Series of Abstractions of Hybrid Automata for Monotonic CTL Model Checking

R. Gentilini\textsuperscript{2}, K. Schneider\textsuperscript{2} and B. Mishra\textsuperscript{1,3}

\textsuperscript{1} Courant Institute, New York University, New York, NY, U.S.A.
\textsuperscript{2} University of Kaiserslautern, Department of Computer Science, Germany
\textsuperscript{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 on undecidable hybrid automata, force to choose between the refinement of either an overapproximation or an underapproximation of the set of reachable states. When the analysis of branching 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.

In this paper, we deal with the problem of defining series of abstractions of hybrid automata that (1) allow the detection/refinement of both overapproximated and underapproximated reachable sets (2) preserve fully branching temporal properties (interpreted on a dense time domain). Moreover, we ask for our series of abstractions, to satisfy a monotonicity property with respect to the set of model checked formulæ.

1 Introduction

Over the few past years, the interest of the automatic verification research community has taken into consideration the analysis of hybrid automata, modeling systems in which continuous variables interact with discrete modes. As originally defined in [9, 8], hybrid automata combine the traditional automata tools from the Logic community with differential equations systems, and their long tradition in the mathematical community. With this respect, the 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 hybrid automata modeling capability and the decidability of even simple questions such as reachability, was also immediately understood. To date, the major effort of the hybrid automata research community has been devoted to the study of decidable classes of hybrid automata, having at least the reachability problem as decidable [9, 8, 11, 12, 2]. Following a chronological order, the (main) decidable families of timed automata [1], singular automata [9, 8], rectangular automata [8], and o-minimal automata [11] were defined so far in the literature. Unfortunately, for each one of the above families, the lack in the expressiveness of either the discrete or the continuous dynamics [2] that has to be paid for the decidability result, strongly casts the possibility of faithfully capturing complex hybrid systems arising, for example, in the system biology area [14, 7].
Motivated by the above reason, many authors have recently considered the development
of techniques for the symbolic analysis of undecidable—and yet reasonably expressive—
hybrid automata [14, 7, 17, 5, 15]. However, any method developed so far relies either
on the definition of abstractions simulating the underlying hybrid automata [7, 17, 15]
or on symbolic bounded reachability techniques [14, 5]. In the first case, only an over-
approximation of the reachable state-space is possible. Usually, those techniques target
the proof of safety property, stating that something bad 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 temporal logic. In the second case, only an underapproximation
of the reachable state-space can be explored and used for generating counterexample
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, 16] on (undecidable) hybrid automata.
To the knowledge of the authors, no other symbolic technique for the analysis of unde-
cidable hybrid automata is preservative for both true and false reactive systems proper-
ties. Our framework is based on the design of a series of abstraction and a corresponding
three valued semantics for the logic CTL, allowing for the monotonic preservation of
true/false formulæ along the series of abstractions. Given a structure $\mathcal{A}$ in our series,
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. For space constraints, we
collect the proofs of the results in this paper in [6].

2 Preliminaries

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

**Definition 1 (Hybrid Automata [2]).** A Hybrid Automaton is a tuple $H = (L, E, X,$
$Init, \text{Inv}, F, G, R)$ 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 of $\mathbb{R}$
- 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 $\ell$
- $G : E \mapsto 2^{\mathbb{R}^n}$, the guard edge labeling
- $R : E \times \mathbb{R}^n \mapsto 2^L$, the reset edge labeling.

We write $v$ to represent a valuation $(v_1, \ldots, v_n) \in \mathbb{R}^n$ of the variables’ vector $x =$
$(x_1, \ldots, x_n)$, whereas $\dot{x}$ denotes the first derivatives of the variables in $x$ (they all de-
pend on the time, and are therefore rather functions than variables). A state in $H$ is a
pair $s = (\ell, v)$, where $\ell \in L$ is called the discrete component of $s$ and $v$ is called the
continuous component of $s$. A run of $H = (L, E, X, \text{Init}, \text{Inv}, F, G, R)$, starts at any
$(\ell, v) \in \text{Init}$ and consists of continuous evolutions (within a location) and discrete
transitions (between two locations). Formally, a run of $H$ is a path with alternating con-
tinuous and discrete steps in the time abstract transition system of $H$, defined below:
Definition 2. The time abstract transition system of the hybrid automaton $H = (L, E, X, \text{Init}, \text{Inv}, F, G, R)$ is the transition system $T_H = (Q, Q_0, \ell_-, \rightarrow)$, where:

- $Q \subseteq L \times \mathbb{R}^n$ and $(\ell, v) \in Q$ if and only if $v \in \text{Inv}(\ell)$
- $Q_0 \subseteq Q$ and $(\ell, v) \in Q_0$ if and only if $v \in \text{Init}(\ell) \cap \text{Inv}(\ell)$
- $E \cup \{\delta\}$ is the set of edge labels, that are determined as follows:
  - there is a continuous transition $(\ell, v) \xrightarrow{\delta} (\ell, v')$, 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. $f(0) = v$ and $f(t) = v'$
    2. for all $\varepsilon \in (0, t)$, $f(\varepsilon) \in \text{Inv}(\ell)$, and $\dot{f}(\varepsilon) = F(\ell, f(\varepsilon))$.
  - there is a discrete transition $(\ell, v) \xrightarrow{a} (\ell', v')$ if and only if there exists an edge $e = (\ell, \ell') \in E$, $v \in G(\ell)$ and $v' \in R((\ell, \ell'), v)$.

A region is a subset of the states $Q$ of $T_H = (Q, Q_0, \ell_-, \rightarrow)$. Given a region $B$ and a transition label $a \in \ell_-$, the predecessor region $\text{Pre}_a(B)$ is defined as the region \{ $q \in Q$ | $\exists q' \in B. q \xrightarrow{a} q'$ \}. The bisimulation and the simulation relations are two fundamental tools in the context of hybrid automata abstraction.

Definition 3 (Bisimulation). Let $T^1 = (Q^1, Q^1_0, \ell^1_-, \rightarrow^1)$, $T^2 = (Q^2, Q^2_0, \ell^2_-, \rightarrow^2)$ be two edge-labeled transition systems and let $P$ be a partition on $Q_1 \cup Q_2$. A bisimulation for $T_1, T_2$ is a not empty 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 $[p]_P = [q]_P$, where $[p]_P$ denotes the class of $p$ in $P$.
- for each label $a \in \ell_-$, 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 $a \in \ell_-$, 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 bisimulation equivalent (or bisimilar), denoted $T_1 \equiv_B T_2$.

Definition 4 (Simulation). Let $T^1 = (Q^1, Q^1_0, \ell^1_-, \rightarrow^1)$, $T^2 = (Q^2, Q^2_0, \ell^2_-, \rightarrow^2)$ be two edge-labeled transition systems and let $P$ be a partition on $Q_1 \cup Q_2$. A simulation from $T_1$ to $T_2$ is a not empty relation on $\preceq_s \subseteq Q_1 \times Q_2$ such that, for all $p \preceq_s q$:

- $p \in Q^1_0$ iff $q \in Q^2_0$ and $[p]_P = [q]_P$.
- for each label $a \in \ell_-$, if there exists $p'$ such that $p \xrightarrow{a} p'$, then there exists $q'$ such that $p' \preceq_s q'$ 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 $T_1 \preceq_s T_2$. If $T_1 \preceq_s T_2$ and $T_2 \preceq_s T_1$, then $T_1$ and $T_2$ are said simulation equivalent (or similar) and we write $T_1 \equiv_s T_2$.

Definition 6 recall the semantics of the temporal logic CTL (where the $\text{neXt}$ temporal operator is omitted because of the density of the underlying time framework) on hybrid automata [1, 9].
Definition 5 (CTL for Hybrid Automata). Let $\text{AP}$ be a finite set of propositional letters and $p \in \text{AP}$. CTL is the set of formulas defined by the following syntax:

$$\phi ::= p \mid \neg \phi \mid \phi_1 \land \phi_2 \mid E \phi_1 U \phi_2 \mid A \phi_1 U \phi_2 \mid E \phi_1 R \phi_2 \mid A \phi_1 R \phi_2$$

Definition 6 (CTL Semantics). Let $H = \langle L, E, X, \text{Init}, \text{Inv}, F, G, R \rangle$ be an hybrid automaton, let $\text{AP}$ be a set of propositional letters, and let $\ell_{\text{AP}} : L \times X \rightarrow 2^{\text{AP}}$. Given $\phi \in \text{CTL}$ and $s \in Q$, $s \models \phi$ is inductively defined as follows:

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

$H \models \phi$ if 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 [19, 18, 20] for a more comprehensive introduction to o-minimality.

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

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

Given an o-minimal structure $\mathcal{M} = \langle R, <, \ldots \rangle$, 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 $\text{Th}(\cdot)$.

The class of o-minimal structures over the reals is quite rich: both the structure $\mathcal{L}(\mathbb{R}) = \langle \mathbb{R}, <, +, -, 0, 1 \rangle$, used to express linear constraints over the reals, and the ordered real field $\text{OF}(\mathbb{R}) = \langle \mathbb{R}, <, +, -, *, 0, 1 \rangle$ 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 [11]. The variety of o-minimal theories over the reals ensures that the family of o-minimal hybrid automata as introduced in [11, 12] (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 [11].

**Definition 8.** Let \( F : \mathbb{R}^n \to \mathbb{R}^n \) 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 \( \phi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n \) given by \( \phi(\mathbf{v}, t) = \gamma_{\mathbf{v}}(t) \).

**Definition 9 (O-Minimal Hybrid Automata [11]).** The hybrid automaton \( H = (L, E, X, \text{Init}, \text{Inv}, F, G, R) \) 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 \( R : E \to \mathbb{R}^n \) does not depend on continuous variables (constant resets)
- c) for each \( \ell \in L \) and \( (\ell, \ell') \in E \), the sets \( \text{Inv}(\ell), R(\ell, \ell'), G(\ell), \text{Init}(\ell) \) and the flow of \( F(\ell, \cdot) \) are definable in the same o-minimal structure.

Given an o-minimal structure \( M \), the o-minimal hybrid automata induced by \( M \) are called o-minimal(\( M \)) hybrid automata. O-minimal hybrid automata admit a finite bisimulation quotient [11]. Computability of such a bisimulation quotient (and hence decidability) depends on the underlying o-minimal structure: in [12], the class of o-minimal (\( \text{OF}(\mathbb{R}) \)) hybrid automata and various subclasses of o-minimal (\( \text{OF}_{\exp}(\mathbb{R}) \)) automata were proven to be decidable with respect to reachability.

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

Throughout this section and Section 4, we solve the problem of defining a series 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 ask that the set of formulæ evaluating to \( \perp \) (according to the three valued semantics over the abstractions) decreases monotonically its size along the series of abstractions.

A first idea is that of exploiting the classical notion of \( n \)-bounded bisimulation. In fact, a simulation preorder relates successively finer bounded bisimulations, allowing to establish a monotonic preservation result for the set of true universally quantified formulæ along the series of abstractions. Since the \( n \)-bounded bisimulation is characterized by the \( n \)-bounded modal logic (where at most \( n \) \text{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 above 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 \text{neXt}
temporal operator gets meaningless in the corresponding logics. Second, a quest like
$s \models E\phi_1 U\phi_2$ can never be checked considering only a finite path departing from $s$ in
the time abstract transition system $T_H$ of an hybrid automaton $H$. In fact, due to the
dense nature of the underlying time framework, each finite path of the kind $z \xrightarrow{\delta} z'$ sub-
sumes an uncountable number of continuous transitions’ infinite paths in $T_H$, on which
$E\phi_1 U\phi_2$ 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

On the ground of the above discussion, we develop here a new series of hybrid automata
abstractions suitable for our purposes, that we call discrete bounded bisimulation ab-
stractions. It is well known that the classic bisimulation equivalence can be charac-
terized as a coarsest partition stable with respect to a given transition relation $[10]$. 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 $T_H = (Q, Q_0, \ell, \rightarrow)$ of a hybrid automaton $H$, and let $P$ be a
partition on $Q$:

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

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 (a) $p \equiv_{n-1} q$
   - (b) $\forall p'(p \xrightarrow{\delta} p') \Rightarrow \exists q'(p' \equiv_n q' \land q \xrightarrow{\delta} q')$
   - (c) $\forall q'(q \xrightarrow{\delta} q') \Rightarrow \exists p'(p' \equiv_n q' \land p \xrightarrow{\delta} p')$
   - (d) $\forall p'(p \xrightarrow{e} p') \Rightarrow \exists q'(p' \equiv_{n-1} q' \land q \xrightarrow{e} q')$
   - (e) $\forall q'(q \xrightarrow{e} q') \Rightarrow \exists p'(p' \equiv_{n-1} q' \land p \xrightarrow{e} p')$

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

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

- $Q' = Q_{/\equiv_n}$, $Q'_0 = Q_0_{/\equiv_n}$ and $\ell' = \ell_{/\equiv_n}$.
\[ \forall \alpha, \beta \in Q': \\
\begin{itemize}
  \item \( \alpha \rightarrow^s \beta \) iff \( \exists s \in \alpha, \exists q \in \beta (s \rightarrow q) \) by traversing the only regions \( \alpha \) and \( \beta \)
  \item \( \alpha \delta \rightarrow \beta \) iff \( \exists s \in \alpha, \exists q \in \beta (s \delta \rightarrow q) \)
\end{itemize}

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

**Lemma 1.** Let \( H \) be an hybrid automaton, and consider the series 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 \( T_H \equiv_B H_{/\equiv n} \).

On the ground of Lemma 2, it is also possible to use the series 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 traverse at most \( n \) locations. Given two states in an hybrid automaton \( H \), we use the notation \( q \overset{n}{\rightarrow} q' \) to state that \( q' \) is reachable from \( q \) following a run that contain at most \( n \) discrete edges (i.e. traverses at most \( n \) locations of \( H \)).

**Lemma 2.** Let \( p, q \) be two state in an 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:

- For all \( p' \) such that \( p \overset{m}{\rightarrow} p' \), there exists \( q' \) such that \( p' \equiv_n q' \) and \( p' \overset{m}{\rightarrow} q' \).
- For all \( q' \) such that \( q \overset{m}{\rightarrow} q' \), there exists \( p' \) such that \( p' \equiv_n q' \) and \( p' \overset{m}{\rightarrow} q' \).

### 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 \( n \)-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 \( \text{DBB}(n, H, \mathcal{P}_0) \) terminates, then algorithm \( \text{DBB}(n, H, \mathcal{P}_0) \) computes the quotient of the \( n \)-DBB equivalence with respect to \( \mathcal{P}_0 \) on \( T_H \).

Using a number of techniques developed in [11, 12], it is rather easy to obtain \( n \)-DBB finiteness and computability results for the broad undecided family of fully o-minimal hybrid automata (cfr. Definition 12). The latter extends the o-minimal based
systems in [11] 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 [9, 8], or the undecidable classes studied in [3, 13]).

**Definition 12 (Fully O-Minimal Hybrid Automata).** The hybrid automaton $H = (L, E, X, \text{Init}, \text{Inv}, F, G, R)$ 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 $\text{Inv}(\ell)$, $G(\ell)$, $\text{Init}(\ell)$, 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 $P$ be an initial finite partition over the state-space of $H$ definable in $\mathcal{M}$. Then, the algorithm 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 (OF(\mathbb{R})) hybrid automata has a finite and effectively computable $n$-DBB abstraction. The result depends on the fact that OF(\mathbb{R}) is a decidable theory admitting quantifier elimination. Therefore, the theory OF(\mathbb{R}) provides the means for representing sets, computing post-images and boolean compositions, as well as checking for set emptiness.
Corollary 1. Given $n \in \mathbb{N}$, the $n$-DBB abstraction on fully o-minimal\((\text{OF}_n(\mathbb{R}))\) hybrid automata is finite and computable.

Techniques similar to the ones adopted in [11] can be used to obtain further computability results\(^1\) on subclasses of o-minimal(\(\text{OF}_\exp(\mathbb{R}))\) hybrid automata.

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

- for each $\ell \in L$, the vector field is of the form $F(\ell, x) = Ax$, 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 $ir$, $r \in \mathbb{Q}$, with diagonal real Jordan form.

- for each $\ell \in L$ and $(\ell, \ell') \in E$, the sets $\text{Inv}(\ell)$, $\text{G}(\ell)$, $\text{Init}(\ell)$, and the reset function $R(\ell, \ell') : X \rightarrow X$ are definable inside \(\text{OF}(\mathbb{R})\).

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 series \(\langle H_{\equiv_n} \rangle_{n \in \mathbb{N}}\). The latter allows us to use unbounded runs in the evaluation of (not purely existential) CTL properties. In other words, each CTL\(\backslash\)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 \(\{\text{tt}, \text{ff}, \bot\} \setminus \{\bot\}\). This is a key point in endowing our framework to face both the refutation and the proof of safety or liveness properties over an 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 $P$ be the partition on $Q$ induced by the labelling function $\ell_P : Q \mapsto 2^{\text{AP}}$. Consider the $n$-DBB abstraction of $H$ with respect to $P$, $H_{\equiv_n}$. Given the node $[s]_{\equiv_n}$ in $H_{\equiv_n}$, the value $[s]_{\equiv_n} \models_3 \phi \in \{\text{tt}, \text{ff}, \bot\}$ is inductively defined on $n$ as follows:

- If $\phi = p$, then $[s]_{\equiv_n} \models_3 \phi$ is defined as:
  $$\begin{cases} 
  \text{tt} & \text{iff } p \in \ell_p([s]_{\equiv_n}) \\
  \text{ff} & \text{iff } p \notin \ell_p([s]_{\equiv_n}) 
  \end{cases}$$

- If $\phi = \neg \phi_1$, then $[s]_{\equiv_n} \models_3 \phi$ is defined as:
  $$\begin{cases} 
  \text{tt} & \text{iff } [s]_{\equiv_n} \models_3 \phi_1 = \text{ff} \\
  \text{ff} & \text{iff } [s]_{\equiv_n} \models_3 \phi_1 = \text{tt} \\
  \bot & \text{otherwise.}
  \end{cases}$$

\(^1\) In particular, techniques analogous to the ones used in [12] can be used to show that, though the flow characterizing the automata in Corollary 2 are definable inside $\text{OF}_\exp(\mathbb{R})$, set preimages with respect to the continuous transitions can be expressed inside $\text{OF}(\mathbb{R})$. 
Theorem 3. Let \( \langle H_{/m} \rangle_{m \in \mathbb{N}} \) be the series of DBB abstractions for an hybrid automaton \( H \). For any CTL formula \( \phi \), for any \( n \in \mathbb{N} \) it holds:

\[
(\forall m > n (\lnot [H_{/m} \models \phi] = b) \to \forall m > n (\lnot [H_{/m} \models \phi] = b)
\]

Note that, according to Definition 13, a simple trick can be used to exploit also unbounded runs in the evaluation of a purely existential property. Namely, let \( \phi = E\phi_1 U \phi_2 \) be a formula in ECTL: Then, it is sufficient to consider, in place of the property of interest \( \phi \), the stronger condition \( E\phi_1 U (A\phi_1 U \phi_2) \).

Theorem 2 establishes a preservation result for true/false CTL formulæ evaluated over a DBB abstraction of a given hybrid automaton \( H \). Finally, theorem 3 guarantees the monotonicity of the above preservation result along our series of DBB abstractions.
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 problems: Namely, the definition of a procedure for the evaluation of $[H/\alpha_n]_\gamma$ for all $\phi$, were $\phi$ is either a universal or an existential formula of $\gamma$. Let $\phi$ be an $\alpha$-CTL formula, and let $\alpha$ be a node in $H/\alpha_n$. According to Definition 13, $[\alpha]=_\gamma \phi = \text{tt}$ iff $\alpha = \phi$ (with respect to the classical 2-valued semantics for $\gamma$ on the finite transition system $H/\alpha_n$). Hence, it is sufficient to use a classical (2-valued) CTL model checking algorithm on $H/\alpha_n$ to detect those nodes in $H/\alpha_n$ for which $[\alpha]=_\gamma \phi = \text{tt}$ in time $O(|H/\alpha_n| \times |\phi|)$.

The problem of collecting the nodes $\alpha$ in $H/\alpha_n$ for which $[\alpha]=_\gamma \phi = \text{tt}$ reduces to the problem of (1) rewriting $\neg \phi$ as a formula $\gamma$ in ECTL (2) determining the set of states $\alpha$ in $H/\alpha_n$ for which $[\alpha]=_\gamma \phi = \text{tt}$.

Summarizing the above observations, we can derive an overall $O(|H/\alpha_n| \times |\phi|)$ algorithm for computing $H/\alpha_n$, if we prove possible to recognize all those nodes $\alpha$ for which $[\alpha]=_\gamma \phi = \text{tt}$, $\gamma \in ECTL$, in time $O(|H/\alpha_n| \times |\gamma|)$.

Let $\gamma$ be a ECTL formula: we solve the above subproblem proposing an efficient ($O(|H/\alpha_n| \times |\gamma|)$) strategy to distribute on the nodes in $H/\alpha_n$ the set $\text{of labels } \{ \psi, \text{tt}, m \}$, were:

- $\psi$ is a subformula of $\gamma$ and $m \leq n$.
- $\alpha = [s]_\alpha$ receives the label $\{ \psi, \text{tt}, m \}$ if and only if $[s]_\alpha = \text{tt} \land \forall m' < m ([s]_m' = \perp)$.

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

The case for which $\gamma = E\gamma_1 U\gamma_2$ requires instead some more attention. As illustrated in Figure 2 (cfr. lines (3.1)–(3.12) of the procedure PROCESS), given $N = |H/\alpha_n|$, we first build the two vectors $A_1[1, \ldots, N], A_2[1, \ldots, N]$ of lists of nodes in $H/\alpha_n$, were $\alpha \in A_i[i]$ iff the label $\gamma_j$, $\text{tt}, i$ has been inductively associated to $\alpha$. Note that $A_1[1 \ldots N]$ (resp. $A_2[1 \ldots N]$) requires space $O(|H/\alpha_n|)$, since the lists in each slot of the array $A$ are disjoint. Then, by induction on $i = 0 \ldots n$, we build the sets $S_0, \ldots, S_n$, were $S_i$ contains all the nodes of $H/\alpha_n$ that need to be labeled with $\{ \gamma, \text{tt}, i \}$. 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 otherways. Given the above colorage we let $S_i$ to contain:
ALGO1(H, A, n, φ)
Input: n ∈ N, the discrete n-bounded bisimulation abstraction H, a formula φ ∈ ACTL
Output: |H|=φ ∈ {tt, ff, ⊥}

(1) Let γ ≡ ¬φ, where γ is in negation normal form
(2) if (φ ∈ ACTL) then PROCESSA(H, φ) End
(3) if (φ ∈ ACTL) then PROCESSA(H, γ) End
(4) if (∀α ∈ Init(H, φ)) then return tt
(5) if (∃α ∈ Init(H, φ)) then return ff else return ⊥

PROCESSA(H, φ)
Input: n ∈ N, the discrete n-bounded bisimulation abstraction H, a formula φ ∈ ACTL
Output: ∀[s] ∈ H, [s] is labeled ⟨φ, tt, ⊥⟩ iff [[s] |= φ] = tt

/* Use a 2-valued model checking procedure to process the formula φ on H */
(1) for each (α ∈ H) do
(2) Aφ0Uφ1 then ε(α) ← ε(α) ∪ \{⟨φ, tt, ⊥⟩\}

Fig. 2. The linear algorithm for ACTL/ECTL 3-valued Model Checking on H.
Each not red node in the list \( A_2[i] \).
- Each yellow node in the list \( A_1[i] \).
- Each node \( \beta \in H_{/m_n} \) admitting a path \( p \) to \( S_{i-1} \) such that:
  1. \( p \) contains at most one edge labeled \( e \)
  2. the label \( (\gamma_1 \lor \gamma_2, tt, i) \) is associated to each node of \( p \).

Finally, if \( \gamma = E\gamma_1 R\gamma_2 \), we can reduce to process the formula \( \gamma' = E\gamma_2 U \) according to the three-valued CTL semantics in Definition 13.

The pseudocode of the overall algorithm above sketched for determining \( [H_{/m_n} \models \phi] \), \( \phi \in \text{ECTL} \cup \text{ACTL} \), is given in Figure 2. Given the formula \( \phi \in \text{ECTL} \cup \text{ACTL} \) in negation normal form, Theorem 4 states that our procedure \( \text{ALGO1}(H_{/m_n}, \phi) \) computes in time \( O(|H_{/m_n}| * |\phi|) \) the value \( [H_{/m_n} \models \phi] \).

**Theorem 4.** The algorithm \( \text{ALGO1}(H_{/m_n}, n, \phi) \) computes \( [H_{/m_n} \models \phi] \in \{tt, ff, \bot\} \) in time \( O(|H_{/m_n}| * |\phi|) \) and space \( O(|H_{/m_n}|) \).

### 3-Valued CTL Model Checking

In this Subsection we extend the techniques outlined in Subsection 5.1 to design a \( O(|H_{/m_n}| * |\phi|) \) algorithm for computing \( H_{/m_n} \models \phi \), were \( \phi \) is a general CTL formula. We start with some preliminary observations to illustrate the bottlenecks related to the above extension. Let \( \phi_1 = EpUq \), \( \phi_2 = ApUq \) and consider the two formulæ belonging to \( \text{CTL} \setminus (\text{ECTL} \cup \text{ACTL}) \):

- \( \psi_1 = ArU\phi_1 \) \hspace{1cm} (1)
- \( \psi_2 = ErU\phi_2 \) \hspace{1cm} (2)

Since \( \phi_1, \phi_2 \), and \( r \) belong to \( \text{ECTL} \cup \text{ACTL} \), we can assume having determined with \( \text{ALGO1} \) the set of nodes \( \alpha \in H_{/m_n} \) for which \( [\alpha] = tt \), \( \varsigma \in \{\phi_1, \phi_2, r\} \). Then, the task of processing the formula (1) according to Definition 13 does not present any problem. In fact, determining each node \( \beta \) for which \( [\beta] = tt \) boils down to apply a classical model checking subprocedure targeting the operator \( AU \) (were 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 \( \psi_2 = ErU\phi_2 \) in (2) according to the 3-valued semantics in Definition 13. Namely, for each \( \alpha = [s]_{m_n} \) such that \( [\alpha] = tt \), we need to know the least \( m \leq n \) for which \( [s]_{m_n} = tt \). If \( \phi_2 \) were a formulæ having an existential main path quantifier, say \( \phi_2 = \exists \psi_3 U \psi_4 \), then such minimum indexes would have been dependent to the number of discrete edges that one need to traverse to evaluate \( \phi_2 \) (assuming a previous labeling relative to \( \phi_3 \) and \( \phi_4 \)). However, formulæ having a main universal path quantifier are evaluated by considering any path in \( H_{/m_n} \), regardless to 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 \( O(|H_{/m_n}|) \). The formal description of a compact partition tree is given in Definition 14.

---

\footnote{In fact, the indices \( m \) are determined ‘on the fly’ in our procedure \text{PROCESSE} for ECTL formulæ.
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 \( \equiv_n, \phi, \gamma \equiv \neg \phi \in \text{CTL} \), where \( \phi \) and \( \gamma \) are in negation normal form

Output: \([H_{/\equiv_n}] = \{\text{tt, ff, } \bot\} \)

\begin{enumerate}
  \item \( (1.1) \) if \( (\phi \in \{p, \neg p\} \text{ for some } p \in \text{AP}) \) then
  \item \( (1.2) \) for each \( \alpha \in Q_{/\equiv_n} \) do
    \item \( (1.3) \) if \( \ell_0(\alpha) = \text{tt} \) then \( \ell(\alpha) \leftarrow \ell(\alpha) \cup \{\{p, \text{tt}, 0\} \} \) else \( \ell(\alpha) \leftarrow \ell(\alpha) \cup \{\neg p, \text{tt}, 0\} \)
  \item \( (2.1) \) if \( (\phi = \phi_1 \lor \phi_2, \phi \in \{\lor, \land\}) \) then
    \item \( (2.2) \) for each \( \alpha \in Q_{/\equiv_n} \) do
      \item \( (2.3) \) if \( ([\phi_1, \text{tt}, m_1] \in \ell(\alpha)] \lor ([\phi_2, \text{tt}, m_2] \in \ell(\alpha)] \) then
        \item \( (2.4) \) \( \ell(\alpha) \leftarrow \ell(\alpha) \cup \{\langle \phi, \text{tt}, \max(m_1, m_2)\} \)
  \item \( (3.1) \) if \( (\phi = \text{E} \phi_1 \lor \phi_2 \land \gamma = A \gamma_1 R \gamma_2) \) then
    \item \( (3.2) \) ALGO2\( (H_{/\equiv_n}, T_n, n, \phi_1, \gamma_1) \); ALGO2\( (H_{/\equiv_n}, T_n, n, \phi_2, \gamma_2) \)
    \item \( (3.3) \) PROCESSER\( (H_{/\equiv_n}, T_n, n, \phi_1, \phi_2) \); PROCESSAU\( (H_{/\equiv_n}, T_n, n, \gamma_1, \gamma_2) \)
  \item \( (4.1) \) if \( (\phi = \text{A} \phi_1 \lor \phi_2 \land \gamma = A \gamma_1 R \gamma_2) \) then
    \item \( (4.2) \) ALGO2\( (H_{/\equiv_n}, T_n, n, \phi_1, \gamma_1) \); ALGO2\( (H_{/\equiv_n}, T_n, n, \phi_2, \gamma_2) \)
    \item \( (4.3) \) PROCESSER\( (H_{/\equiv_n}, T_n, n, \phi_1, \phi_2) \); PROCESSAU\( (H_{/\equiv_n}, T_n, n, \gamma_1, \gamma_2) \)
  \item \( (5.1) \) if \( (\phi = \text{E} \phi_1 \lor \phi_2 \land \gamma = A \gamma_1 E \gamma_2) \) then
    \item \( (5.2) \) ALGO2\( (H_{/\equiv_n}, T_n, n, \phi_1, \gamma_1) \); ALGO2\( (H_{/\equiv_n}, T_n, n, \phi_2, \gamma_2) \)
    \item \( (5.3) \) PROCESSER\( (H_{/\equiv_n}, T_n, n, \phi_1, \phi_2) \); PROCESSAU\( (H_{/\equiv_n}, T_n, n, \gamma_1, \gamma_2) \)
  \item \( (6.1) \) if \( (\phi = \text{A} \phi_1 \lor \phi_2 \land \gamma = A \gamma_1 E \gamma_2) \) then
    \item \( (6.2) \) ALGO2\( (H_{/\equiv_n}, T_n, n, \phi_1, \gamma_1) \); ALGO2\( (H_{/\equiv_n}, T_n, n, \phi_2, \gamma_2) \)
    \item \( (6.3) \) PROCESSER\( (H_{/\equiv_n}, T_n, n, \phi_1, \phi_2) \); PROCESSAU\( (H_{/\equiv_n}, T_n, n, \gamma_1, \gamma_2) \)
  \item \( (7.1) \) if \( (\forall \alpha \in \text{Init}(Q_{/\equiv_n}) (\langle \phi, \text{tt}, \neg \rangle) \in \ell(\alpha)) \) then return \text{tt}
  \item \( (7.2) \) if \( (\exists \alpha \in \text{Init}(Q_{/\equiv_n}) (\langle \gamma, \text{tt}, \neg \rangle) \in \ell(\alpha)) \) then return \text{ff} else return \text{\bot}
\end{enumerate}

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

Definition 14 (Compact Partition Tree associated to \( H_{/\equiv_n} \)). Let \( H \) be an 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 \( \ell(f) = (0, 0, \alpha) \).
- \( T_{n+1} \) 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 \( \alpha \) 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 $\ell(f) = (m, n - 1)$. For each $\alpha \subseteq \alpha_i \in H_{\psi}$, create a new successor of $f$, $f_i$, and associate to $f_i$ the label $\langle n, n, \alpha_i \rangle$.

**Fig. 4.** The subprocedures $\text{PROCESSAU}$, and $\text{PROCESSEU}$ used in $\text{ALGO2}$
Lemma 4. Let $H$ be an hybrid automaton. The space complexity of the compact partition tree associated to $H_{/w_n}$ is $O(|H_{/w_n}|)$.

Compact partition trees can be easily computed along discrete bounded bisimulations.

Let $\phi \in \text{CTL}$ and assume that the main path quantifier in $\phi$ is universal. Suppose having determined the set of nodes $S = \{[s]_{m} \in H_{/w_n} \mid [s]_{m} \models \phi\} = \{s\}$. Then, the compact partition tree $T_n$ can be used as follows to associate to each node $\alpha = [s]_{m} \in S$ (in time $O(|H_{/w_n}|)$) the required label $\langle \phi, tt, m \rangle$, were $m$ is the minimum index such that $[s]_{m} \models \phi = tt$. 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 descendent of the node $k$ is associated to a class $\alpha \in H_{/w_n}$ for which $\alpha \models \phi = tt$
2. $k$ is a node of minimal depth having property 1.

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

\[
\text{PROCESSAR}(H_{/w_n}, T_n, n, \phi_1, \phi_2) \\
\text{Input: For } \psi \in \{\phi_1, \phi_2\}, \text{ each node } \alpha = [s]_{m} \text{ in } H_{/w_n} \text{ is assumed to be labeled with } \langle \psi, tt, m \rangle \\
\text{if } [[s]_{m} \models \psi] = tt \land \forall k < m \iff [[s]_{k} \models \psi] = \bot \\
\text{Out: } \forall [s]_{n} \in H_{/w_n}, [s]_{n} \text{ is labeled } \langle \phi = A\phi_1 R\phi_2, tt, m \rangle \iff [[s]_{m} \models \phi] = tt \land \forall k < m \iff [[s]_{k} \models \phi] = \bot
\]

/* Use a 2-valued model checking procedure targeting the AR operator to process the formula $A\phi_1 R\phi_2$ */
/* assuming, for $\psi \in \{\phi_1, \phi_2\}$, that $\alpha \models \psi$ iff $\psi tt, -) \in \ell(\alpha)$ */

(1.1) for each $\alpha \in H_{/w_n}$ do
(1.3) if $\alpha \models A\phi_1 R\phi_2$ then $\ell(\alpha) \leftarrow \ell(\alpha) \cup \{A\phi_1 R\phi_2, tt, n\}$
(1.4) Use a depth first search like algorithm on $T_n$ to determine the set of $T_n$ nodes $S$ /*
(1.5) for each $(k \in S, \text{leaf } f \text{ descending from } k)$ do
(1.6) if $(\ell(f) = (-, -, \alpha) \land \ell(k) = (m, -))$
(1.7) then $\ell(\alpha) \leftarrow (\ell(\alpha) \setminus \{A\phi_1 R\phi_2, tt, n\}) \cup \{A\phi_1 R\phi_2, tt, m\}$

\[
\text{PROCESSEU}(H_{/w_n}, T_n, n, \phi_2, \phi_1) \\
\text{Input: For } \psi \in \{\phi_1, \phi_2\}, \text{ each node } \alpha = [s]_{m} \text{ in } H_{/w_n} \text{ is assumed to be labeled with } \langle \psi, tt, m \rangle \\
\text{if } [[s]_{m} \models \psi] = tt \land \forall k < m \iff [[s]_{k} \models \psi] = \bot \\
\text{Out: } \forall [s]_{n} \in H_{/w_n}, [s]_{n} \text{ is labeled } \langle \phi = E\phi_1 R\phi_2, tt, m \rangle \iff [[s]_{m} \models \phi] = tt \land \forall k < m \iff [[s]_{k} \models \phi] = \bot
\]

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

Fig. 5. The subprocedures PROCESSER, and PROCESSAR used in ALGO2
Figure 3 illustrates the pseudocode of our final algorithm ALGO2, computing \( [H/f_n]|=\phi \), where \( \phi \in \text{CTL} \), in time \( O(|H/f_n| \star |\phi|) \) and space \( O(|H/f_n|) \). ALGO2 requires five parameters in input. Namely, the DBB abstraction \( H/f_n \) and its index \( n \), the compact partition tree \( T_n \) associated to \( \equiv n \), the formula of interest \( \phi \) and \( \gamma \equiv \neg \phi \) (where \( \phi, \gamma \) are assumed in negation normal form). ALGO2 proceeds by structural induction on \( \phi \); It uses an efficient strategy to distribute on the nodes in \( H/f_n \) the set of of labels \( \langle \psi, \text{tt}, m \rangle \), were:

- \( \psi \) is a subformula either of \( \phi \) or of \( \gamma \), and \( m \leq n \).
- \( \alpha = [s]_m \) receives the label \( \langle \psi, \text{tt}, m \rangle \) if and only if
  \[
  [s]_m|=3\psi = \text{tt} \land \forall m' < m([s]_{m'}|=3\psi) = \bot
  \]

Theorem 5 finally states correctness and complexity of our three valued model checking algorithm on discrete bounded bisimulation abstractions for CTL. Let \( \phi, \gamma \) be two formulæ of CTL in negation normal form, where \( \gamma \equiv \neg \phi \).

**Theorem 5.** The algorithm ALGO2(\( H/f_n, T_n, n, \phi, \gamma \)) computes \( [H/f_n]|=\phi \) \( \in \{\text{tt, ff, } \bot\} \) in time \( O(|H/f_n| \star |\phi|) \) and space \( O(|H/f_n|) \).

### 6 Conclusions

In this paper we develop a framework to both prove and disprove reactive system properties expressed by means of CTL logic on (undecidable) hybrid automata. To the knowledge of the authors, no other symbolic technique for the analysis of undecidable hybrid automata can cope with both the proof and the refutation of, for example, safety or liveness properties. Our framework is based on the design of a series of abstraction and a corresponding three valued semantics for the logic CTL, allowing for monotonic preservation of formulæ evaluating to \( \{\text{tt, ff, } \bot\} \setminus \{\bot\} \) along the series of abstractions. We show that the three valued CTL model checking problem on our DBB abstractions, is linear in the length of the formula and in the size of the abstraction.

### References


7 Appendix

Proof of Lemma 1 Let $H = (L, E, X, \text{Init, Inv, } F, G, R)$ be an hybrid automaton, let $\mathcal{P}$ be a partition on the state space $Q = L \times X$ of $H$, and consider the two abstractions $H_{/\equiv_n}$ and $H_{/\equiv_n+1}$, where $n \in \mathbb{N}$. We prove that the two relations $\leq_1^{n+1} \subseteq Q_{/\equiv_n+1} \times Q_{/\equiv_n} = \{([s]_{\equiv_n+1}, [s]_{\equiv_n}) \mid s \in Q\}$ and $\leq_2^{n} \subseteq Q \times Q_{/\equiv_n} = \{([s]_{\equiv_n+1}) \mid s \in Q\}$ are two simulation preorders from $H_{/\equiv_n}$ to $H_{/\equiv_n+1}$, and from $H_{/\equiv_n}$ to $H$, respectively. The first condition in the definition of simulation (cfr. Definition 4) is immediately established for $\leq_1^{n}$, $\leq_2^{n}$, on the ground of item 1.(a) in Definition 10. For the second condition in Definition 4, let $[s]_{\equiv_n+1} \xrightarrow{a} [q]_{\equiv_n+1}$ (resp. $s \xrightarrow{a} q$). Then Definition 11 guarantees that $[s]_{\equiv_n+1} \xrightarrow{a} [q]_{\equiv_n}$.

Finally, assume that $H_{/\equiv_n}$ coincides with $H_{/\equiv_n+1}$. Then, by Definition 10 and Definition 3 we can immediately conclude that $T_H \equiv_B H_{/\equiv_n}$.
Proof of Lemma 2 Let $\equiv_n$ be the $n$-DBB equivalence on the hybrid automaton $H$ with respect to the partition $\mathcal{P}$. Assume $p \equiv_n q$ and $p \xrightarrow{m} p'$, where $m \leq n$. By induction on $m$, we prove that $q \xrightarrow{m} q'$, for some $p' \equiv_{n-m} q'$.

For the base case, assume $p \xrightarrow{0} p'$ and let $p \xrightarrow{\delta} p_1 \xrightarrow{\delta} \ldots \xrightarrow{\delta} p_k \xrightarrow{\delta} p'$ be the shortest path in $T_H$ witnessing $p \xrightarrow{0} p'$. Then, by items 2.(a)–2.(c) in Definition 10, $T_H$ admits a path $q \xrightarrow{\delta} q_1 \xrightarrow{\delta} \ldots \xrightarrow{\delta} q_k \xrightarrow{\delta} q'$ such that for all $1 \leq i \leq k$, $p_i \equiv_0 q_i$ and $p' \equiv_0 q'$. For the inductive step, let $p \xrightarrow{m} p'$, $1 \leq m \leq n$. If $p \xrightarrow{1} p'$, then the inductive hypothesis is sufficient to ensure our thesis. Otherwise, $p \xrightarrow{m-1} p'' \xrightarrow{0} p''' \xrightarrow{0} p'$. By Items 2.(a)–2.(c) in Definition 10, and using our inductive hypothesis, we get $q \xrightarrow{m-1} q'' \xrightarrow{0} q'$, where $p'' \equiv_1 q''$, $p''' \equiv_0 q'''$, and $p' \equiv_0 q'$. □

Proof of Theorem 1 Within our main proof of Theorem 1, we use the results in Proposition 1 due to [11]. The latter refers to the termination of the algorithm reported in Figure 6 targeting the bisimulation on o-minimal hybrid automata with constant resets.

Proposition 1 ([(11, 3)]. Let $\ell$ be a location in an o-minimal ($\mathcal{M}$) hybrid automaton, and let $F(\ell) : \mathbb{R}^n \mapsto \mathbb{R}^n$ be the corresponding vector field, whose flow is definable in $\mathcal{M}$. If $P_\ell$ is a finite partition over $\{\ell\} \times \mathbb{R}^n$ having classes definable in $\mathcal{M}$, then the procedure $\text{SPLITLOC}(F(\ell), P_\ell)$ terminates.

<table>
<thead>
<tr>
<th>$\text{BISIMLOC}(H = \langle L, E, X, \text{Init}, \text{Inv}, F, G, R \rangle)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ell$</td>
</tr>
<tr>
<td>for each $\ell \in L$</td>
</tr>
<tr>
<td>define $A_\ell = {\text{Inv}(\ell), \text{Init}(\ell), \text{Final}(\ell)} \cup {G(\ell, \ell' \mid (\ell, \ell') \in E)} \cup$</td>
</tr>
<tr>
<td>$\cup {R(\ell', \ell \mid (\ell', \ell) \in E)}$</td>
</tr>
<tr>
<td>let $P_\ell$ be the coarsest partition of ${\ell} \times \mathbb{R}^n$ respecting each block ${\ell} \times Z$, where $Z \in A_\ell$;</td>
</tr>
<tr>
<td>$\ell$</td>
</tr>
<tr>
<td>for each $\ell \in L$</td>
</tr>
<tr>
<td>$P_\ell \leftarrow \text{SPLITLOC}(F(\ell), P_\ell)$;</td>
</tr>
<tr>
<td>return $(\bigcup_{\ell \in L} P_\ell)$</td>
</tr>
<tr>
<td>$\text{SPLITLOC}(F(\ell), P_\ell)$</td>
</tr>
<tr>
<td>$\ell$</td>
</tr>
<tr>
<td>while $(\exists B, B' \in P_\ell$ such that $\emptyset \neq B \cap \text{Pre}_e(B') \neq B)$</td>
</tr>
<tr>
<td>$B_1 \leftarrow B \cap \text{Pre}_e(B')$, $B_2 \leftarrow B \setminus \text{Pre}_e(B')$;</td>
</tr>
<tr>
<td>$P_\ell \leftarrow (P_\ell \setminus {B}) \cup {B_1, B_2}$;</td>
</tr>
<tr>
<td>return $P_\ell$</td>
</tr>
</tbody>
</table>

Fig. 6. Partition refinement bisimulation algorithm for o-minimal automata [11]

Main Proof of Theorem 1: Let $\mathcal{M}$ be the o-minimal structure underlying the definition of $H$. First of all, notice that if $B$ is a class definable in $\mathcal{M}$, then both $\text{Pre}_e(B)$ and $\text{Pre}_r(B)$ are definable in $\mathcal{M}$, by set/map o-minimal definability closedness properties and by definition of $\text{Pre}_e(\cdot)$, and $\text{Pre}_r(\cdot)$:
Proof of Theorem 2  By structural induction on $\phi$ we prove that for all $[s]_{\equiv_n} \in Q_{\equiv_n}$:

$$\models_{\equiv_n} \phi$$

The thesis follows easily from the above result. In fact, assume $[H]_{\equiv_n} = tt$. Then, for all $[s]_{\equiv_n} \in Q_{\equiv_n}$, $[s]_{\equiv_n} \models_{\equiv_n} \phi$ i.e., by the result in (1) and by Definition 11, for all $s \in Q^0$ it holds that $s \models \phi$, implying $H \models \phi$. On the other hand, if $[H]_{\equiv_n} = ff$, then there exists $[s]_{\equiv_n} \in Q^0_{\equiv_n}$ such that $[s]_{\equiv_n} \models_{\equiv_n} \phi$ i.e., $[s]_{\equiv_n} \models_{\equiv_n} \neg \phi$. Hence, by the result in (1) and by Definition 11, there exists an initial state of $H$ such that $\gamma(s) \models \neg(H \models \phi)$.

We report here a complete proof of (1) only for the operators $\boxplus, \boxdot, \boxcup$, since the other cases are either symmetrical or simpler.

- Let $\phi = E\phi_1 U\phi_2$. We prove our claim in (1) by induction on the index $n$ for the DBB-abstraction.

Base ($n = 0$) Assume $[s_1]_{\equiv_n} \equiv_{\equiv_n} \phi_1 \models \phi_2$ = $tt$. Then, by Definition 13, there exists a path $[s_1]_{\equiv_0} \delta \ldots \delta [s_k]_{\equiv_0}$ in $H_{\equiv_0}$ such that $[s_k]_{\equiv_0} \equiv_{\equiv_n} \phi_2 = tt$ and for all $1 \leq i \leq k$, $[s_i]_{\equiv_0} \equiv_{\equiv_n} \phi_1 \lor \phi_2 = tt$. By Items 1(b)–1(c) in Definition 10, by Definition 11, and by our structural inductive hypothesis on $\phi$, $H$ admits a run departing from $s_1$ and traversing the classes $[s_i]_{\equiv_n}, 1 \leq i \leq k$, which witnesses $s_1 \models \phi$.

Inductive Step ($n > 0$) Assume $[s_1]_{\equiv_n} \equiv_{\equiv_n} \phi_1 \models \phi_2 = tt$. Then, by Definition 13, there exists a path $([s_i]_{\equiv_n})_{1 \leq i \leq k}$ such that:

1. For all $i < k$ ($[s_i]_{\equiv_n} \equiv_{\equiv_n} \phi_1 \lor \phi_2 = tt$)
2. \([s_k]_{=n} \models \exists \phi \theta = \text{tt} \lor ([s_k]_{=n} \models \exists \phi \theta = \text{tt})\)

In case the first branch in the second condition is satisfied, we get to our claim by Items 2.\((a)\)–2.\((c)\) in Definition 10 and by our inductive hypothesis. Otherwise, assume that \([s_k]_{=n} \models \exists \phi \theta = \text{tt} \lor ([s_k]_{=n} \models \exists \phi \theta = \text{tt})\). Then:

- By Items 2.\((d)\)–2.\((c)\) in Definition 10, each state in \([s_k]_{=n}\) has a successor in \([s_{k+1}]_{=n}\).
- By Items 2.\((b)\)–2.\((c)\) in Definition 10, for all \(1 \leq i < k\), each state in \([s_i]_{=n}\) has a successor in \([s_{i+1}]_{=n}\).

Hence, we can build a run \(\rho(s_1)\) of \(H\) having a prefix \(s' \xrightarrow{\delta} s'' \xrightarrow{\gamma} s'''\) where \(s' \xrightarrow{\delta} s''\) traverses the classes \([s_1]_{=n}\) \(\ldots [s_i]_{=n}\) and \(s'' \in [s_{i+1}]_{=n+1}\). By our inductive hypothesis, \(\rho(s_1)\) witnesses \(s_1 \models \phi\).

- Let \(\phi = A\phi_1 U\phi_2\) and assume \([s_1]_{=n} \models \exists \phi \theta = \text{tt}\). Let \(\phi\) be a run departing from \(s_1\) in \(H\). Then, by Definition 11, \(H/_{=n}\) admits a path \(p_\rho = ([s_1]_{=n} \models \exists \phi \theta = \text{tt})\) i.e. such that \(\rho\) successively traverses the classes in \(p_\rho\). From Definition 13 and by our assumption \([s_1]_{=n} \models \exists \phi \theta = \text{tt}\), we obtain the existence of an index \(k\) for the path \(p_\rho\) such that:
  1. \(\forall i < k ([i]_{=n} \models \exists \phi \theta_1 \lor \phi_2 = \text{tt})\)
  2. \([s_k]_{=n} \models \exists \phi \theta_2 = \text{tt}\)

By our structural inductive hypothesis, we can conclude that the path \(p_\rho\) models the path formula \(\phi_1 U\phi_2\) and, by our arbitrary selection of \(p_\rho\), we get our thesis.

- Let \(\phi = E\phi_1 R_2\) and assume \([s_1]_{=n} \models \exists \phi \theta = \text{tt}\). Then, by Definition 13 we have that \([s_1]_{=n} \models \exists E\phi_2 U\phi_1 = \text{tt}\). Hence, using the results already proved for the operator \(EU\), we obtain \(s_1 \models E\phi_2 U\phi_1\) which implies \(s_1 \models E\phi_1 R_2\).

- Let \(\phi = A\phi_1 R_2\) and assume \([s_1]_{=n} \models \exists \phi \theta = \text{tt}\) i.e. \([s_1]_{=n} \models \exists E\phi_1 U\phi_2 = \text{ff}\). Denoted \(\gamma_1 = \neg \phi_1, \gamma_2 = \neg \phi_2\), suppose by absurd that \(s_1 \models E\gamma_1 U\gamma_2\) and let \(\rho\) be a corresponding witnessing run where \(\rho(t) \models \gamma_2 \land \forall t' \leq t \rho(t') \models \gamma_1 \lor \gamma_2\). Let \(\langle [s_i]_{=n} \rangle_{1 \leq i \leq k}\) the path of \(H/_{=n}\) abstracting the prefix of \(\rho\) until time \(t\). Then, if we assume \([s_k]_{=n} \models \exists \phi \theta_2 = \text{ff}\) we get a contradiction to our inductive structural hypothesis, since since \(\rho(t) \in [s_k]_{=n} \land \rho(t) \models \gamma_2\). Hence, we must have \([s_k]_{=n} \models \exists \phi \theta_2 = \text{tt}\). By Definition 13, since \([s_1]_{=n} \models \exists E\gamma_1 U\gamma_2 = \text{ff}\), we obtain the existence of an index \(j \leq k\) such that \([s_j]_{=n} \models \exists \gamma_1 \land \gamma_2 = \text{tt}\). Let \(s \models \rho U[s_j]_{=n}\). Then \(s \models \gamma_1 \lor \gamma_2\) and we got a contradiction to our structural inductive hypothesis.

\[\square\]

**Proof of Theorem 3** Using a structural inductive argument on \(\phi\), we prove that: for all \([s]_{=n} \in Q_{=n}\):

\[\models [s]_{=n} \models \exists \phi \theta = b \in \{\text{tt, ff}\} \rightarrow [s]_{=n+1} \models \exists \phi \theta = b \in \{\text{tt, ff}\}\]  \hspace{1cm} (1)

We present here a complete proof only for the cases \(\phi := E\phi_1 U\phi_2\) \(A\phi_1 U\phi_2\) and \(b = \text{tt}\), since the other cases are either symmetrical or simpler.

- Assume \([s_1]_{=n} \models \exists A\phi_1 U\phi_2 = \text{tt}\). Let \(p = \langle [s_i]_{=n+1} \rangle_{i \in N}\) be a path of \(H/_{=n+1}\) departing from \([s_1]_{=n}\) and consider the corresponding (stuttering) path \(p' = \langle [s_i]_{=n} \rangle_{i \in N}\) in \(H/_{=n}\). Using our structural induction on \(\phi\) we obtain that \(p\) models the path formula \(\phi_1 U\phi_2\), and by our arbitrary choice of \(p\) we conclude that \([s_1]_{=n+1} \models \exists A\phi_1 U\phi_2 = \text{tt}\).
Assume \([s_1]_{\equiv_n} \models_{\top} E \phi_1 U \phi_2\) = tt. By Definition 13 and by the results in Theorem 2, we obtain that for all \(s \in [s_1]_{\equiv_n}\), there exists a run departing from \(s\), \(\rho\), and a time \(t\), such that:

- \(\rho(t) = \phi_2\).
- \(\forall t' \leq t(\rho(t')) = \phi_1 \lor \phi_2\).
- The prefix of \(\rho\) leading to \(\rho(t)\) traverses at most \(n\) locations in \(H\).

By Definitions 10, 11, as well as by our inductive structural hypothesis on \(\phi\), the above conditions are established also on each state \(s \in [s_1]_{\equiv_{n+1}}\) implying \([s_1]_{\equiv_{n+1}} \models_{\top} E \phi_1 U \phi_2\) = tt.

\[\square\]

**Proof of Theorem 4** Consider the node \(\alpha\) in the finite abstraction structure \(H/\equiv_n\). By Definition 13, \(\alpha \models \phi \iff [\alpha]_{\equiv_n} = \top\). This ensures that the subprocedure \(\text{PROG}\) in ALGO1 operates correctly in time \(O(|H/\equiv_n| * |\phi|)\) and space \(O(|H/\equiv_n|)\).

Hence, to get to our thesis we need only to consider the subprocedure \(\text{PROG}\) showing that:

1. \(\text{PROG}(H/\equiv_n, n, \phi)\), correctly assigns to each \([s]_{\equiv_n} \in H/\equiv_n\) the label \((\phi, tt, m)\) iff \([s]_{\equiv_n} \models_{\equiv_n} \phi\] = tt and for all \(m' < m\), \([s]_{\equiv_m} \models_{\equiv_m} \phi\] = \(\bot\).
2. \(\text{PROG}\) operates in time \(O(|H/\equiv_n| * |\phi|)\) and space \(O(|H/\equiv_n|)\).

Below, we proceed in proving the above two items.

1. We use an inductive argument on \(\phi \in \text{ECTL}\), and we present only a complete proof for the case \(\phi = E \phi_1 U \phi_2\), since the other cases are rather simple to be dealt with.

Let \(\phi = E \phi_1 U \phi_2\). By induction on the number of iterations of the for-loop at line (3.6), we show that:

\[\text{At the end of the } i\text{-th iteration of the loop at line (3.6) of ALGO1, the set } S_i \text{ contains each node } [s]_{\equiv_n} \text{ such that } [s]_{\equiv_1} \models_{\equiv_1} \phi = \top \land \forall j < i \left( [s]_{\equiv_j} \models_{\equiv_j} \phi = \bot \right)\]

A global correct labelling follows immediately from the above statement since the for-loop at lines (3.14)–(3.15) assigns to each node in \(S_i\), \(i = 0 \ldots n\), the label \((\phi, pp, i)\)

**Base** (\(i = 0\)) We should prove that before entering for the first time the loop at line (3.6), each class \([s]_{\equiv_n}\) such that \([s]_{\equiv_n} \models_{\equiv_n} \phi = \top\) is assigned to \(S_0\). Consider \([s]_{\equiv_n}\) and assume \([s]_{\equiv_n} \models_{\equiv_n} \phi = \top\). Then, by Definition 13, and by Theorem 3 \(H/\equiv_n\) admits a path \(\langle [s_i]_{\equiv_n} \rangle_{1 \leq i \leq k}\) such that for all \(1 \leq i < k\), \(\langle [s_i]_{\equiv_n} \rangle_{1 \leq i \leq k}\) and \([s]_{\equiv_n} \models_{\equiv_n} \phi = \top\). Assuming by structural induction that ALGO1 performs a correct labelling for the subformule \(\phi_2, \phi_1 \lor \phi_2\), we obtain that \([s]_{\equiv_n}\) is inserted in the set \(S_0\) at line (2.5).

**Inductive Step** (\(n \leq i > 0\)) Let \([s]_{\equiv_n} \in H/\equiv_n\), and assume that \([s]_{\equiv_n}\) such that \([s]_{\equiv_n} \models_{\equiv_n} \phi = \top \land \forall j < i \left( [s]_{\equiv_j} \models_{\equiv_j} \phi = \bot \right)\). Then, by Definition 13, and by Theorem 3 \(H/\equiv_n\) admits a path \(\langle [s_i]_{\equiv_n} \rangle_{1 \leq i \leq k}\) such that for all \(1 \leq i < k\):

(a) \(\langle [s_i]_{\equiv_n} \rangle_{1 \leq i \leq k}\) and \(\langle [s_{i+1}]_{\equiv_n} \rangle_{1 \leq i \leq k}\)

(b) \([s_i]_{\equiv_n} \models_{\equiv_n} \phi = \top \land [s_{i+1} \models_{\equiv_n} \phi = \top]\)
(c) \( (\forall j < i; ([s_k]_i \models 3\phi_2) = \bot) \lor (\forall j < i - 1; ([s_k+1]_i \models 3\phi) = \bot) \lor (\exists l < k; (\forall j < i; ([s_l]_i \models 3\phi_1) = \bot)) \)

By our inductive hypothesis and assuming a correct labelling for the subformulæ \( \phi_1, \phi_2 \), the following applies:

- In case the first branch of condition (c) is satisfied, \([s]_i \models 0 \rightarrow \alpha \in A_2[i] \) and all internal nodes are assigned the label \( \langle \phi_1 \lor \phi_2, \text{tt}, - \rangle \).
- Otherwise, if second branch of condition (c) is satisfied, \([s]_i \models 1 \leftarrow \alpha \in S_{i-1} \) and all internal nodes are assigned the label \( \langle \phi_1 \lor \phi_2, \text{tt}, - \rangle \).
- Otherwise, if only the third branch of condition (c) is satisfied, \([s]_i \models 0 \rightarrow [p]_i \models a \rightarrow [q]_i \), where \([p]_i \in A_1[i], [q]_i \in S_{j < i} \).

Hence, the body of the loop at line (3.6) correctly assigns \([s]_i \) to \( S_i \).

2. Our claim for the complexity of \textsc{Processe} follows easily once we prove that processing the case \( \phi = E\phi_1 U \phi_2 \) requires \( O(|H_{un}|) \) time and space.

The initialization phase for the case \( \phi = E\phi_1 U \phi_2 \) in \textsc{Processe} (lines (3.1)-(3.3)) requires \( O(|H_{un}|) \) time and space since the disjoint union of the lists in each slot of the array \( A_1 \) (resp. \( A_2 \)) gives \( Q_{un} \). The loop at line (3.6) can be performed in time and space \( O(|H_{un}|) \). In fact, consider a given iteration \( i \) of such a loop.

Using a breadth first like algorithm\(^3\) it is possible to complete \( S_i \) with a number of steps bounded by:

\[
2(|\text{Adj}_{\beta}^{-1}(S_i \cup S_{i-1})| + |\text{Adj}_{e}^{-1}(S_i \cup S_{i-1})|)
\]

Since \( \forall 1 \leq i \leq n, j < i; (S_i \cap S_j = \emptyset) \), the global cost of the loop at line (3.6) is

\[
\sum_{i \leq n} |\text{Adj}_{\beta}^{-1}(S_i \cup S_{i-1})| + |\text{Adj}_{e}^{-1}(S_i \cup S_{i-1})| \leq 4|\text{Adj}(Q_{un})| = O(|H_{un}|)
\]

**Proof of Lemma 4** It follows immediately by the fact that each internal node has at least two immediate successors, and the number of leaves is \( O(|Q_{un}|) \). \(\square\)

**Proof of Theorem 5** The thesis follows immediately if we prove that the subprocedure \textsc{Processe} (resp. \textsc{Processe}, \textsc{Processo}, \textsc{Processa}):

1. Assigns to \([s]_i \) in \( H_{un} \) the label \( \langle \phi, \text{tt}, m \rangle \), where \( \phi = E\phi_1 U \phi_2 \) (resp. \( \phi = A\phi_1 U \phi_2 \), \( \phi = E\phi_1 R \phi_2, \phi = A\phi_1 R \phi_2 \)) if and only if \([s]_m \models 3\phi \) = \( \text{tt} \land \forall k < m; ([s]_k \models 3\phi) = \bot \).
2. Operates in time and space \( O(|H_{un}|) \)

We consider here only the subprocedures \textsc{Processe} and \textsc{Processa}, since the other cases are either symmetrical or simpler. As far as \textsc{Processe} is concerned, the two items above can be proved using the same circle of arguments provided for the processing of the operator \( \textsc{E} \) within \textsc{Processo} in \textsc{algo}.

\(^3\) where the set of initial nodes is \( S_{i-1} \cup \{ \alpha \mid \alpha \in A_2[i] \land \text{color}(\alpha) \neq \text{red} \} \cup \{ \alpha \mid \alpha \in A_1[i] \land \text{color}(\alpha) = \text{yellow} \} \) and green nodes are discovered and red-colored following \( \rightarrow \).
The complexity of PROCESSAU follows immediately from Lemma 4 and from the complexity of classical two-valued model checking. As far as the correctness of the procedure PROCESSAU is concerned, assume a labelling of $H_{/=n}$ by means of the labels $\langle \phi_1, \text{tt}, - \rangle$, $\langle \phi_1, \text{tt}, - \rangle$. Then, by Definition 13, $[[s]|_n = A\phi_1 U\phi_2] = \text{tt}$ if and only if $[[s]|_n = A\phi_1 U\phi_2]$ according to classical two-valued model checking and to the given labelling. Hence, if a node is assigned the label $\langle \phi, \text{tt}, n \rangle$ at lines (1.1)–(1.3) in procedure PROCESSAU then the first two components of the label are correct. By Theorem 3, if $[[s]|_{m,n} = A\phi_1 U\phi_2] = \text{tt}$, then for all $[s]_{m,n} \subseteq [s]_{m,n}$, $[[s]|_{m,n} = A\phi_1 U\phi_2] = \text{tt}$. Hence, lines (1.4)–(1.7) in procedure PROCESSAU, correctly overwrite the third component in the label $\langle \phi_1, \text{tt}, n \rangle$, for each node of $H_{/=n}$. □