

Proving the existence of fair paths in infinite-state systems

Alessandro Cimatti¹, Alberto Griggio¹, and Enrico Magnago¹

Fondazione Bruno Kessler, Trento, Italy
{cimatti,griggio,magnago}@fbk.eu

Abstract. In finite-state systems, true existential properties admit witnesses in form of lasso-shaped fair paths. When dealing with the infinite-state case (e.g. software non-termination, model checking of hybrid automata) this is no longer the case. In this paper, we propose a compositional approach for proving the existence of fair paths of infinite-state systems. First, we describe a formal approach to prove the existence of a non-empty under-approximation of the original system that only contains fair paths. Second, we define an automated procedure that, given a set of hints (in form of basic components), searches for a suitable composition proving the existence of a fair path. We experimentally evaluate the approach on examples taken from both software and hybrid systems, showing its wide applicability and expressiveness.

1 Introduction

LTL model checking for infinite-state systems is a well-known undecidable problem. Most of the research has concentrated on proving that the properties are universally verified, i.e. all traces satisfy the property. In this work, we focus on its dual problem: the falsification of LTL properties, which amounts to proving that one trace satisfies (the negation of) the property. Notable instances of this problem are proving software non-termination (with the fair path to be found corresponding to a non-terminating execution) and finding counterexamples and scenarios in hybrid systems and in infinite-state fair transition systems. Model checking can be reduced to proving the language emptiness of an infinite-state fair transition system. In order to prove that the LTL property does not hold it is necessary and sufficient to prove the existence of a fair infinite execution.

The problem is conceptually harder than in the finite-state case, since fair paths may have no regular structure. Hence, in general they cannot be presented in lasso-shaped form as $\alpha \cdot \beta^\omega$, where α and β are finite sequences of states and β^ω is the infinite repetition of β .

In this paper, we propose an approach to prove the existence of fair paths in infinite-state fair transition systems. The approach is based on the following insights. We define an underapproximation of the given transition system, extended with formulae describing regions of the state space of the system, which we call \mathcal{R} -abstraction. We identify a set of conditions over the underapproximation that are sufficient for the existence of a fair path. Such abstraction enjoys

the property that each fair loop over its regions entails the existence of a fair path in the original system. In this sense, each lasso-shaped execution over the regions represents a non-empty envelope containing only fair paths of the original system. We formally present the hypotheses necessary for the \mathcal{R} -abstraction to represent a suitable non-empty under-approximation of the fair transition system. This argument, based on a monolithic underapproximation, is refined into a compositional approach. Intuitively, the monolithic underapproximation is presented as the composition of smaller transition systems enriched with a set of regions and assumptions. We define a set of conditions that, if satisfied by the components, entail that the composition proves the existence of the fair path.

Based on this framework, we describe a search procedure to identify a compositional presentation of the under-approximation. The procedure takes in input a candidate set of components, and looks for a suitable composition of a subset of them that represents an adequate under-approximation of the original system.

We study a generalization to enforce the divergence of a specific symbol. This is required, for example, to deal with conditions resulting from the conversion of hybrid systems into fair transition systems, and the analysis is to be restricted to non-zeno paths, where time diverges to infinity.

We implemented and evaluated the proposed approach. The procedure works on symbolically represented infinite-state fair transition systems, and is able to produce suitable compositions and to exhibit proofs of existence of fair paths based on manual hints produced with moderate effort. The results, obtained for benchmarks of diverse nature, derived from software termination and hybrid automata, demonstrate the expressiveness of the framework and the effectiveness of the approach.

The paper is organised as follows. In Sec. 2 and 3 we present the background and a running example. In Sec. 4 and 5, we define the monolithic and compositional frameworks. The search procedure is described in Sec. 6. In Sec. 7 we discuss symbol-divergence. In Sec. 8 we contrast our approach with related work. Sec. 9 reports the experimental evaluation of the approach. Sec. 10 concludes and outlines future works. The proofs of all the theorems are reported in the extended version of this document¹.

2 Background

We work in the setting of SMT, with the theory of quantified real arithmetic. We assume the standard notions of interpretation, model, satisfiability, validity and logical consequence. We write $\text{NNF}(\phi)$ for the negation normal form of ϕ . A symbolic fair transition system M is a tuple $\langle S, I, T, F \rangle$, where S is the set of state variables; I and F are formulae over S , representing respectively the initial and fair states; T is a formula over S and S' representing the transitions, where $S' \doteq \{s' \mid s \in S\}$ and the primed version of a variable refers to the next state. We denote with \mathbf{S} or \mathbf{s} a total assignment over S , i.e. a state. A fair path of M is

¹ The extended version is available at https://enricomagnago.com/proving_the_existence_of_fair_paths_in_infinite-state_systems_extended.pdf

an infinite sequence of states, s_0, s_1, \dots , such that $s_0 \models I$, $s_i s'_{i+1} \models T$ for all i , and for each i there exists $j > i$ such that $s_j \models F$. A deadlock is a reachable state that has no outgoing transitions.

We also assume the standard notions of trace, reachability, and temporal logic model checking, using **E**, **A** for path quantifiers and **G**, **F** for “always” and “eventually” (CTL* [17]).

We overload the \models symbol: when ϕ and ψ are SMT formulae, then $\phi \models \psi$ stands for entailment in SMT; when M is a fair transition system and ψ is a temporal property, then $M \models \psi$ is to be interpreted with the LTL semantics.

3 Running example

For explanatory purposes, we consider a bouncing ball subject to the gravitational acceleration. The ball follows the classical laws of a uniformly accelerated motion, losing a fraction of its velocity at every bounce. The bounce is an instantaneous transition where $v' = -v \frac{c}{c+1}$, with v being the velocity and c the number of bounces. It is also possible for the ball to get stuck to the ground. Let h be the distance of the ball from the ground. The dynamics are partitioned into three phases: in the first phase, the ball is falling down ($v < 0 \wedge h > 0$); in the second, the ball is bouncing $h = 0$; finally, the ball is moving upwards ($v > 0 \wedge h > 0$). Unless stopped, the ball goes infinitely through the phases, but for shorter and shorter periods of time: the interval between two consecutive bounces c and $c + 1$ is given by $\frac{1}{c}$.

```

1 MODULE main
2   VAR h : real; v : real; delta : real; c : integer;
3     stop : boolean;
4   DEFINE g := 9.81;
5   INIT c = 1 & h = 0 & v = g / 2;
6   INVAR (h = 0 & v < 0) -> delta = 0;
7   INVAR delta >= 0 & h >= 0;
8   TRANS (h = 0 & v < 0) -> next(c) = c + 1;
9   TRANS !(h = 0 & v < 0) -> next(c) = c;
10  TRANS (stop & h = 0) -> (next(h) = 0 & next(v) = 0);
11  TRANS (!stop & h = 0 & v = 0) -> (next(h) = 0 & next(v) = 0);
12  TRANS (!stop & h = 0 & v < 0) -> (next(h) = 0 &
13                                     next(v) = - v*c / (c + 1));
14  TRANS (!stop & !(h = 0 & v <= 0)) -> (next(v) = v - g*delta &
15                                     next(h) = h + v * delta - 0.5 * g * pow(delta, 2));
16  LTLSPEC ! G F (h = 0 & v > 0);

```

Fig. 1. The SMV encoding of the bouncing ball

Systems like the bouncing ball are usually described as hybrid systems. Here we consider the corresponding infinite state transition system, presented symbolically in Figure 1 using a variant of the SMV language. The symbol `delta` represents the amount of time elapsing at each transition. The nondeterministic `stop` variable controls the ball getting stuck to the ground. The (universal) property states that the ball cannot bounce up infinitely often.

4 Fair Paths: Sufficient conditions

This section presents the main argument used to prove the existence of a fair path for a fair transition system M . First, we identify a transition system, A , organized according to a set of regions \mathcal{R} , each corresponding to a location. Then, we show that if A is a non-empty under-approximation of M and satisfies some conditions, then the existence of a fair cycle in M is ensured. When clear from context, with a slight abuse of notation, we write \mathcal{R} for the formula $\bigvee_{i=0}^{m-1} R_i$ denoting the region space.

We call A an \mathcal{R} -abstraction with respect to a system M and regions \mathcal{R} if the following conditions hold. The region space \mathcal{R} must be reachable in M - intuitively, this corresponds to finding the “stem” of the fair path. A must be an underapproximation of M , so that the transitions taken in a path in A can be performed also in M . A must never deadlock in \mathcal{R} , and there must be no outgoing transitions from \mathcal{R} , so that from every state in \mathcal{R} there exists an infinite path starting from it and contained in \mathcal{R} . Finally, we require that the set of fair locations F_A is visited infinitely often. These conditions are formalised in the following definition.

Definition 1 (\mathcal{R} -abstraction). Let $M \triangleq \langle S_M, I_M(S_M), T_M(S_M, S'_M), F_M(S_M) \rangle$ be a fair transition system. A transition system $A \triangleq \langle S_A, I_A(S_A), T_A(S_A, S'_A) \rangle$ is an \mathcal{R} -abstraction of M with respect to a list of formulae $\mathcal{R}(S_A) \triangleq [R_0(S_A), \dots, R_{m-1}(S_A)]$, also called regions, iff the following hold:

H.0 $S_M \subseteq S_A$,

H.1 There exists some initial state in M from which it is possible to reach an initial state of A , for some assignment to the $S_A \setminus S_M$:

$$M \not\models \mathbf{AG} \neg I_A(S_A)$$

H.2 The set of initial states of A is a subset of the union of the regions:

$$A \models \mathcal{R}(S_A)$$

H.3 The transition relation of A underapproximates the transition relation of M :

$$\mathcal{R}(S_A) \wedge T_A(S_A, S'_A) \models T_M(S_M, S'_M)$$

H.4 Every state in R_0 , projected over the symbols in S_M corresponds to a fair state of M :

$$A \models \mathbf{AG}(R_0(S_A) \rightarrow F_M(S_M))$$

H.5 Every reachable state in A has at least one successor via its transition relation T_A :

$$A \models \mathbf{AGEX} \top$$

H.6 For each region $R_i \in \mathcal{R}$, with $i > 0$, every state in R_i can remain in such region at most a finite number of steps and must eventually reach a region with a lower index $j < i$:

$$A \models \bigwedge_{i=1}^{m-1} \mathbf{AG}(R_i \rightarrow \mathbf{A}[R_i \mathbf{U} \bigvee_{j=0}^{i-1} R_j])$$

H.7 All states reachable in one step from R_0 are in \mathcal{R} :

$$A \models \mathbf{AG}(R_0 \rightarrow \mathbf{AX} \bigvee_{i=0}^{m-1} R_i)$$

In order to prove the existence of a fair path in M , we seek a \mathcal{R} -abstraction A . This is sufficient since, as shown by the following theorem, *all* paths of A are fair paths in M .

Theorem 1. Let $M \doteq \langle S_M, I_M, T_M, F_M \rangle$ be a fair transition system. Let A be an \mathcal{R} -abstraction of M with respect to a sequence of regions \mathcal{R} over S . Then M admits a fair path, i.e. $M \not\models \mathbf{FG} \neg F_M(S_M)$, and all infinite paths of A starting from some state reachable in M correspond to a fair path of M .

Example Consider the LTL model checking problem defined in Figure 1 and let $M \doteq \langle S^M, I^M, T^M, F^M \rangle$ be the fair transition system whose fair executions are counterexamples for the LTL property. Then, I^M and T^M are defined as in the system described in Figure 1 and the fairness condition is $F^M \doteq h = 0 \wedge v > 0$. Figure 2 shows a possible \mathcal{R} -abstraction A for M that proves the existence of at least one counterexample for the LTL property. A has two regions R_0 and R_1 . The transition relation T_A is shown with annotations on the edges connecting the two locations. In both regions the ball is on the ground ($h = 0$), but its velocity is negative in R_1 and positive in R_0 , hence the latter is fair. More formally, A is defined as $\langle S, R_0 \vee R_1, T_A \rangle$ over two regions $\{R_0, R_1\}$, where:

$$R_0 \doteq c \geq 1 \wedge \text{delta} = \frac{1}{c} \wedge v = \frac{g}{2c} \wedge h = 0$$

$$R_1 \doteq c \geq 1 \wedge \text{delta} = 0 \wedge v = -\frac{g}{2c} \wedge h = 0$$

$$T_{0,1} \doteq c' = c \wedge \text{delta}' = 0 \wedge v' = v - g * \text{delta} \wedge h' = h$$

$$T_{1,0} \doteq c' = c + 1 \wedge \text{delta}' = \frac{1}{c+1} \wedge v' = -v \frac{c}{c+1} \wedge h' = h$$

$$T_A \doteq \bigvee_{i \in \{0,1\}} (R_i \wedge T_{i,1-i} \wedge R_{1-i}')$$

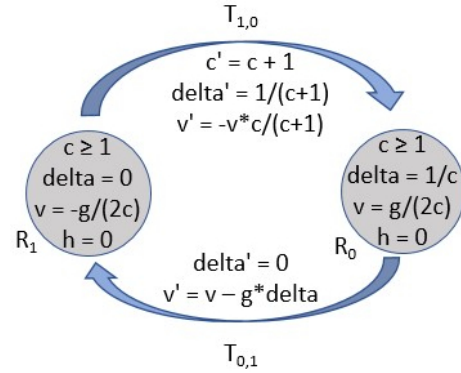


Fig. 2. \mathcal{R} -abstraction for the running example.

It is easy to see that A is an \mathcal{R} -abstraction and satisfies all required hypotheses of Definition 1.

4.1 Comparison with recurrent sets

In the context of software non-termination the notion of recurrent set has been introduced by Gupta et al. in [25]. They show that the existence of an (open) recurrent set is a sufficient and necessary condition for a not well-founded relation. Cook et al. in [9] introduce the notion of closed recurrence sets, which is used also in [14]. Closed recurrence sets, instead of characterising a set of states that contain some infinite sequence, require the existence of at least one sequence in the set and that every sequence remaining in such set is infinite. In the same work they show that every closed recurrence set is also an open recurrent set and that if a open recurrent set exists, then there exists a corresponding closed recurrence set for some underapproximation of the transition relation.

These works are concerned with software non-termination and do not consider fairness conditions. Since, as we show below, an \mathcal{R} -abstraction corresponds to a closed recurrent set when the fairness condition is trivial (i.e. \top), our notion of \mathcal{R} -abstraction is strictly more expressive than what considered in the works above.

A not well-founded relation exists iff there exists an open recurrent set [25]. Cook et al. [14] show that if a system admits some recurrent set then there exist an underapproximation of it that admits a closed recurrence set. Therefore, Theorem 2 below implies that a not well-founded relation exists, for a system with trivial fairness, iff it admits an \mathcal{R} -abstraction.

We report the definition of closed recurrence set from [9], where we explicitly state that we are interested in an underapproximation of the transition relation. A set \mathcal{G} is a closed recurrence set for a transition system $M \doteq \langle S, I(S), T(S, S') \rangle$, with respect to some underapproximation $T_{\mathcal{G}}$ of T iff the following hold:

$$\begin{aligned} \exists S : \mathcal{G}(S) \wedge I(S) \\ \forall S \exists S' : \mathcal{G}(S) \rightarrow T_{\mathcal{G}}(S, S') \\ \forall S, S' : \mathcal{G}(S) \wedge T_{\mathcal{G}}(S, S') \rightarrow \mathcal{G}(S') \\ \forall S, S' : T_{\mathcal{G}}(S, S') \rightarrow T(S, S') \end{aligned}$$

Theorem 2. *A system $M \langle S, I(S), T(S, S'), \top \rangle$ admits an \mathcal{R} -abstraction $A \doteq \langle S_A, I_A(S_A), T_A(S_A, S'_A) \rangle$ if and only if there exists a closed recurrence set \mathcal{G} .*

5 Decomposition of the sufficient conditions

Finding a monolithic \mathcal{R} -abstraction satisfying all the hypotheses for some fair transition system is a challenging problem. Here we refine the framework to present the \mathcal{R} -abstraction compositionally, as a network of smaller components,

by considering a subset of the symbols at a time. For each subset of the symbols we identify some smaller components that could represent the behaviour of the system projected only on those variables. The monolithic \mathcal{R} -abstraction is the composition of these smaller components, one for each subset of variables. We describe the interaction between the components in an assume-guarantee fashion. Each component, that we call \mathcal{AG} -skeleton (for Assume-Guarantee skeleton), describes the behaviour of a subset of the symbols while assuming some properties about the others. These properties represent the conditions that are necessary for this behaviour to be enabled and we need to prove that such conditions are ensured by some other \mathcal{AG} -skeleton.

The following is the outline of the approach. We first formally define \mathcal{AG} -skeletons and a composition operator over such structures. In order to find an \mathcal{R} -abstraction, given a set of \mathcal{AG} -skeletons we apply such operator until we obtain a composed \mathcal{AG} -skeleton with an empty set of assumptions, which, by definition of the composition operator, implies that we considered one \mathcal{AG} -skeleton for each subset of the symbols. This \mathcal{AG} -skeleton is a transition system associated with a list of regions that either does not allow any loop over the regions or satisfies hypotheses H.5, H.6 and H.7. Among all possible compositions the procedure described in Sec. 6 searches for one that admits at least one such loop that also satisfies H.1, H.2, H.3 and H.4, hence an \mathcal{R} -abstraction.

Formally, let M be given. Let $\{S^0, \dots, S^{n-1}\}$ be pairwise disjoint and a covering² of S^M . Let $\{H^j\}_{j=0}^{n-1}$ be a set of transition systems of the form $\langle S^j \cup S^{\neq j}, I^j, T^j \rangle$ and $m^j \in \mathbb{N}$ be the number of regions of H^j . We say that S^j are the symbols controlled by H^j or its local symbols. We also write S for $\bigcup_{j=0}^{n-1} S^j$ and $S^{\neq i}$ for $S \setminus S^i$. Let $\mathcal{R}^j \doteq \{R_i^j(S) \mid 0 \leq i < m^j\}$ be the set of regions of H^j and $\mathcal{A}^j \doteq \{A_i^j(S^{\neq j}) \mid 0 \leq i < m^j\}$ the set of assumptions of H^j . Let $\mathcal{A}_i^j(S^{\neq j})$ be the assumptions of H^j in its i^{th} region on the other components. We assume such assumptions are in *cartesian* form, by requiring

$$A_i^j(S^{\neq j}) \doteq \bigwedge_{k \neq j} A_i^{j,k}(S^k)$$

where $A_i^{j,k}(S^k)$ are (independent) assumptions on H^k of H^j in the i^{th} region.

Notice that the regions $R_i^j(S)$ of H^j can depend on all the variables S , while the assumptions $A_i^j(S^{\neq j})$ cannot refer to the “local variables” S^j of H^j . The *restricted region* i of H^j is $(R_i^j \wedge A_i^j)$.

Every \mathcal{AG} -skeleton H^j must satisfy the following condition.

- I .** If there is pair of states satisfying the transition relation, such that the first one is in the restricted region i and the latter in the restricted region i' , then for every state in the restricted region i the transition relation allows for a

² Hence, $S^M \subseteq \bigcup_j S^j$ and $\forall j \neq k : S^j \cap S^k = \emptyset$

successor state in the restricted region i' :

$$\begin{aligned} \forall i, i' : 0 \leq i < m^j \wedge 0 \leq i' < m^j &\rightarrow \\ \exists S, S' : (R_i^j(S) \wedge A_i^j(S^{\neq j}) \wedge T^j(S, S') \wedge R_{i'}^j(S') \wedge A_{i'}^j(S^{\neq j'})) &\models \\ \forall S \exists S^{j'} \forall S^{\neq j'} : R_i^j(S) \wedge A_i^j(S^{\neq j}) \wedge A_{i'}^j(S^{\neq j'}) &\rightarrow R_{i'}^j(S') \wedge T^j(S, S') \end{aligned}$$

This is related to *must*-abstractions, presented for example in [36], in the sense that for every assignment to the current state symbols S there must exist an assignment to the next state symbols. However, in our case we restrict the existential quantification only to the symbols local to the \mathcal{AG} -skeleton $S^{j'}$.

Definition 2 (compatible transitions). Let $\{H^{j_0}, \dots, H^{j_k}\} \subseteq \{H^i\}_{i=0}^{n-1}$ be a subset of the \mathcal{AG} -skeletons. A transition from state \hat{S} to \hat{S}' is compatible iff the transitions of the \mathcal{AG} -skeletons, from every pair of states in the same regions, meet the respective assumptions of the \mathcal{AG} -skeletons.

$$\begin{aligned} \text{compatible}_{\{j_0, \dots, j_k\}}(\hat{S}, \hat{S}') \doteq \forall S, S' : & \bigwedge_{0 \leq i_0 < m^{j_0}, 0 \leq i'_0 < m^{j_0}, \dots, 0 \leq i_k < m^{j_k}, 0 \leq i'_k < m^{j_k}} \\ (R_{i_0}^{j_0}(\hat{S}) \wedge A_{i_0}^{j_0}(\hat{S}) \wedge R_{i'_0}^{j_0}(\hat{S}') \wedge A_{i'_0}^{j_0}(\hat{S}') \wedge \dots \wedge R_{i_k}^{j_k}(\hat{S}) \wedge A_{i_k}^{j_k}(\hat{S}) \wedge R_{i'_k}^{j_k}(\hat{S}') \wedge A_{i'_k}^{j_k}(\hat{S}') \rightarrow & \\ \bigwedge_{0 \leq t \leq k} ((R_{i_t}^t(S) \wedge A_{i_t}^t(S^{\neq j_t}) \wedge A_{i'_t}^t(S^{\neq \{j_s\}_{s=1}^k}) \wedge & \\ \bigwedge_{0 \leq s \leq k \wedge s \neq t} T^s(S, S') \wedge R_{i'_s}^s(S') \wedge A_{i'_s}^s(S^{\neq j_s'})) \rightarrow \bigwedge_{0 \leq h \leq k \wedge h \neq t} A_{i'_t}^{t, j_h}(S^{j_h'}) & \\) & \\) & \end{aligned}$$

Compatible holds iff the existence of a transition from some state \hat{S} to \hat{S}' in the intersection of some restricted regions, implies that every transition between the same intersection of restricted regions implies that the assumptions made by each \mathcal{AG} -skeleton are met.

We now define the composition of \mathcal{AG} -skeletons as the standard product of transition systems restricted to the *compatible* transitions and show that this operation is closed: the composition of k \mathcal{AG} -skeletons is an \mathcal{AG} -skeleton.

Definition 3 (composition of \mathcal{AG} -skeletons). We define the composition of $\{H^{j_0}, \dots, H^{j_k}\} \subseteq \{H^j\}_{j=0}^{n-1}$, such that the sets of local symbols $\{S^{j_i}\}_{i=0}^k$ are pairwise disjoint, as $H^c \doteq \bigotimes_{t=0}^k H^{j_t} = \langle S, I^c, T^c \rangle$ where:

- $S^c \doteq \bigcup_{t=0}^k S^{j_t}$;
- $m^c \doteq \prod_{t=0}^k m^{j_t}$;
- $\mathcal{R}^c \doteq \{\bigwedge_{t=0}^k R_{i_t}^{j_t}(S) \wedge \bigwedge_{0 \leq s \leq k \wedge s \neq t} A_{i_t}^{j_t, j_s}(S^{j_s}) \mid \forall t \in \{0, \dots, k\}, i_t \in \{0, \dots, m^{j_t} - 1\} : R_{i_t}^{j_t}(S) \in \mathcal{R}^{j_t} \text{ and } \forall s . 0 \leq s \leq k \wedge s \neq t : A_{i_t}^{j_t, j_s}(S^{j_s}) \in \mathcal{A}^{j_t}\}$;
- $\mathcal{A}^c \doteq \{\bigwedge_{t=0}^k \bigwedge_{j_s \notin \{j_0, \dots, j_k\}} A_{i_t}^{j_t, j_s}(S^{j_s}) \mid \forall t \in \{0, \dots, k\}, j_s \notin \{j_0, \dots, j_k\}, i_t \in \{0, \dots, m^{j_t} - 1\} : A_{i_t}^{j_t, j_s}(S^{j_s}) \in \mathcal{A}_{i_t}^{j_t}(S^{\neq j_s})\}$;

- $I^c(S) \doteq \bigwedge_{t=0}^k I^{j_t}(S)$;
- $T^c(S, S') \doteq \text{compatible}_{j_0, \dots, j_k}(S, S') \wedge \bigwedge_{t=0}^k T^{j_t}(S, S')$.

For compactness we will use $S^{\neq c}$ for $S \setminus S^c$ which is equal also to $\bigcup_{t=0}^k S^{\neq j_t}$ and $A_i^j(S^{\neq c})$ for $\bigwedge_{j_s \notin \{j_0, \dots, j_k\}} A_i^{j_s}(S^{j_s})$.

Theorem 3 (\mathcal{AG} -skeletons are closed under \otimes). *Given a set of \mathcal{AG} -skeletons $\{H^{j_0}, \dots, H^{j_k}\} \subseteq \{H^j\}_{j=0}^{n-1}$, their composition $H^c \doteq \bigotimes_{t=0}^k H^{j_t} = \langle S, I^c, T^c \rangle$ is still an \mathcal{AG} -skeleton, i.e. it satisfies hypothesis [I](#).*

By composing a sequence of \mathcal{AG} -skeletons such that their local symbols are pairwise disjoint and cover the set of symbols S^M of the fair transition system M , we obtain an \mathcal{AG} -skeleton with an empty set of assumptions. By definition, the composition satisfies [I](#): every pair of regions either do not admit any transition between them or from one it is always possible to reach the other in one step and there is no deadlock. Therefore, such \mathcal{AG} -skeleton is a transition system associated with a list of regions such that [H.5](#) and [H.6](#) hold, and, in case a region is a subset of the fair states of M , also [H.7](#) holds. In the next section we describe a procedure that (i) computes such a composition of \mathcal{AG} -skeletons, and (ii) among all possible compositions it looks for one that admits some loop over the regions satisfying also the remaining hypotheses ([H.1](#), [H.2](#), [H.3](#) and [H.4](#)), thus ensuring that it is an \mathcal{R} -abstraction.

5.1 Example: decomposition

In the following, for compactness, we write R^j for $R^j(S)$, A^j for $A^j(S^{\neq j})$, T^j for $T^j(S, S')$ and $R^{j'}$, $A^{j'}$ for $R^j(S')$ and $A^j(S^{\neq j'})$ respectively. We now show how the \mathcal{R} -abstraction in [Figure 2](#) can be represented as composition of smaller \mathcal{AG} -skeletons. Consider the partitioning of S given by $S^C \doteq \{c\}$, $S^H \doteq \{h\}$ and $S^{DV} \doteq \{d, v\}$. We define three corresponding \mathcal{AG} -skeletons:

$$C \doteq \langle S^C, c \geq 1, c' = c + 1 \vee c' = c \rangle$$

with no assumptions and a single region $c \geq 1$.

$$\begin{aligned} H \doteq & \langle S^H, (R_0^H \wedge A_0^H) \vee (R_1^H \wedge A_1^H), \\ & (R_0^H \wedge A_0^H \wedge T_{0,0}^H \wedge R_0^{H'} \wedge A_0^{H'}) \vee \\ & (R_0^H \wedge A_0^H \wedge T_{0,1}^H \wedge R_1^{H'} \wedge A_1^{H'}) \vee \\ & (R_1^H \wedge A_1^H \wedge (T_{1,0,0}^H \vee T_{1,0,1}^H) \wedge R_0^{H'} \wedge A_0^{H'}) \rangle \end{aligned}$$

where $R_0^H \equiv R_1^H \doteq h = 0$, $A_0^H \doteq \text{delta} = 0$, $A_1^H \doteq \text{delta} = \frac{2v}{g}$, $T_{0,0}^H \equiv T_{0,1}^H \equiv T_{1,0,0}^H \doteq h' = h$ and $T_{1,0,1}^H \doteq h' = h + v * \text{delta} - \frac{g}{2} \text{delta}^2$. Finally we define

$$\begin{aligned} DV \doteq & \langle S^{DV}, (R_0^{DV} \wedge A_0^{DV}) \vee (R_1^{DV} \wedge A_1^{DV}), \\ & (R_0^{DV} \wedge A_0^{DV} \wedge T_{0,1}^{DV} \wedge R_1^{DV'} \wedge A_1^{DV'}) \vee \\ & (R_1^{DV} \wedge A_1^{DV} \wedge T_{1,0}^{DV} \wedge R_0^{DV'} \wedge A_0^{DV'}) \rangle \end{aligned}$$

where $R_0^{DV} \doteq \text{delta} = 0 \wedge v = -\frac{g}{2c}$ and $R_1^{DV} \doteq \text{delta} = \frac{1}{c} \wedge v = \frac{g}{2c}$, the two assumptions are $A_0^{DV} \equiv A_1^{DV} = c \geq 1 \wedge h = 0$ and the two components of the transition relation are defined as $T_{0,1}^{DV} \doteq \text{delta}' = \frac{1}{c+1} \wedge v' = -v\frac{c}{c+1}$ and $T_{1,0}^{DV} \doteq \text{delta}' = 0 \wedge v' = v - g * \text{delta}$.

The three \mathcal{AG} -skeletons satisfy [1](#). Applying the composition operator and removing empty regions and transitions we obtain

$$B \doteq C \otimes DV \otimes H = \langle S^B, R_0^B \vee R_1^B, (R_0^B \wedge T_{0,1}^B \wedge R_1^{B'}) \vee (R_1^B \wedge T_{1,0}^B \wedge R_0^{B'}) \rangle$$

with two regions $\{R_0^B, R_1^B\}$ and no assumptions, where:

$$\begin{aligned} R_0^B &\doteq c \geq 1 \wedge \text{delta} = \frac{1}{c} \wedge v = \frac{g}{2c} \wedge h = 0 \\ R_1^B &\doteq c \geq 1 \wedge \text{delta} = 0 \wedge v = -\frac{g}{2c} \wedge h = 0 \\ T_{0,1}^B &\doteq c' = c \wedge \text{delta}' = 0 \wedge v' = v - g * \text{delta} \wedge h' = h \\ T_{1,0}^B &\doteq c' = c + 1 \wedge \text{delta}' = \frac{1}{c+1} \wedge v' = -v\frac{c}{c+1} \wedge h' = h \end{aligned}$$

Region R_0^B implies the fairness condition $h = 0 \wedge v > 0$ and we obtain the \mathcal{R} -abstraction $\langle S, \{R_0^B, R_1^B\}, T^B \rangle$, where $T^B \doteq \bigvee_{i \in \{0,1\}} (R_i^B \wedge T_{i,1-i}^B \wedge R_{1-i}^{B'})$ which is exactly the definition of H shown in [Figure 2](#).

6 Search of the composition

Let $M \doteq \langle S, I(S), T(S, S'), F(S) \rangle$ be a fair transition system and \mathcal{H} be a set of \mathcal{AG} -skeletons. We want to find a subset $\{H^0, \dots, H^n\} \subseteq \mathcal{H}$, with a composition $C \doteq H^0 \otimes \dots \otimes H^n$ such that: (i) the symbols associated to the \mathcal{AG} -skeletons in the subset are pairwise disjoint and define a covering of S ; (ii) C is an underapproximation of M ; (iii) C admits a loop over the regions such that there exists a reachable region in the loop and one of the regions underapproximates the fair states $F(S)$ of M .

We propose an incomplete procedure to find such C , that relies on a reduction to a sequence of reachability problems and SMT queries. [Algorithm 1](#) shows the main steps required by our procedure. The function `FILTER-INCORRECT-HINTS` ([line 1](#)) filters the list of hints by keeping only those that satisfy [condition 1](#): a satisfiability query checks whether two regions admit some transition between them and if this is the case the unsatisfiability of the $\exists \forall \exists$ formula is decided by employing a variant of the approach presented in [\[16\]](#). Once the correctness of the \mathcal{AG} -skeletons has been established the problem of identifying an \mathcal{R} -abstraction is encoded as a reachability problem by calling the function `GET-REACHABILITY-PROBLEM` ([line 6](#)). Then, `CHECK-REACHABILITY` ([line 7](#)) relies on a model checker to identify a witness for the reachability problem. From the witness `COMPOSITION-FROM-TRACE` ([line 11](#)) constructs a candidate composition. At this point `CHECK-ASSUMPTIONS` ([line 12](#)) checks whether the candidate composition satisfies also the compatibility requirement of the composition operator,

via a sequence of SMT validity checks. If all those checks succeed then we found a composition that meets all the requirements and the procedure stops; otherwise, at least one validity check failed, and the SMT solver provides an assignment that describes a transition for each \mathcal{AG} -skeleton such that those transitions are not *compatible*. In the pseudocode, we refer to this assignment as *bad*. We can refine our reachability encoding by forbidding such composition, by adding $\neg bad$ as an additional invariant constraint to the reachability problem. In this way, we keep refining the encoding and asking the model checker for a candidate composition, until either a valid composition is found or the target state becomes unreachable. In this second case the procedure must stop without providing a definite answer (line 9).

Algorithm 1 FIND-COMPOSITION(M, \mathcal{H})

```

1:  $\mathcal{H} \leftarrow \text{FILTER-INCORRECT-HINTS}(\mathcal{H})$ 
2:  $constr \leftarrow \top$ 
3:  $bad \leftarrow \perp$ 
4: while true do
5:    $constr \leftarrow constr \wedge \neg bad$ 
6:    $prob \leftarrow \text{GET-REACHABILITY-PROBLEM}(\mathcal{H}, M, constr)$ 
7:    $trace \leftarrow \text{CHECK-REACHABILITY}(prob)$ 
8:   if  $trace = \emptyset$  then
9:     return  $\emptyset$ 
10:  end if
11:   $comp \leftarrow \text{COMPOSITION-FROM-TRACE}(trace, \mathcal{H})$ 
12:   $bad \leftarrow \text{CHECK-ASSUMPTIONS}(comp)$ 
13:  if  $bad = \perp$  then
14:    return  $comp$ 
15:  end if
16: end while

```

We now describe how we build the transition system and the reachability problem returned in line 6. We begin by computing, using a sequence of SMT validity checks, underapproximations of T and F that will allow us to construct a composition satisfying conditions H.3 and H.4, while H.5, H.6 and H.7 are implied by I if the composition allows for at least a loop over the regions.

Condition H.3 [resp. H.4] requires us to decide whether the transition relation [resp. some region] of the composed \mathcal{AG} -skeleton implies the transition relation [resp. fairness condition] of M . The transition relation and regions of the composed \mathcal{AG} -skeleton, by definition of the composition operator, are given by the conjunction of the transition relations and restricted regions of the \mathcal{AG} -skeletons involved in the composition. Therefore, we need to decide the validity of a formula of shape $(\bigwedge_{j=0}^k c_j) \rightarrow \phi$, where ϕ is either the transition relation or the fairness condition of M and the c_j are, respectively, the transition relations or the restricted regions of the components. Assume ϕ is in negated normal form.

We apply the following rewriting recursively:

$$((\bigwedge_{j=0}^k c_j) \rightarrow \phi) \mapsto \begin{cases} ((\bigwedge_{j=0}^k c_j) \rightarrow \phi_0) \wedge ((\bigwedge_{j=0}^k c_j) \rightarrow \phi_1) & \text{if } \phi \doteq \phi_0 \wedge \phi_1 \\ ((\bigwedge_{j=0}^k c_j) \rightarrow \phi_0) \vee ((\bigwedge_{j=0}^k c_j) \rightarrow \phi_1) & \text{if } \phi \doteq \phi_0 \vee \phi_1 \end{cases}$$

Notice that in the second case, if the formulae contain some non-convex theory it might be the case that the original formula holds while our rewritten formula does not. Therefore, we are guaranteed that if the rewritten formula holds, so does the original implication, but the vice-versa might not hold. We apply this rewriting until we obtain a formula that is the conjunction and disjunction of implications with a single positive or negated literal on the right hand side. Finally, we again underapproximate the truth assignment of each implication $(\bigwedge_{j=0}^k c_j) \rightarrow l$, where l is either a positive or negative literal by checking whether for some c_j the following is valid: $c_j \rightarrow l$. We rely on the SMT-solver to decide the validity of such implications, and include such results in our encoding of the problem such that any composition will satisfy conditions H.3 and H.4. In the following we detail how we include these observations in the encoding of our problem. We remark that we need to handle the case in which the SMT-solver is unable to provide a definite answer (e.g. because it runs out of resources and/or the support for the underlying theory is incomplete). Let \mathcal{P}^T and \mathcal{P}^F be the set of atomic formulas occurring in $\text{NNF}(T)$ and $\text{NNF}(F)$ respectively. We introduce, for each \mathcal{AG} -skeleton H^j in \mathcal{H} , for each predicate $f_k \in \mathcal{P}^F$, a boolean variable $\text{IST}(f_k^H, i)$, and for each predicate $t_k \in \mathcal{P}^T$, a boolean variable $\text{IST}(t_k^H, i, i')$. We define, for each regions $R_i^j, R_{i'}^j \in \mathcal{R}^j$,

$$\begin{aligned} \text{eval}(\text{IST}(f_k^H, i)) &:= \begin{cases} \top & \text{if } R_i^j \wedge A_i^j \models p_k^F \\ \perp & \text{if } R_i^j \wedge A_i^j \models \neg p_k^F \\ ? & \text{otherwise} \end{cases} \\ \text{eval}(\text{IST}(t_k^H, i, i')) &:= \begin{cases} \top & \text{if } R_i^j \wedge A_i^j \wedge T^j \wedge \\ & R_{i'}^j \wedge A_{i'}^j \models p_k^F \\ \perp & \text{if } R_i^j \wedge A_i^j \wedge T^j \wedge \\ & R_{i'}^j \wedge A_{i'}^j \models \neg p_k^F \\ ? & \text{otherwise} \end{cases} \end{aligned}$$

We then combine the predicates for all \mathcal{AG} -skeletons in \mathcal{H} by defining

$$\text{IST}(f_k, i) \doteq (\bigvee_{H^j \in \mathcal{H}} \text{IST}(f_k^{H^j}, i) = \top) \wedge \bigwedge_{H^j \in \mathcal{H}} \text{IST}(f_k^{H^j}, i) \neq \perp$$

and its negated counterpart as

$$\text{ISF}(f_k, i) \doteq (\bigvee_{H^j \in \mathcal{H}} \text{IST}(f_k^{H^j}, i) = \perp) \wedge \bigwedge_{H^j \in \mathcal{H}} \text{IST}(f_k^{H^j}, i) \neq \top$$

Similarly we define $\text{IST}(t_k, i, i')$ and $\text{ISF}(t_k, i, i')$ over the $\text{IST}(t_k^{H^j}, i, i')$. The *universal abstraction* of F , denoted as \widehat{F} , at region R_i is obtained by replacing in

$\text{NNF}(F)$ every positive literal f_k with $\text{IST}(f_k, i)$ and every negative literal $\neg f_k$ with $\text{ISF}(f_k, i)$. The universal abstraction of T , denoted as \widehat{T} , between regions R_i and $R_{i'}$ is obtained similarly by replacing in $\text{NNF}(T)$ every positive [negative, resp.] occurrence $t_k \in T$ with $\text{IST}(t_k, i, i')$ [$\text{ISF}(t_k, i, i')$, resp.].

Since $\widehat{T} \models T$ and $\widehat{F} \models F$, in our encoding we need to ensure that \widehat{T} holds in every transition and that there exists a region in the loop that satisfies \widehat{F} .

With the construction above, we can now define the transition system $E \doteq \langle S^E, I^E, T^E, F^E \rangle$ as follows:

- $S^E \doteq S \cup S^H \cup S^{\text{Choice}} \cup \{\text{prefix}\} \cup S^{\text{l2s}} \cup S^P$, where:
 - S are the symbols of the input system M ;
 - $S^H \doteq \{l_{H^j} \mid H^j \in \mathcal{H}\}$ are symbols used to keep track of the index of the current region of each \mathcal{AG} -skeleton H^j ;
 - $S^{\text{Choice}} \doteq \{\text{enable}_{H^j} \mid H^j \in \mathcal{H}\}$ is a set of booleans;
 - prefix is an integer;
 - $S^{\text{l2s}} \doteq \{\text{inLoop}, \text{fairLoop}\} \cup \{\text{lBack}_{H^j} \mid H^j \in \mathcal{H}\}$, where the first two are booleans and the lBack_{H^j} are used to nondeterministically choose the loop-back region for the \mathcal{AG} -skeleton H^j ;
 - $S^P \doteq \{\text{IST}(f_k^{H^j}) \mid p_k^F \in F \text{ and } H^j \in \mathcal{H}\} \cup \{\text{IST}(t_k^{H^j}) \mid p_k^T \in T \text{ and } H^j \in \mathcal{H}\}$ are symbols with domain $\{\top, \perp, ?\}$;
- $I^E \doteq I \wedge \text{prefix} > 0 \wedge \neg \text{inLoop} \wedge \neg \text{fairLoop} \wedge I^{\text{Choice}}$ is the initial condition, where I^{Choice} constrains the assignments over S^{Choice} such that the symbols of the enabled components are pairwise disjoint and a covering of S (where the set of enabled components in a state s is $\{H^j \in \mathcal{H} \mid s \models \text{enable}_{H^j}\}$).
- $T^E \doteq T^{\text{Enable}} \wedge T^{\text{Prefix}} \wedge T^{\text{Loop}}$, where:
 - $T^{\text{Enable}} \doteq \bigwedge_{\text{enable}_H \in S^{\text{Choice}}} \text{enable}'_H = \text{enable}_H$ ensures that the choice of enabled components is fixed for each trace;
 - $T^{\text{Prefix}} \doteq \text{prefix} > 0 \rightarrow T(S, S') \wedge \text{prefix}' = \text{prefix} - 1 \wedge \neg \text{inLoop}' \wedge \neg \text{fairLoop}'$ allows E to perform prefix steps following the transition relation of M ; this prefix ensures the reachability of the resulting composition (hypothesis H.1);
 - $T^{\text{Loop}} \doteq \text{prefix} = 0 \rightarrow T^{\text{Aut}} \wedge T^{\text{l2s}} \wedge \widehat{T} \wedge \text{prefix}' = 0$ ensures that, as soon as the prefix finishes, \widehat{T} , which implies T , holds at every step (hypothesis H.3) and E must follow the transition relation of the enabled \mathcal{AG} -skeletons, where:
 - * $T^{\text{l2s}} \doteq \text{inLoop}' = (\text{inLoop} \vee \text{lBack}) \wedge \text{fairLoop}' = (\text{fairLoop} \vee \widehat{F})$ and $\text{lBack} \doteq \bigwedge_{H^j \in \mathcal{H}} l_{H^j} = \text{lBack}_{H^j}$ holds iff all components are in their loopback location.
 - * $T^{\text{Aut}} \doteq \bigwedge_{H^j \in \mathcal{H}} (\text{enable}_{H^j} \rightarrow T^{H^j \text{ enabled}}) \wedge (\neg \text{enable}_{H^j} \rightarrow l_{H^j} = l'_{H^j} \wedge \bigwedge_{p^H \in S^P} p^H = ?)$ defines how the \mathcal{AG} -skeletons evolve: disabled components never change their location and they cannot contribute in satisfying \widehat{T} and \widehat{F} , whereas enabled ones evolve according to their transition relation: $T^{H^j \text{ enabled}} \doteq T^{\text{PredAbs}}_{H^j} \wedge T^{H^j}(S, S') \bigwedge_{i, i'} (l_{H^j} = i \wedge l'_{H^j} = i') \rightarrow R_i^{H^j}(S) \wedge A_i^{H^j}(S)$, where $T^{\text{PredAbs}}_{H^j}$ encodes the truth assignments to the S^P as follows: $T^{\text{PredAbs}}_{H^j} \doteq (\bigwedge_{R_i \in \mathcal{R}_{H^j}} l_{H^j} = i \rightarrow$

$\bigwedge_{p_k \in F} \text{IST}(f_k^{H^j}, i) = \text{eval}(\text{IST}(f_k^{H^j}, i)) \wedge (\bigwedge_{R_i, R_{i'} \in \mathcal{R}^{H^j}} (l_{H^j} = i \wedge l'_{H^j} = i') \rightarrow \bigwedge_{p_k \in T} \text{IST}(t_k^H, i, i') = \text{eval}(\text{IST}(t_k^H, i, i')))$, where $\text{eval}(\text{IST}(f_k^{H^j}, i))$ and $\text{eval}(\text{IST}(t_k^H, i, i'))$ are the assignments we computed previously for the fairness and transition predicates respectively. This transition relation requires to find an assignment for S and S' such that the conjunction of the enabled transitions is satisfied, ensuring that hypothesis **I** is not trivially satisfied because of the lack of any transition between the regions.

Using this encoding, the reachability problem asks whether there exists a path in E that reaches a state in which $\text{fairLoop} \wedge \text{lBack}$ holds. T^{l2s} ensures that the path found by the model checker (line 7) will be a lasso-shape over the regions³, and there will be at least one region in the loop that satisfies \hat{F} , which implies F .

If the model checker finds a path (line 7), the assignments to enable_{H^j} and l_{H^j} for each $H^j \in \mathcal{H}$ describe the subset of \mathcal{AG} -skeletons, the locations and transitions to be considered at every state and transition to obtain the composed \mathcal{R} -abstraction. In this way we can construct the candidate composition from the obtained trace (line 11).

In the following we show that the \mathcal{AG} -skeleton found by Algorithm 1 meets all the hypothesis required for an \mathcal{R} -abstraction.

- Hypothesis **H.0** holds since in the initial condition I^E we ensure that the local symbols of the \mathcal{AG} -skeletons are pairwise disjoint and cover S^M .
- Hypothesis **H.1** holds since in the encoding we allow for *prefix* steps starting from I^M before reaching some conjunction of the regions of the enabled \mathcal{AG} -skeletons.
- Hypothesis **H.2** holds since the initial condition of the \mathcal{R} -abstraction is exactly the state reached after prefix steps, which by construction is in one of the regions.
- Hypothesis **H.3** holds since T holds at every step in which *prefix* > 0 , and for *prefix* $= 0$ \hat{T} must hold, which implies T .
- The liveness-to-safety construction ensures that there exist a region in the composed \mathcal{AG} -skeleton that satisfies \hat{F} , and hence implies F . We call such region R_0 in the \mathcal{R} -abstraction, hence **H.4** holds.
- Hypotheses **H.5**, **H.6** and **H.7** are implied by **I**. The liveness-to-safety construction allows the procedure to find a sequence of regions R_0, \dots, R_k , such that R_0 is fair and $R_k = R_0$, then the encoding E ensures that for all $0 \leq i < k$, $\exists S, S' : R_i(S) \wedge T(S, S') \wedge R_{i+1}(S')$, hence **I** is not trivially satisfied due to the lack of transitions.

7 Ensuring divergence of a given symbol

In timed and hybrid systems there is an additional requirement for an infinite counterexample to be valid: there is an explicit notion of “time” whose assign-

³ Note that this is the liveness-to-safety construction of [7].

ments must diverge to infinity. When encoding a hybrid system as a transition system, time is typically modeled with an additional variable δ representing the duration of each transition (where $\delta = 0$ for discrete transitions and $\delta \geq 0$ for transitions corresponding to time elapses). In order for a transition system trace (of infinite length) to be valid for the original hybrid system, it must not impose any upper bound on the total time elapsed; in other words, the assignments of δ along the trace must describe a series that diverges to infinity. We call such traces *non-zeno*.

This section identifies an approach to restrict the language of an \mathcal{AG} -skeleton or a \mathcal{R} -abstraction to a non-empty set such that “time” is guaranteed to diverge to infinity in all infinite executions in the language. Theorem 4 shows that the composition operator preserves this property.

Theorem 4. *If all infinite executions of the \mathcal{AG} -skeleton A responsible for δ are non-zeno, then also every infinite path of every composition, involving A , is non-zeno.*

Therefore, if it is possible to prove this property locally for the \mathcal{AG} -skeleton we are guaranteed that the composition will preserve it. However, if the local information is insufficient to determine whether all its traces are non-zeno, a global analysis of the final composition is required. For this reason, we show how to shrink the language of an \mathcal{AG} -skeleton or a \mathcal{R} -abstraction so that all its paths are non-zeno, while preserving hypothesis I in the first case, and the hypotheses required by Definition 1 of \mathcal{R} -abstraction in the second one.

In the following we will refer generically to regions and transitions meaning the restricted region $R_i \wedge A_i$ in the case of an \mathcal{AG} -skeleton and the region R_i in the case of an \mathcal{R} -abstraction. We write $a +_n b$, with $n \in \mathbb{N}$ to represent the sum of a and b modulo n . We assume that the domain of δ are the positive reals and that the predicates involving δ in every region i and transition from region i to $i +_n 1$ can be written respectively as $\delta \bowtie f(S \setminus \{\delta\})$ and $\delta' \bowtie g(S \setminus \{\delta\}, S' \setminus \{\delta'\})$, where $\bowtie \in \{<, \leq, =, \geq, >\}$.

Consider one loop over the regions at a time. Let $n \in \mathbb{N}$ be the length of such loop and $R_i(S_i)$ be the i^{th} region in the loop. For each transition $T_{i,i+n1}(S, S') \doteq R_i(S) \wedge T(S, S') \wedge R_{i+n1}(S')$ from R_i to R_{i+n1} in the loop assume we are given a function $low_{i,i+n1} : S \rightarrow \mathbb{R}$ that maps every assignment in $R_i(S)$ to a real value such that:

$$\sum_{it=0}^{\infty} \sum_{i=0}^{n-1} low_{i,i+n1}(S_i^{it}) = +\infty$$

where S_i^{it} is the assignment prescribed by the infinite unrolling of the loop at location i during the it^{th} iteration. We want to restrict the paths corresponding to our loop over the regions to only the paths such that:

$$\bigwedge_{i=0}^{n-1} R_i(S_i^{it}) \wedge T_{i,i+n1}(S_i^{it}, S_{i+n1}^{it}) \wedge \delta_{i+n1}^{it} \geq low_{i,i+n1}(S_i^{it})$$

for all iterations it and where δ_{i+n1} is the evaluation of δ at location $i +_n 1$. Since the sum of the $low_{i,i+n1}$ diverges to infinity and it is a lower bound for the assignments to δ , every path satisfying the condition above is non-zeno.

We now identify some sufficient conditions for this additional constraint to preserve the required hypotheses.

The sufficient condition requires $low_{i,i+n1}$ to be a lower bound for the smallest upper bound for $\delta_{i,i+n1}$ at every transition $T_{i,i+n1}$ for all paths starting from some S_0 . We define such bound for transition $T_{i,i+n1}$ as $min_\delta(S_i, S_{i+n1})$. $min_\delta(S_i, S_{i+n1})$ is the minimum of all $g(S_i, S_{i+n1})$ and $f(S_{i+n1})$ such that $\delta' \lesssim g(S_i, S_{i+n1}) \in T_{i,i+n1}(S_i, S_{i+n1})$ and $\delta' \lesssim f(S_{i+n1}) \in R_{i+n1}(S_{i+n1})$, for some $\lesssim \in \{<, \leq\}$ and functions f, g that do not contain any of δ and δ' . We define the following condition for $low_{i,i+n1}$:

$$\models \forall S_0, \dots, S_{n-1} : \left(\bigwedge_{i=0}^{n-1} R_i(S_i) \wedge T_{i,i+n1}(S_i, S_{i+n1}) \wedge R_{i+n1}(S_{i+n1}) \right) \rightarrow \sum_{i=0}^{n-1} min_\delta(S_i, S_{i+n1}) > low_{i,i+n1}(S_i)$$

Theorem 5. *Given a loop over $n \in \mathbb{N}$ regions $R_0(S), \dots, R_{n-1}(S)$ and n functions $low_{i,i+n1} : S \rightarrow \mathbb{R}$ that map every state in $R_i(S)$ to a real value, such that the following holds:*

$$\models \forall S_0, \dots, S_{n-1} : \left(\bigwedge_{i=0}^{n-1} R_i(S_i) \wedge T_{i,i+n1}(S_i, S_{i+n1}) \wedge R_{i+n1}(S_{i+n1}) \right) \rightarrow \sum_{i=0}^{n-1} min_\delta(S_i, S_{i+n1}) > low_{i,i+n1}(S_i)$$

where $min_\delta(S_i, S_{i+n1})$ is defined as above. Then replacing every transition $T_{i,i+n1}(S_i, S_{i+n1})$ with $T_{i,i+n1}(S_i, S_{i+n1}) \wedge \delta_{i+n1} \geq low_{i,i+n1}(S_i)$ preserves hypothesis **I** in the case of an \mathcal{AG} -skeleton and all the hypotheses of Definition 1 in the case of a \mathcal{R} -abstraction.

7.1 Example: diverging “time”

Consider the \mathcal{AG} -skeleton DV defined in subsection 5.1 for the bouncing ball example. We know that in every loop of DV δ is equal to zero in region R_0^{DV} and to $\frac{1}{c}$ in R_1^{DV} . However, we do not have any information about c and we are unable to conclude anything about the summation of the assignments to the symbol δ in its executions. Then, we need to consider the \mathcal{R} -abstraction represented in Figure 2. In this case we also know that $c \geq 1$ and its value increases by 1 in every iteration. We can define $low_{1,0}(c) \doteq \frac{1}{c+1}$ and $low_{0,1}(c) \doteq 0$. Their summation can be written as:

$$\sum_{it=0}^{+\infty} low_{0,1}(c^{it}) + low_{1,0}(c^{it}) = \sum_{it=0}^{+\infty} \frac{1}{c^{it}+1}$$

This corresponds to the well-known diverging harmonic series. Therefore, we can use $low_{0,1}$ and $low_{1,0}$ as lower bounds for δ . In this case this has no effect on the language of \mathcal{R} -abstraction, hence all its executions were already non-zero paths.

8 Related work

Most of the literature in verification of temporal properties of infinite-state transition systems, hybrid automata and termination analysis focuses on the universal case, while the existential one has received relatively little attention.

The most closely related works to ours are proving *program non-termination*. [25] and [14] are based on the notion of closed recurrence set, that corresponds to proving the non-termination of a relation. We compare our approach with such techniques in subsection 4.1. [9] and [32] search for non-terminating executions via a sequence of safety queries. Other approaches look for specific classes of programs ([21] and [26] prove the decidability of termination for linear loops over the integers), or specific non-termination arguments (in [33] non-termination is seen as the sum of geometric series).

A first obvious difference is that these approaches rely on the existence of a control flow graph, whereas we work at the level of transition system. Moreover, none of these works deals with fairness and our approach can be seen as building a generalization of a closed recurrence set to the fair case. Another key difference with all the above approaches is that they synthesize a monolithic non-termination argument. We propose the composition of a finite number of partial non-termination arguments to prove the non-termination of the whole system. Assume-guarantee style compositional reasoning [23] is a broad topic concerned with the verification of properties. Instead, we employ such kind of reasoning for the falsification of temporal properties.

The only work that explicitly deals with fairness for infinite-state programs is [15], that supports full CTL* and is able to deal with existential properties and to provide fair paths as witnesses. The approach is fully automatic, but it focuses on programs manipulating integer variables, with an explicit control-flow graph, rather than more general symbolic transition systems expressed over different theories (including non-linear real arithmetic). Another approach supporting full CTL* is proposed in [28]. The work presents a model checking algorithm for the verification of CTL* on finite-state systems and a deductive proof system for CTL* on infinite-state systems. In the first case they reduce the verification of CTL* properties to the verification of properties without temporal operators and a single fair path quantifier in front of the formula. To the best of our knowledge there is no generalisation of this algorithm, first reported in [29] and then also in [30], to the infinite-state setting. The rules presented in the second case have been exploited in [6] to implement a procedure for the verification of CTL properties, while our objective is the falsification of LTL properties.

Moreover, in these settings ([15], [28]) there is no notion of non-zenoness.

The analysis of *hybrid systems* deals with more general dynamics than our setting. Most of the works focus on the computation of the set of reachable states, with tools such as FLOW* [10], SpaceEx [20], CORA [1], PHAVer [18] and PHAVerLite [3], that compute an overapproximation of the reachable states using different structures, for example Taylor models, polytopes, polyhedra, support functions. Interestingly, Ariadne [5] computes both an over and under approximation of the reachable set, and can prove and disprove a property, but

limited to the case of reachability properties. The few works on *falsification of temporal properties* [35, 37, 38] have the common trait of being the restriction to logic fragments (*bounded-time* MTL, LTL safety properties) for which finite witnesses are sufficient. Tools in this context, such as TALiRO [2], rely on simulations to find such finite witnesses. Instead, we are interested in identifying infinite witnesses for more general temporal properties. Finally, the HyCOMP model checker [13] supports hybrid systems verification of LTL via a reduction to infinite-state model checking. Its verification procedure k-zeno [12] can only disprove the property when lasso-shaped counterexamples exist.

The works on *timed automata* are less relevant: although the concrete system may exhibit no lasso-shape witnesses, due to the divergence of clocks, the problem is decidable, and lasso-shaped counterexamples exist in finite bi-simulating abstractions. This view is adopted in Uppaal [4], CTAV [34] and LTSmin [27]. Other tools directly search for non lasso-shaped counterexamples, but the proposed techniques are specific for the setting of timed automata [11, 31] and lack the generality of the method proposed in this paper.

9 Experimental evaluation

In order to evaluate the practical feasibility of our approach, we have implemented the procedure described in Sec. 6 by relying on the PYSMT library [22] to interact with SMT solvers, and the NUXMV model checker [8] to perform the reachability checks. Our prototype tool FAIRFIND takes as input a symbolic transition system, a fairness condition and a set of \mathcal{AG} -skeletons used as building blocks (or *hints*) for constructing the \mathcal{R} -abstraction and implements Algorithm 1. When successful, FAIRFIND returns a suitable set of regions \mathcal{R} and a \mathcal{R} -abstraction A of M satisfying all the conditions H.0–H.7 presented in Sec. 4. A is the result of a suitable composition of a subset of the input hints. When successful, FAIRFIND is able to produce a *proof* of the validity of the produced \mathcal{R} -abstraction as a sequence of SMT queries, which can be independently checked. This additional check increases the confidence on the correctness of the obtained results. In our evaluation, we have successfully verified the correctness

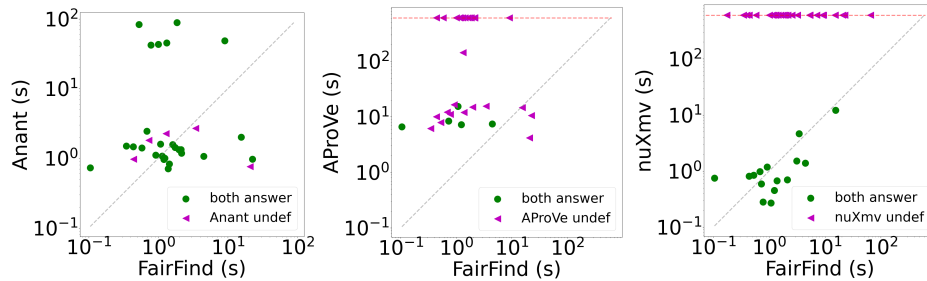


Fig. 3. Execution time of FAIRFIND compared to ANANT, APROVE and NUXMV.

of our results in all cases except 3, for which this additional correctness check fails to provide a definite answer. This fact supports the significance of the approach in the sense that it was able to identify a \mathcal{R} -abstraction for which we are unable to directly prove the validity of the required hypotheses.

We have tested FAIRFIND on 43 benchmark instances: 31 are non-linear software non-termination problems, and 12 are LTL verification problems, 9 on hybrid systems and 3 on infinite state transition systems. 29 of the software benchmarks have been taken from [14], while the remaining 2 are new benchmarks we defined. Among the hybrid systems benchmarks, 4 are variations of our running example, whereas the remaining 5 have been taken from the ARCH competition on hybrid systems verification [19]. In our experiment, we have defined the hints manually. In most cases, the \mathcal{AG} -skeletons are responsible for the evolution of a single variable of the input system. We defined an average of 5 hints per benchmark (with a minimum of 2 and a maximum of 17). We ran FAIRFIND with a total timeout of 600 seconds per benchmark, and a timeout of 5 seconds for each SMT query⁴. FAIRFIND was able to produce a witness \mathcal{R} -abstraction for all the benchmarks, suggesting the practical viability of the approach.

We also compared FAIRFIND with two fully automatic procedures for program (non-)termination, ANANT [14] and APROVE [24] (limited to the software non-termination benchmarks), and with the LTL model checker NUXMV [8] (on all the benchmarks). The objective is not to directly contrast the performance of the various tools, as they operate under very different assumptions: FAIRFIND is more general, but it requires human assistance, whereas ANANT and APROVE are specialised tools for software (non-)termination, and NUXMV has very limited support for LTL counterexamples on infinite-state systems [11]. Rather, the goal here is to assess the significance of the benchmarks w.r.t. the state of the art. The results of this experiments are presented in the scatter plots of Fig. 3. From the plots, we can see that none of the other tools is able to solve all the benchmarks solved by FAIRFIND, and in fact there are 13 instances that are uniquely solved by FAIRFIND (3 of the software benchmarks, 1 of the transition systems and all the hybrid benchmarks).

Fig. 4 shows the increase in execution time of FAIRFIND as we increase the number of \mathcal{AG} -skeletons provided. The objective of this evaluation is to test the robustness of the approach with respect to an increasing number of unnecessary and/or redundant hints. For this reason we increase the number of \mathcal{AG} -skeletons such that in all cases the procedure selects the same

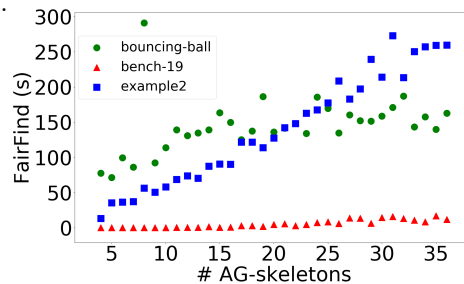


Fig. 4. Execution time of FAIRFIND with increasing number of \mathcal{AG} -skeletons.

⁴ This allows the procedure to make progress even if the solver is unable to provide a definite answer for some query. Many of the benchmarks require reasoning in mixed integer/real non-linear arithmetic (in general undecidable).

set of hints. We considered 3 benchmarks: our non-linear, hybrid, running example, one of the software benchmarks taken from [14] (*bench-19*) and one of the non-linear software benchmarks we defined (*example-2*). We let FAIRFIND run with a number of \mathcal{AG} -skeletons ranging from 4 to 36. We observe a worst case linear increase in execution time in these experiments. In addition, these benchmarks show three different behaviours. In the bouncing ball case, most of the execution time is spent by the model checker in identifying candidate compositions. In the example-2 case the execution time is dominated by the time required to compute the validity of the implications required for the approximations \hat{T} and \hat{F} . Finally, in the bench-19 case, the execution time is much lower than in the other two cases, but FAIRFIND performs a higher number of refinements of candidate compositions. In all these cases the procedure has to deal with many non-linear expressions, and this could cause high execution times and instabilities; in fact, sometimes, by increasing the number of \mathcal{AG} -skeletons the required time decreases. However, the results we obtained seem promising and we did not observe a blow-up in the time required to identify the \mathcal{R} -abstraction.

10 Conclusions

We tackled the problem of proving the existence of fair paths in infinite-state fair transition systems, proposing a deductive framework based on a combination of under-approximations. The framework also encompasses diverging fair paths, required to deal with zenoness. Then, we defined and implemented a procedure to search for a proof based on a suitable composition of \mathcal{AG} -skeletons. The experimental evaluation shows that the framework is highly expressive, and the procedure effectively finds fair paths on benchmarks from software non-termination and hybrid systems falsification.

In the future, we will extend the automation of the search procedure and integrate it with a complementary procedure to demonstrate the dual universal property. In order to increase the automation we plan to exploit current techniques in the context of software non-termination and syntax-guided approaches as procedures to synthesise \mathcal{AG} -skeletons. Many of the \mathcal{AG} -skeletons that have been used in our benchmarks could be synthesised by such techniques. However, some of them, such as the ones in our running example, require the ability to heavily reason about non-linear systems and might be harder to synthesise automatically. For this reason the possibility of taking and verifying hints from the user might be relevant to successfully identify an \mathcal{R} -abstraction for complex systems.

We will also experiment the applicability in the finite state case, and integrate the method into satisfiability procedures for temporal logics over hybrid traces.

References

1. Althoff, M.: An introduction to CORA 2015. In: Frehse, G., Althoff, M. (eds.) 1st and 2nd International Workshop on Applied verification for Continuous

- and Hybrid Systems, ARCH@CPSWeek 2014, Berlin, Germany, April 14, 2014 / ARCH@CPSWeek 2015, Seattle, WA, USA, April 13, 2015. EPiC Series in Computing, vol. 34, pp. 120–151. EasyChair (2015), <http://www.easychair.org/publications/paper/248657>
2. Annpureddy, Y., Liu, C., Fainekos, G.E., Sankaranarayanan, S.: S-taliro: A tool for temporal logic falsification for hybrid systems. In: Abdulla, P.A., Leino, K.R.M. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 17th International Conference, TACAS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26–April 3, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6605, pp. 254–257. Springer (2011). https://doi.org/10.1007/978-3-642-19835-9_21, https://doi.org/10.1007/978-3-642-19835-9_21
 3. Becchi, A., Zaffanella, E.: Revisiting polyhedral analysis for hybrid systems. In: Chang, B.E. (ed.) Static Analysis - 26th International Symposium, SAS 2019, Porto, Portugal, October 8–11, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11822, pp. 183–202. Springer (2019). https://doi.org/10.1007/978-3-030-32304-2_10, https://doi.org/10.1007/978-3-030-32304-2_10
 4. Behrmann, G., David, A., Larsen, K.G.: A tutorial on UPPAAL. In: Bernardo, M., Corradini, F. (eds.) Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004. pp. 200–236. No. 3185 in LNCS, Springer-Verlag (September 2004)
 5. Benvenuti, L., Bresolin, D., Collins, P., Ferrari, A., Geretti, L., Villa, T.: Assume-guarantee verification of nonlinear hybrid systems with ariadne. *International Journal of Robust and Nonlinear Control* **24**(4), 699–724 (2014)
 6. Beyene, T.A., Popeea, C., Rybalchenko, A.: Solving existentially quantified horn clauses. In: Sharygina, N., Veith, H. (eds.) Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13–19, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8044, pp. 869–882. Springer (2013). https://doi.org/10.1007/978-3-642-39799-8_61, https://doi.org/10.1007/978-3-642-39799-8_61
 7. Biere, A., Artho, C., Schuppan, V.: Liveness checking as safety checking. *Electron. Notes Theor. Comput. Sci.* **66**(2), 160–177 (2002). [https://doi.org/10.1016/S1571-0661\(04\)80410-9](https://doi.org/10.1016/S1571-0661(04)80410-9), [https://doi.org/10.1016/S1571-0661\(04\)80410-9](https://doi.org/10.1016/S1571-0661(04)80410-9)
 8. Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., Tonetta, S.: The nuXmv Symbolic Model Checker. In: CAV. Lecture Notes in Computer Science, vol. 8559, pp. 334–342. Springer (2014)
 9. Chen, H.Y., Cook, B., Fuhs, C., Nimkar, K., O’Hearn, P.W.: Proving non-termination via safety. In: Ábrahám, E., Havelund, K. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5–13, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8413, pp. 156–171. Springer (2014). https://doi.org/10.1007/978-3-642-54862-8_11, https://doi.org/10.1007/978-3-642-54862-8_11
 10. Chen, X., Sankaranarayanan, S., Ábrahám, E.: Flow* 1.2: More effective to play with hybrid systems. In: Frehse, G., Althoff, M. (eds.) 1st and 2nd International Workshop on Applied verification for Continuous and Hybrid Systems, ARCH@CPSWeek 2014, Berlin, Germany, April 14, 2014 / ARCH@CPSWeek 2015, Seattle, WA, USA, April 13, 2015. EPiC Series in Computing, vol. 34, pp.

- 152–159. EasyChair (2015), <http://www.easychair.org/publications/paper/248659>
11. Cimatti, A., Griggio, A., Magnago, E., Roveri, M., Tonetta, S.: Extending nuxmv with timed transition systems and timed temporal properties. In: Dillig, I., Tasiran, S. (eds.) *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15–18, 2019, Proceedings, Part I. Lecture Notes in Computer Science*, vol. 11561, pp. 376–386. Springer (2019). https://doi.org/10.1007/978-3-030-25540-4_21, https://doi.org/10.1007/978-3-030-25540-4_21
 12. Cimatti, A., Griggio, A., Mover, S., Tonetta, S.: Verifying LTL properties of hybrid systems with k-liveness. In: Biere, A., Bloem, R. (eds.) *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18–22, 2014. Proceedings. Lecture Notes in Computer Science*, vol. 8559, pp. 424–440. Springer (2014). https://doi.org/10.1007/978-3-319-08867-9_28, https://doi.org/10.1007/978-3-319-08867-9_28
 13. Cimatti, A., Griggio, A., Mover, S., Tonetta, S.: Hycomp: An smt-based model checker for hybrid systems. In: Baier, C., Tinelli, C. (eds.) *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11–18, 2015. Proceedings. Lecture Notes in Computer Science*, vol. 9035, pp. 52–67. Springer (2015). https://doi.org/10.1007/978-3-662-46681-0_4, https://doi.org/10.1007/978-3-662-46681-0_4
 14. Cook, B., Fuhs, C., Nimkar, K., O’Hearn, P.W.: Disproving termination with overapproximation. In: *Formal Methods in Computer-Aided Design, FMCAD 2014, Lausanne, Switzerland, October 21–24, 2014*. pp. 67–74. IEEE (2014). <https://doi.org/10.1109/FMCAD.2014.6987597>, <https://doi.org/10.1109/FMCAD.2014.6987597>
 15. Cook, B., Khlaaf, H., Piterman, N.: Verifying increasingly expressive temporal logics for infinite-state systems. *J. ACM* **64**(2), 15:1–15:39 (2017). <https://doi.org/10.1145/3060257>, <https://doi.org/10.1145/3060257>
 16. Dutertre, B.: Solving exists/forall problems with yices. In: *Workshop on satisfiability modulo theories* (2015)
 17. Emerson, E.A., Halpern, J.Y.: ”sometimes” and ”not never” revisited: on branching versus linear time temporal logic. *J. ACM* **33**(1), 151–178 (1986). <https://doi.org/10.1145/4904.4999>, <https://doi.org/10.1145/4904.4999>
 18. Frehse, G.: Phaver: Algorithmic verification of hybrid systems past hytech. In: Morari, M., Thiele, L. (eds.) *Hybrid Systems: Computation and Control, 8th International Workshop, HSCC 2005, Zurich, Switzerland, March 9–11, 2005, Proceedings. Lecture Notes in Computer Science*, vol. 3414, pp. 258–273. Springer (2005). https://doi.org/10.1007/978-3-540-31954-2_17, https://doi.org/10.1007/978-3-540-31954-2_17
 19. Frehse, G., Althoff, M. (eds.): ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems, part of CPS-IoT Week 2019, Montreal, QC, Canada, April 15, 2019, EPiC Series in Computing, vol. 61. EasyChair (2019)
 20. Frehse, G., Guernic, C.L., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: Spaceex: Scalable verification of hybrid systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) *Computer Aided Veri-*

- fication - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6806, pp. 379–395. Springer (2011). https://doi.org/10.1007/978-3-642-22110-1_30, https://doi.org/10.1007/978-3-642-22110-1_30
21. Frohn, F., Giesl, J.: Termination of triangular integer loops is decidable. In: Dillig, I., Tasiran, S. (eds.) Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11562, pp. 426–444. Springer (2019). https://doi.org/10.1007/978-3-030-25543-5_24, https://doi.org/10.1007/978-3-030-25543-5_24
 22. Gario, M., Micheli, A.: Pysmt: a solver-agnostic library for fast prototyping of smt-based algorithms. In: SMT Workshop 2015 (2015)
 23. Giannakopoulou, D., Namjoshi, K.S., Pasareanu, C.S.: Compositional reasoning. In: Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R. (eds.) Handbook of Model Checking, pp. 345–383. Springer (2018). https://doi.org/10.1007/978-3-319-10575-8_12, https://doi.org/10.1007/978-3-319-10575-8_12
 24. Giesl, J., Brockschmidt, M., Emmes, F., Frohn, F., Fuhs, C., Otto, C., Plücker, M., Schneider-Kamp, P., Ströder, T., Swiderski, S., Thiemann, R.: Proving termination of programs automatically with aprove. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8562, pp. 184–191. Springer (2014). https://doi.org/10.1007/978-3-319-08587-6_13, https://doi.org/10.1007/978-3-319-08587-6_13
 25. Gupta, A., Henzinger, T.A., Majumdar, R., Rybalchenko, A., Xu, R.: Proving non-termination. In: Necula, G.C., Wadler, P. (eds.) Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008. pp. 147–158. ACM (2008). <https://doi.org/10.1145/1328438.1328459>, <https://doi.org/10.1145/1328438.1328459>
 26. Hosseini, M., Ouaknine, J., Worrell, J.: Termination of linear loops over the integers. In: Baier, C., Chatzigiannakis, I., Flocchini, P., Leonardi, S. (eds.) 46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece. LIPIcs, vol. 132, pp. 118:1–118:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019). <https://doi.org/10.4230/LIPIcs.ICALP.2019.118>, <https://doi.org/10.4230/LIPIcs.ICALP.2019.118>
 27. Kant, G., Laarman, A., Meijer, J., van de Pol, J., Blom, S., van Dijk, T.: LTSmin: High-Performance Language-Independent Model Checking. In: Baier, C., Tinelli, C. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9035, pp. 692–707. Springer (2015). https://doi.org/10.1007/978-3-662-46681-0_61, https://doi.org/10.1007/978-3-662-46681-0_61
 28. Kesten, Y., Pnueli, A.: A compositional approach to ctl* verification. Theor. Comput. Sci. **331**(2-3), 397–428 (2005). <https://doi.org/10.1016/j.tcs.2004.09.023>, <https://doi.org/10.1016/j.tcs.2004.09.023>
 29. Kesten, Y., Pnueli, A., Raviv, L.: Algorithmic verification of linear temporal logic specifications. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) Automata,

- Languages and Programming, 25th International Colloquium, ICALP'98, Aalborg, Denmark, July 13-17, 1998, Proceedings. Lecture Notes in Computer Science, vol. 1443, pp. 1–16. Springer (1998). <https://doi.org/10.1007/BFb0055036>, <https://doi.org/10.1007/BFb0055036>
30. Kesten, Y., Pnueli, A., Raviv, L., Shahar, E.: Model checking with strong fairness. *Formal Methods Syst. Des.* **28**(1), 57–84 (2006). <https://doi.org/10.1007/s10703-006-4342-y>, <https://doi.org/10.1007/s10703-006-4342-y>
 31. Kindermann, R., Junttila, T.A., Niemelä, I.: Beyond lassos: Complete smt-based bounded model checking for timed automata. In: Giese, H., Rosu, G. (eds.) *Formal Techniques for Distributed Systems - Joint 14th IFIP WG 6.1 International Conference, FMOODS 2012 and 32nd IFIP WG 6.1 International Conference, FORTE 2012, Stockholm, Sweden, June 13-16, 2012. Proceedings. Lecture Notes in Computer Science*, vol. 7273, pp. 84–100. Springer (2012). https://doi.org/10.1007/978-3-642-30793-5_6, https://doi.org/10.1007/978-3-642-30793-5_6
 32. Larraz, D., Nimkar, K., Oliveras, A., Rodríguez-Carbonell, E., Rubio, A.: Proving non-termination using max-smt. In: Biere, A., Bloem, R. (eds.) *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings. Lecture Notes in Computer Science*, vol. 8559, pp. 779–796. Springer (2014). https://doi.org/10.1007/978-3-319-08867-9_52, https://doi.org/10.1007/978-3-319-08867-9_52
 33. Leike, J., Heizmann, M.: Geometric nontermination arguments. In: Beyer, D., Huisman, M. (eds.) *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part II. Lecture Notes in Computer Science*, vol. 10806, pp. 266–283. Springer (2018). https://doi.org/10.1007/978-3-319-89963-3_16, https://doi.org/10.1007/978-3-319-89963-3_16
 34. Li, G.: Checking timed büchi automata emptiness using lu-abstractions. In: Ouaknine, J., Vaandrager, F.W. (eds.) *Formal Modeling and Analysis of Timed Systems, 7th International Conference, FORMATS 2009, Budapest, Hungary, September 14-16, 2009. Proceedings. Lecture Notes in Computer Science*, vol. 5813, pp. 228–242. Springer (2009). https://doi.org/10.1007/978-3-642-04368-0_18, https://doi.org/10.1007/978-3-642-04368-0_18
 35. Nghiem, T., Sankaranarayanan, S., Fainekos, G.E., Ivancic, F., Gupta, A., Pappas, G.J.: Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In: Johansson, K.H., Yi, W. (eds.) *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2010, Stockholm, Sweden, April 12-15, 2010. pp. 211–220. ACM* (2010). <https://doi.org/10.1145/1755952.1755983>, <https://doi.org/10.1145/1755952.1755983>
 36. Pasareanu, C.S., Pelánek, R., Visser, W.: Predicate abstraction with under-approximation refinement. *Log. Methods Comput. Sci.* **3**(1) (2007). [https://doi.org/10.2168/LMCS-3\(1:5\)2007](https://doi.org/10.2168/LMCS-3(1:5)2007), [https://doi.org/10.2168/LMCS-3\(1:5\)2007](https://doi.org/10.2168/LMCS-3(1:5)2007)
 37. Plaku, E., Kavraki, L.E., Vardi, M.Y.: Falsification of LTL safety properties in hybrid systems. *Int. J. Softw. Tools Technol. Transf.* **15**(4), 305–320 (2013). <https://doi.org/10.1007/s10009-012-0233-2>, <https://doi.org/10.1007/s10009-012-0233-2>

38. Sankaranarayanan, S., Fainekos, G.E.: Falsification of temporal properties of hybrid systems using the cross-entropy method. In: Dang, T., Mitchell, I.M. (eds.) Hybrid Systems: Computation and Control (part of CPS Week 2012), HSCC'12, Beijing, China, April 17-19, 2012. pp. 125–134. ACM (2012). <https://doi.org/10.1145/2185632.2185653>, <https://doi.org/10.1145/2185632.2185653>