node of minimal depth having the above property$\}$
(1.5)  for each ($\langle k \in S$, leaf $f$ descending from $k\rangle$) do
(1.6)    if ($\ell(f) = \langle -, -, \alpha\rangle \wedge \ell(k) = \langle m, -\rangle$)
(1.7)      then $\ell(\alpha) \leftarrow (\ell(\alpha) \setminus \{\langle \mathsf{A}\phi_1\mathsf{R}\phi_2, \mathsf{tt}, n\rangle\}) \cup \{\langle \mathsf{A}\phi_1\mathsf{R}\phi_2, \mathsf{tt}, m\rangle\}$
```

```
PROCESSER($H_{/\equiv_n}, T_n, n, \phi_1, \phi_2$)
Input: For $\psi \in \{\phi_1, \phi_2\}$, each node $\alpha = [s]_n$ in $H_{/\equiv_n}$ is assumed to be labeled with $\langle \psi, \mathsf{tt}, m\rangle$
        if $[[s]_m \models_3 \psi] = \mathsf{tt} \wedge \forall k < m\ [[s]_k \models_3 \psi] = \perp$
Out: $\forall\ [s]_n in H_{/\equiv_n}$, $[s]_n$ is labeled $\langle \phi = \mathsf{E}\phi_1\mathsf{R}\phi_2, \mathsf{tt}, m\rangle$ iff $[[s]_m \models_3 \phi] = \mathsf{tt} \wedge \forall k < m\ [[s]_k \models_3 \phi] = \perp$

(1.1)  PROCESSEU($H_{/\equiv_n}, T_n, n, \phi_2, \phi_1$)
(1.2)  for each ($\alpha \in H_{/\equiv_n}$) do
(1.3)      if $\langle \mathsf{E}\phi_2\mathsf{U}\phi_1, \mathsf{tt}, m\rangle \in \ell(\alpha)$ then $\ell(\alpha) \leftarrow \ell(\alpha) \cup \{\langle \mathsf{E}\phi_1\mathsf{R}\phi_2, \mathsf{tt}, m\rangle\}$
```

Fig. 4. The four subrocedures used within the algorithm ALGO2, in Figure 3.