Reachability Games for Linear Hybrid Systems*

Massimo Benerecetti
bene@na.infn.it

Marco Faella
mfaella@na.infn.it

Stefano Minopoli
minopoli@na.infn.it

Università di Napoli
"Federico II", Italy

We consider the problem of computing the controllable region of a Linear Hybrid Automaton with controllable and uncontrollable transitions, w.r.t. a reachability objective. We provide a semi-algorithm for the problem, by proposing the first algorithm in the literature for computing the set of states that must reach a given polyhedron while avoiding another one, subject to a polyhedral constraint on the slope of the trajectory. Experimental results are presented, based on an implementation of the proposed algorithm on top of the tool PHAVer.

Categories and Subject Descriptors

B.5.2 [**Design Aids**]: Automatic synthesis

General Terms

Theory, Verification

Keywords

Hybrid Automata, Controller Synthesis, Formal Methods

1. INTRODUCTION

Hybrid systems are an established formalism for modeling physical systems which interact with a digital controller. From an abstract point of view, a hybrid system is a dynamic system whose state variables are partitioned into discrete and continuous ones.

Hybrid automata [12] are the most common syntactic variety of hybrid system: a finite set of locations, similar to the states of a finite automaton, represents the value of the discrete variables. The current location, together with the current value of the (continuous) variables, form the instantaneous description of the system. Change of location happens via discrete transitions, and the evolution of the variables is governed by differential equations attached to each location. In a Linear Hybrid Automaton (LHA), the allowed differential equations are in fact differential inclusions of the type $\dot{\mathbf{x}} \in P$, where $\dot{\mathbf{x}}$ is the vector of the first derivatives of all variables and $P \subseteq \mathbb{R}^n$ is a convex polyhedron. Notice that differential inclusions are non-deterministic, allowing for infinitely many solutions.

We study LHAs whose discrete transitions are partitioned into controllable and uncontrollable ones, and we wish to compute the set of states from which the controller can ensure a given goal, regardless of the trajectory followed by the continuous variables and despite the occurrence of uncontrollable discrete transitions. Hence, the problem can be viewed as a *two player game* [19]: on one side the controller, who can only issue controllable transitions, on the other side the environment, who can choose the trajectory of the variables and can take uncontrollable transitions whenever they are enabled.

As control goal, we consider reachability, i.e., the objective of reaching a given set T of target states. The problem is known to be undecidable, being harder than the standard reachability verification (i.e., 1-player reachability) for triangular hybrid automata [14], a special case of LHAs. We present a sound and complete semi-algorithm for the problem[1], based on a novel algorithm for computing, within a given location, the set of states that must reach a given polyhedral region while avoiding another one.

We recently presented a semi-algorithm for the control problem of LHAs with *safety* objectives [6, 5]. Although the control goal we examine, as a language of infinite traces, is the dual of safety, the corresponding synthesis problems are not dual, because our game model is asymmetric (the continuous behavior is always uncontrollable). Hence, it is not possible to solve the control problem with reachability goal T by exchanging the roles of the two players and then solving the safety control problem with goal $\overline{T}$ (i.e., the complement of T). To the best of our knowledge, the reachability goal was never considered for LHAs.

We present an implementation of the proposed semi-algorithm in a tool called PHAVer+, and a set of preliminary experiments.

Related work. The idea of automatically synthesizing controllers for dynamic systems arose in connection with discrete systems [17]. Then, the same idea was applied to real-time systems modeled by timed automata [16], thus coming one step closer to the continuous systems that control theory usually deals with. Finally, it was the turn of hybrid systems [13, 10], and in particular of Linear Hybrid Au-

*This work was supported by a "FARO" grant from the Università di Napoli "Federico II".

HSCC'12, April 17–19, 2012, Beijing, China.

[1] In other words, a procedure that may or may not terminate, and that provides the correct answer whenever it terminates.

tomata [20], the very model that we analyze in this paper. Wong-Toi proposed a symbolic semi-algorithm to compute the controllable region of a LHA w.r.t. a safety goal [20]. Our recent work [6] revisits the solution proposed by Wong-Toi, identifying some inaccuracies which prevent completeness of the procedure, and proposes a sound and complete semi-algorithm for the problem.

Tomlin et al. [19] and Balluchi et al. [4] analyze much more expressive models, with generality in mind rather than automatic synthesis.

Asarin et al. [3] investigate the synthesis problem for hybrid systems where all discrete transitions are controllable and the trajectories satisfy given linear differential equations of the type $\dot{\mathbf{x}} = A\mathbf{x}$. The expressive power of these constraints is incomparable with the one offered by LHAs. In particular, linear differential equations give rise to deterministic trajectories, while differential inclusions are nondeterministic. In control theory terms, differential inclusions can represent the presence of environmental *disturbances*. Bouyer et al. [8] propose a general abstraction technique for hybrid systems, and focus on decidable classes of o-minimal automata.

The rest of the paper is organized as follows. Section 2 introduces and motivates the model. The proposed semi-algorithm is presented as divided in two layers: Section 3 illustrates the outer layer, dealing with multiple locations and discrete transitions, while Section 4 focuses on the geometric problem arising from the analysis of a single location. Finally, Section 5 reports some experiments performed on our implementation of the procedure.

2. LINEAR HYBRID AUTOMATA

A *convex polyhedron* is a subset of $\mathbb{R}^n$ that is the intersection of a finite number of strict and non-strict affine half-spaces. A *polyhedron* is a subset of $\mathbb{R}^n$ that is the union of a finite number of convex polyhedra. For a general (i.e., not necessarily convex) polyhedron $G \subseteq \mathbb{R}^n$, we denote by $cl(G)$ its topological closure, and by $[\![G]\!] \subseteq 2^{\mathbb{R}^n}$ its representation as a finite set of convex polyhedra.

Given an ordered set $X = \{x_1, \ldots, x_n\}$ of variables, a *valuation* is a function $v : X \to \mathbb{R}$. Let $Val(X)$ denote the set of valuations over X. There is an obvious bijection between $Val(X)$ and $\mathbb{R}^n$, allowing us to extend the notion of (convex) polyhedron to sets of valuations. We denote by $CPoly(X)$ (resp., $Poly(X)$) the set of convex polyhedra (resp., polyhedra) on X. Let A be a set of valuations, states or points in $\mathbb{R}^n$, we denote by $\overline{A}$ its complement.

We use $\dot{X}$ to denote the set $\{\dot{x}_1, \ldots, \dot{x}_n\}$ of dotted variables, used to represent the first derivatives, and X' to denote the set $\{x'_1, \ldots, x'_n\}$ of primed variables, used to represent the new values of variables after a transition. Arithmetic operations on valuations are defined in the straightforward way. An *activity* over X is a differentiable function $f : \mathbb{R}^{\geq 0} \to Val(X)$. Let $Acts(X)$ denote the set of activities over X. The *derivative* $\dot{f}$ of an activity f is defined in the standard way and it is an activity over $\dot{X}$. A *Linear Hybrid Automaton* $H = (Loc, X, Edg_{\mathrm{c}}, Edg_{\mathrm{u}}, Flow, Inv, Init)$ consists of the following:

- A finite set Loc of *locations*.
- A finite set $X = \{x_1, \ldots, x_n\}$ of continuous, real-valued *variables*. A *state* is a pair $\langle l, v\rangle$ of a location l and a valuation $v \in Val(X)$.
- Two sets Edg_{c} and Edg_{u} of *controllable* and *uncontrollable transitions*, respectively. They describe instantaneous changes of locations, in the course of which variables may change their value. Each transition $(l, \mu, l') \in Edg_{\mathrm{c}} \cup Edg_{\mathrm{u}}$ consists of a *source location* l, a *target location* l', and a *jump relation* $\mu \in Poly(X \cup X')$, that specifies how the variables may change their value during the transition. The projection of μ on X describes the valuations for which the transition is enabled; this is often referred to as a *guard*.
- A mapping $Flow : Loc \to CPoly(\dot{X})$ attributes to each location a set of valuations over the first derivatives of the variables, which determines how variables can change over time.
- A mapping $Inv : Loc \to Poly(X)$, called the *invariant*.
- A mapping $Init : Loc \to Poly(X)$, contained in the invariant, defining the *initial states* of the automaton.

We use the abbreviations $S = Loc \times Val(X)$ for the set of states and $Edg = Edg_{\mathrm{c}} \cup Edg_{\mathrm{u}}$ for the set of all transitions. Moreover, we let $InvS = \bigcup_{l \in Loc}\{l\} \times Inv(l)$ and $InitS = \bigcup_{l \in Loc}\{l\} \times Init(l)$. Notice that $InvS$ and $InitS$ are sets of states. Given a set of states A and a location l, we denote by $A|_l$ the projection of A on l, i.e. $\{v \in Val(X) \mid \langle l, v\rangle \in A\}$.

2.1 Semantics

The behavior of a LHA is based on two types of transitions: *discrete* transitions correspond to the Edg component, and produce an instantaneous change in both the location and the variable valuation; *timed* transitions describe the change of the variables over time in accordance with the $Flow$ component.

Given a state $s = \langle l, v\rangle$, we set $loc(s) = l$ and $val(s) = v$. An activity $f \in Acts(X)$ is called *admissible from* s if *(i)* $f(0) = v$ and *(ii)* for all $\delta \geq 0$ it holds $\dot{f}(\delta) \in Flow(l)$. We denote by $Adm(s)$ the set of activities that are admissible from s. Additionally, for $f \in Adm(s)$, the *span* of f in l, denoted by $span(f, l)$ is the set of all values $\delta \geq 0$ such that $\langle l, f(\delta')\rangle \in InvS$ for all $0 \leq \delta' \leq \delta$. Intuitively, δ is in the span of f iff f never leaves the invariant in the first δ time units. If all non-negative reals belong to $span(f, l)$, we write $\infty \in span(f, l)$.

Runs. Given two states s, s', and a transition $e \in Edg$, there is a *discrete step* $s \xrightarrow{e} s'$ with *source* s and *target* s' iff: *(i)* $s, s' \in InvS$, *(ii)* $e = (loc(s), \mu, loc(s'))$, and *(iii)* $(val(s), val(s')[X'/X]) \in \mu$, where $val(s')[X/X']$ is the valuation in $Val(X')$ obtained from s' by renaming the variables. Whenever there is a discrete step $s \xrightarrow{e} s'$, we say that e is *enabled in* s.

There is a *timed step* $s \xrightarrow{\delta, f} s'$ with *duration* $\delta \in \mathbb{R}^{\geq 0}$ and activity $f \in Adm(s)$ iff: *(i)* $s \in InvS$, *(ii)* $\delta \in span(f, loc(s))$, and *(iii)* $s' = \langle loc(s), f(\delta)\rangle$. For technical convenience, we admit timed steps of duration zero[2]. A special timed step is denoted $s \xrightarrow{\infty, f}$ and represents the case when the system follows an activity forever. This is only allowed if

[2]Timed steps of duration zero can be disabled by adding a clock variable t to the automaton and requesting that each discrete transition happens when $t > 0$ and resets t to 0 when taken.

$\infty \in span(f, loc(s))$. Finally, a *joint step* $s \xrightarrow{\delta,f,e} s'$ represents the timed step $s \xrightarrow{\delta,f} \langle loc(s), f(\delta)\rangle$ followed by the discrete step $\langle loc(s), f(\delta)\rangle \xrightarrow{e} s'$.

A *run* is a sequence

$$r = s_0 \xrightarrow{\delta_0,f_0} s'_0 \xrightarrow{e_0} s_1 \xrightarrow{\delta_1,f_1} s'_1 \xrightarrow{e_1} s_2 \cdots s_n \cdots \quad (1)$$

of alternating timed and discrete transitions, such that either the sequence is infinite, or it ends with a timed transition of the type $s_n \xrightarrow{\infty,f}$. If the run r is finite, we define $len(r) = n$ to be the length of the run, otherwise we set $len(r) = \infty$. The above run is *non-Zeno* if for all $\delta \geq 0$ there exists $i \geq 0$ such that $\sum_{j=0}^{i} \delta_j > \delta$. We denote by $States(r)$ the set of all states visited by r. Formally, $States(r)$ is the set of all states $\langle loc(s_i), f_i(\delta)\rangle$, for all $0 \leq i \leq len(r)$ and all $0 \leq \delta \leq \delta_i$. Notice that the states from which discrete transitions start (states s'_i in (1)) appear in $States(r)$. Moreover, if r contains a sequence of one or more zero-time timed transitions, all intervening states appear in $States(r)$.

Zenoness and well-formedness. A well-known problem of real-time and hybrid systems is that definitions like the above admit runs that take infinitely many discrete transitions in a finite amount of time (i.e., *Zeno* runs), even if such behaviors are physically meaningless. In this paper, we assume that the hybrid automaton under consideration generates no such runs. This is easily achieved by using an extra variable, representing a clock, to ensure that the delay between any two transitions is bounded from below by a constant. We leave it to future work to combine our results with the more sophisticated approaches to Zenoness known in the literature [4, 9].

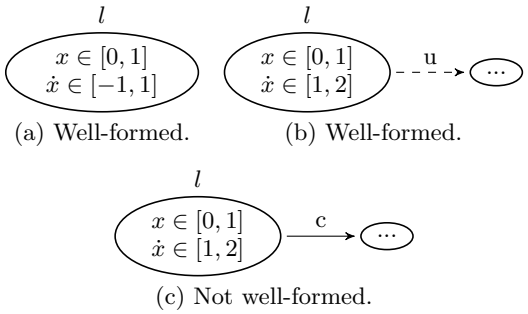
Moreover, we assume that the hybrid automaton under consideration is *non-blocking*, i.e., before all system trajectories leave the invariant, there must be an uncontrollable transition enabled. Formally, for all states s in the invariant, if all activities $f \in Adm(s)$ eventually leave the invariant, there exists one such activity f and a time $\delta \in span(f, loc(s))$ such that $s' = \langle loc(s), f(\delta)\rangle$ is in the invariant and there is an uncontrollable transition $e \in Edg_{\mathrm{u}}$ that is enabled in s'.If a hybrid automaton is non-Zeno and non-blocking, we say that it is *well-formed*. In the following, all hybrid automata are assumed to be well-formed.

Example 1. *Consider the LHAs in Figure 1, in which locations contain the invariant (first line) and the flow constraint (second line). Solid (resp., dashed) edges represent controllable (resp., uncontrollable) transitions, and guards are* `true`*. The fragment in Figure 1(a) is well-formed, because the system may choose derivative $\dot{x} = 0$ and remain indefinitely in location l. The fragment in Figure 1(b) is also well-formed, because the system cannot remain in l forever, but an uncontrollable transition leading outside is always enabled. Finally, the fragment in Figure 1(c) is not well-formed, because the system cannot remain in l forever, and no uncontrollable transition is enabled.*

Strategies. We consider *non-deterministic* and *memoryless* (or *positional*) strategies. Let $\perp$ denotes the null action.

A *strategy* is a function $\sigma : S \to 2^{Edg_{\mathrm{c}} \cup \{\perp\}} \setminus \emptyset$ such that:

(a) for all $s \in S$, if $e \in \sigma(s) \cap Edg_{\mathrm{c}}$, then there exists $s' \in S$ such that $s \xrightarrow{e} s'$;



(a) Well-formed. (b) Well-formed.

(c) Not well-formed.

Figure 1: Three LHA fragments.

(b) if $\perp \in \sigma(s)$, for all $f \in Adm(s)$ there exists $\delta > 0$ such that for all $0 < \delta' < \delta$ it holds $\delta' \notin span(f, loc(s))$ or $\perp \in \sigma(\langle loc(s), f(\delta')\rangle)$.

Condition (a) ensures that a strategy can only choose transitions allowed by the automaton. Condition (b) requires that if a strategy chooses the null action, then it must continue to do so for a positive amount of time along each activity that remains in the invariant. This ensures that the null action is enabled in right-open regions, so that there is an earliest instant in which a controllable transition becomes mandatory.

Notice that a strategy can always choose the null action. The well-formedness condition ensures that the system can always evolve in some way, be it a timed step or an uncontrollable transition. In particular, even if we are on the boundary of the invariant we allow the controller to choose the null action, because, in our interpretation, it is not the responsibility of the controller to ensure that the invariant is not violated.

We say that a run like (1) is *consistent* with a strategy σ if for all $0 \leq i < len(r)$ the following conditions hold:

- for all $\delta \geq 0$ such that $\sum_{j=0}^{i-1} \delta_j \leq \delta < \sum_{j=0}^{i} \delta_j$, we have $\perp \in \sigma(\langle loc(s_i), f_i(\delta - \sum_{j=0}^{i-1} \delta_j)\rangle)$;
- if $e_i \in Edg_{\mathrm{c}}$ then $e_i \in \sigma(s'_i)$.

We denote by $Runs(s, \sigma)$ the set of runs starting from the state s and consistent with the strategy σ.

Reachability control problem. Given a hybrid automaton and a set of states $T \subseteq InvS$, the *reachability control problem* asks whether there exists a strategy σ such that, for all initial states $s \in InitS$ and all runs $r \in Runs(s, \sigma)$ it holds $States(r) \cap T \neq \emptyset$. We call the above σ a *winning strategy*.

3. THE GLOBAL SEMI-ALGORITHM

The semi-algorithm for solving the reachability control problem is based on the so-called *controllable predecessor* operator $CPre^{\mathrm{R}}(\cdot)$. For a set of states A, the operator $CPre^{\mathrm{R}}(A)$ returns the set of states from which the controller can ensure that the system reaches A within the next joint step. Based on the activity chosen by the environment, this may happen for three reasons: *(1)* at some point during the activity a controllable transition is enabled that leads into A, and all uncontrollable transitions enabled in the meanwhile also lead to A; *(2)* the activity naturally enters A, and all uncontrollable transitions enabled in the meanwhile

also lead to A; *(3)* the activity eventually leaves the invariant, and all uncontrollable transitions that are ever enabled along the activity lead to A. Notice that in case *(3)* the system is forced to reach A because, by well-formedness, an uncontrollable transition must be enabled before the activity leaves the invariant.

The three cases can be formalized as the following predicates Φ_i, on an activity f, location l, and target set A. For a set of states A and $x \in \{\mathrm{u}, \mathrm{c}\}$, let $Pre_x(A)$ be the set of states in $InvS$ where some discrete transition belonging to Edg_x is enabled, which leads to A.

$$\Phi_1(f,l,A) = \exists \delta \in span(f,l) : \langle l, f(\delta)\rangle \in Pre_{\mathrm{c}}(A) \text{ and } \forall 0 \le \delta' \le \delta : \langle l, f(\delta')\rangle \notin Pre_{\mathrm{u}}(\overline{A})$$

$$\Phi_2(f,l,A) = \exists \delta \in span(f,l) : \langle l, f(\delta)\rangle \in A \text{ and } \forall 0 \le \delta' < \delta : \langle l, f(\delta')\rangle \notin Pre_{\mathrm{u}}(\overline{A})$$

$$\Phi_3(f,l,A) = \infty \notin span(f,l) \text{ and } \forall \delta \in span(f,l) : \langle l, f(\delta)\rangle \notin Pre_{\mathrm{u}}(\overline{A})$$

We then have:

$$CPre^{\mathrm{R}}(A) = \Big\{ \langle l,u\rangle \in InvS \;\Big|\; \forall f \in Adm(\langle l,u\rangle) : \Phi_1(f,l,A) \text{ or } \Phi_2(f,l,A) \text{ or } \Phi_3(f,l,A) \Big\}.$$

In discrete games, the CPre operator used for solving reachability games is the same as the one used for the safety goal [15]. In both cases, when the operator is applied to a set of states T, it returns the set of states from which Player 1 can force the game into T in one step. In hybrid games, the situation is different: a joint step represents a possibly complex behavior, extending over a (possibly) non-zero time interval. While the CPre for reachability only requires T to be visited once during such interval, CPre for safety requires that the entire behavior constantly remains in T. Hence, in Section 3.1 we present a novel algorithm for computing $CPre^{\mathrm{R}}$.

The following theorem states the general procedure for solving the reachability control problem. Due to space constraints, we only provide a proof sketch for one direction of the theorem.

THEOREM 1. *The answer to the reachability control problem for target set $T \subseteq InvS$ is positive if and only if*

$$InitS \subseteq \mu W \,.\, T \cup CPre^{\mathrm{R}}(W), \tag{2}$$

where μW denotes the least fixpoint.

PROOF SKETCH. [***if***] Assume equation 2 holds, we shall build a winning strategy in two steps. Let

- $W_0 = T$,
- $W_\alpha = T \cup CPre^{\mathrm{R}}(W_{\alpha-1})$, for a successor ordinal α, and
- $W_\alpha = \bigcup_{\beta<\alpha} W_\beta$ for a limit ordinal α.

Moreover, let $W^* = \mu W.T \cup CPre^{\mathrm{R}}(W)$. By Knaster-Tarski theorem and the well-ordering of ordinals, if $s \in W^*$ then there exists the smallest ordinal α such that $s \in W_\alpha$. If α is a limit ordinal, by definition of W_α there exists $\beta < \alpha$ such that $s \in W_\beta$, which contradicts minimality of α. Hence α is either 0 or a successor ordinal.

Let σ be a strategy defined as follows, for all states s:

- $\bot \in \sigma(s)$ and
- for all $s \in W^* \setminus T$, let $\alpha = \beta + 1$ be the smallest ordinal such that $s \in W_\alpha$; for all $e \in Edg_{\mathrm{c}}$, we have $e \in \sigma(s)$ if and only if $s \xrightarrow{e} s'$ and $s' \in W_\beta$.

While σ is clearly a strategy, it is not necessarily a winning strategy, as it may admit consistent runs which delay a controllable action either beyond the winning set W^* or beyond its availability. We can, however, recover a winning strategy by removing the null action $\bot$ from certain states. Let σ' be any strategy which coincides with σ on all the states, except for the states $s \in W^*$ with $\sigma(s) \cap Edg_{\mathrm{c}} \neq \emptyset$, where it satisfies $\sigma'(s) \cap Edg_{\mathrm{c}} = \sigma(s) \cap Edg_{\mathrm{c}}$ and the following two conditions (a) and (b). For all $f \in Adm(s)$, let $D_{f,s} = \{\delta > 0 \mid \forall 0 \le \delta' \le \delta : \langle loc(s), f(\delta')\rangle \in W_\alpha \text{ and } \sigma(\langle loc(s), f(\delta')\rangle) \cap Edg_{\mathrm{c}} \neq \emptyset\}$:

(a) If there is $f \in Adm(s)$ such that $D_{f,s} = \emptyset$ then $\bot \notin \sigma'(s)$;

(b) For all $f \in Adm(s)$, if $D_{f,s} \neq \emptyset$ then there exists $\delta \in D_{f,s}$ such that $\bot \notin \sigma'(\langle loc(s), f(\delta)\rangle)$ and $\bot \in \sigma'(\langle loc(s), f(\delta')\rangle)$ for all $0 \le \delta' < \delta$.

Intuitively, the new strategy σ' ensures that following any activity from a state $s \in W^*$ in which some controllable action is enabled, such an action will always be taken before none of them is available and before leaving W^*. It can be proved that σ' is a winning strategy.

3.1 Computing the Predecessor Operator

In order to compute the predecessor operator, we introduce the *Must Reach While Avoiding* operator, denoted by RWA^{M}. Given a location l and two sets of variable valuations U and V, $RWA^{\mathrm{M}}_l(U,V)$ contains the set of valuations from which all continuous trajectories of the system reach U while avoiding V [3]. Formally, we have:

$$RWA^{\mathrm{M}}_l(U,V) = \Big\{ u \in Val(X) \;\Big|\; \forall f \in Adm(\langle l,u\rangle)\, \exists \delta \ge 0 : f(\delta) \in U \text{ and } \forall\, 0 \le \delta' \le \delta : f(\delta') \notin V \Big\}. \tag{3}$$

By rephrasing the definition of $CPre^{\mathrm{R}}$, we observe that $s = \langle l,u\rangle \in CPre^{\mathrm{R}}(A)$ iff all activities f starting from u reach a set of "good" points while avoiding a set of "bad" points. Good points include $C_l = Pre_{\mathrm{c}}(A)\downarrow_l$ according to $\Phi_1(f,l,A)$, $A\downarrow_l$ according to $\Phi_2(f,l,A)$, and $\overline{Inv(l)}$ according to $\Phi_3(f,l,A)$. As to the bad points, all predicates Φ_i require that the activity avoids $B_l = Pre_{\mathrm{u}}(\overline{A})\downarrow_l$, with subtle distinctions at the instant when a good point is reached. According to Φ_1, B_l must be avoided also in that instant (when C_l is reached), while Φ_2 permits the activity f to reach B_l at the same time as $A\downarrow_l$. Since satisfaction of one Φ_i is enough for an activity to comply with the requirements of $CPre^{\mathrm{R}}$, the least restrictive avoidance condition prevails, namely, $B_l \setminus A\downarrow_l$. The following lemma formalizes the above argument. We say that a set of states $A \subseteq S$ is *polyhedral* if for all $l \in Loc$, the projection $A\downarrow_l$ is a polyhedron.

[3] In ATL notation [2], we have $RWA^{\mathrm{M}}(U,V) \equiv \langle\!\langle ctr\rangle\!\rangle \overline{V}\, \mathcal{U}\, (U \wedge \overline{V})$, where ctr is the player representing the controller.

LEMMA 1. *For all polyhedral sets of states $A \subseteq InvS$, we have*

$$CPre^{\mathrm{R}}(A) = InvS \cap \bigcup_{l \in Loc} \{l\} \times RWA_l^{\mathrm{M}}\big(A\lfloor_l \cup C_l \cup \overline{Inv(l)}, B_l \setminus A\lfloor_l\big),$$

where $B_l = Pre_{\mathrm{u}}(\overline{A})\lfloor_l$ and $C_l = Pre_{\mathrm{c}}(A)\lfloor_l$.

PROOF. [$\subseteq$] Let $s = \langle l, u\rangle \in CPre^{\mathrm{R}}(A)$ and let $f \in Adm(s)$. If $\Phi_1(f,l,A)$ holds, there is $\delta \in span(f,l)$ such that $f(\delta) \in C_l$ and for all $0 \le \delta' \le \delta$ it holds $f(\delta') \notin B_l$ and hence $f(\delta') \notin B_l \setminus A\lfloor_l$, satisfying the requirements of (3). If $\Phi_2(f,l,A)$ holds, there is $\delta \in span(f,l)$ such that $f(\delta) \in A\lfloor_l$ and for all $0 \le \delta' < \delta$ it holds $f(\delta') \notin B_l$. Since $f(\delta) \notin B_l \setminus A\lfloor_l$, the requirements of (3) are satisfied again. Finally, if $\Phi_3(f,l,A)$ holds, we have $\infty \notin span(f,l)$ and $\langle l, f(\delta)\rangle \notin B_l$ for all $\delta \in span(f,l)$. Pick a time δ^* when f has left $Inv(l)$ and it has never re-entered it. Formally, we have $\delta^* \notin span(f,l)$, $f(\delta^*) \notin Inv(l)$, and $f(\delta) \in span(f,l) \cup \overline{Inv(l)}$ for all $\delta \le \delta^*$. We obtain $f(\delta^*) \in \overline{Inv(l)}$ and $f(\delta) \notin B_l$ for all $0 \le \delta \le \delta^*$, satisfying (3) once again.

[$\supseteq$] Let $l \in Loc$ and $u \in RWA_l^{\mathrm{M}}(A\lfloor_l \cup C_l \cup \overline{Inv(l)}, B_l \setminus A\lfloor_l)$. For all $f \in Adm(\langle l, u\rangle)$, let D_f be the set of all $\delta \ge 0$ such that $f(\delta) \in A\lfloor_l \cup C_l \cup \overline{Inv(l)}$ and for all $0 \le \delta' \le \delta$ it holds $f(\delta') \notin B_l \setminus A\lfloor_l$. By definition of RWA_l^{M}, we have $D_f \neq \emptyset$. Let $\delta^* = \inf D_f$ and assume for simplicity that $\delta^* \in D_f$, as the other case can be treated similarly.

For all $0 \le \delta' < \delta^*$ we have both $f(\delta') \in (\overline{A\lfloor_l} \cap \overline{C_l} \cap Inv(l))$ since $\delta' \notin D_f$, and $f(\delta') \in (\overline{B_l} \cup A\lfloor_l)$ since $\delta' < \delta^*$ and $\delta^* \in D_f$. Moreover, $(\overline{A\lfloor_l} \cap \overline{C_l} \cap Inv(l)) \cap (\overline{B_l} \cup A\lfloor_l) = \overline{B_l} \cap \overline{A\lfloor_l} \cap \overline{C_l} \cap Inv(l)$, and we can conclude that $f(\delta') \in \overline{B_l} \cap \overline{A\lfloor_l} \cap \overline{C_l} \cap Inv(l)$. If $f(\delta^*) \in A\lfloor_l$, we have $\delta^* \in span(f,l)$ and $\Phi_2(f,l,A)$. If $f(\delta^*) \in C_l$, we have $\delta^* \in span(f,l)$ again and $\Phi_1(f,l,A)$. Finally, if $f(\delta^*) \in \overline{Inv(l)}$ we have $\Phi_3(f,l,A)$. Therefore, it holds $\langle l, u\rangle \in CPre^{\mathrm{R}}(A)$. ∎

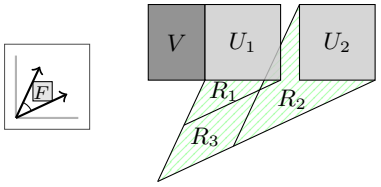
4. THE LOCAL ALGORITHM

The previous section reduces the solution of the reachability control problem to the computation of the operator RWA^{M}. Let us start by examining the basic properties of RWA^{M}.

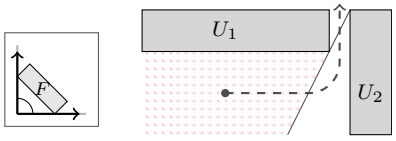
EXAMPLE 2. *As witnessed by Figure 2(a), the first argument of RWA^{M} does not distribute over union, in other words $RWA_l^{\mathrm{M}}(U_1 \cup U_2, V) \neq RWA_l^{\mathrm{M}}(U_1, V) \cup RWA_l^{\mathrm{M}}(U_2, V)$. In particular, in Figure 2(a) we have $RWA_l^{\mathrm{M}}(U_1, V) = U_1 \cup R_1$, $RWA_l^{\mathrm{M}}(U_2, V) = U_2 \cup R_2$, and $RWA_l^{\mathrm{M}}(U_1 \cup U_2, V) = U_1 \cup U_2 \cup R_1 \cup R_2 \cup R_3$. Hence, computing $RWA_l^{\mathrm{M}}(U, V)$ for convex U (a relatively simple task) does not extend to general polyhedra.*

Additionally, it is not possible to restrict the analysis from arbitrary activities (i.e., any differentiable function which stays in the invariant and whose slope belongs to $Flow(l)$) to straight-line activities. In Figure 2(b), the dotted area contains the set of points that must reach $U_1 \cup U_2$ following straight-line activities. On the other hand, $RWA_l^{\mathrm{M}}(U_1 \cup U_2, \emptyset) = U_1 \cup U_2$, because all other points (including those in the dotted area) can avoid $U_1 \cup U_2$ by passing through the gap between U_1 and U_2.

Here, we show how to compute RWA^{M} based on the operator which is used to solve *safety* control problems: the



(a) $RWA_l^{\mathrm{M}}(U_1 \cup U_2, V) \neq RWA_l^{\mathrm{M}}(U_1, V) \cup RWA_l^{\mathrm{M}}(U_2, V)$.



(b) Straight-line activities are not sufficient to avoid $U_1 \cup U_2$.

Figure 2: Basic properties of RWA^{M}. The boxes on the left represent the convex polyhedron $F = Flow(l)$ in the $(\dot{x}, \dot{y})$ plane. Thick arrows represent the *extremal directions* of flow.

May Reach While Avoiding operator $RWA_l^{\mathrm{m}}(U, V)$, returning the set of states from which *there exists* a trajectory that reaches U while avoiding V. Formally,

$$RWA_l^{\mathrm{m}}(U, V) = \Big\{ u \in Val(X) \,\Big|\, \exists f \in Adm(\langle l, u\rangle),\ \delta \ge 0 : f(\delta) \in U \text{ and } \forall\, 0 \le \delta' < \delta : f(\delta') \in \overline{V} \cup U \Big\}.$$

In safety control problems, RWA^{m} is used to compute the states from which the environment may reach an unsafe state (in U) while avoiding the states from which the controller can take a transition to a safe state (in V). Notice that RWA^{m} is a classical operator, known under different names such as *Reach* [19], *Unavoid_Pre* [4], and *flow_avoid* [20]. We recently gave the first sound and complete algorithm for computing it on LHAs in [6, 5].

In this section, we consider a fixed location $l \in Loc$. For a polyhedron G and $p \in G$, we say that p is *l-bounded in G* (resp., *l-thin in G*) if all admissible activities starting from p eventually (resp., immediately) exit from G. Formally, p is l-bounded if for all $f \in Adm(\langle l, p\rangle)$ there exists $\delta \ge 0$ such that $f(\delta) \notin G$; p is l-thin if for all $f \in Adm(\langle l, p\rangle)$ and all $\delta > 0$, it holds $f(\delta) \notin G$. We denote by $bounded_l(G)$ the set of points of G that are l-bounded in it, and we say that G is *l-bounded* (resp., *l-thin*) if all points $p \in G$ are l-bounded (resp., l-thin) in G.

EXAMPLE 3. *Consider the L-shaped polyhedron G depicted in Figure 3, where the only flow direction is upwards. Point p_1 is not l-bounded in G, because G extends indefinitely upwards from p_1. Point p_2 is l-thin (and hence l-bounded) because it sits on the upper boundary of G, and finally p_3 is l-bounded (but not l-thin) in G, as the activity that starts from p_3 eventually (but not immediately) exits from G. The gray region of G is $bounded_l(G)$.*

The following result connects RWA^{M} to RWA^{m}, by exploiting the following idea. All points in $U \setminus V$ belong to $RWA_l^{\mathrm{M}}(U, V)$ by definition. Accordingly, let us set $Under = U \setminus V$, for *under-approximation*.

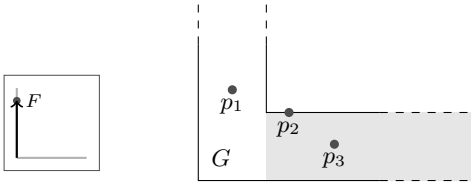


Figure 3: A non-convex polyhedron containing l-thin, l-bounded and non-l-bounded points.

Now, the content of $RWA_l^{\mathrm{M}}(U,V)$ can be partitioned into two regions: the first region is *Under*; the second region must be l-bounded, because each point in the second region must eventually reach *Under*. If we can find a polyhedron *Over* that over-approximates $RWA_l^{\mathrm{M}}(U,V)$ and such that $Over \setminus Under$ is l-bounded, we can use RWA^{m} to refine it. Precisely, we can use RWA^{m} to identify and remove the points of *Over* that may leave *Over* without hitting U first.

If $Over \setminus Under$ was not l-bounded, the above technique would not work, because RWA^{m} cannot identify (and remove) the points that may remain forever in *Over* without ever reaching *Under*.

THEOREM 2. *For all polyhedra U and V, let $Under = U \setminus V$ and let Over be a polyhedron such that:* (i) $RWA_l^{\mathrm{M}}(U,V) \subseteq Over \subseteq \overline{V}$ *and* (ii) $Over \setminus Under$ *is l-bounded. Then,*

$$RWA_l^{\mathrm{M}}(U,V) = Over \setminus RWA_l^{\mathrm{m}}(\overline{Over}, U). \tag{4}$$

PROOF. [$\subseteq$] Let $u \in RWA_l^{\mathrm{M}}(U,V)$. By assumption *(i)*, it holds $u \in Over$. We prove that $u \notin RWA_l^{\mathrm{m}}(\overline{Over}, U)$. Assume the contrary; according to the definition of RWA_l^{m}, there exist an activity $f \in Adm(\langle l, u\rangle)$ and a delay $\delta \geq 0$ such that $f(\delta) \in \overline{Over}$ and $f(\delta') \in \overline{U} \cup \overline{Over}$ for all $0 \leq \delta' < \delta$. Since $\overline{Over} \subseteq \overline{RWA_l^{\mathrm{M}}(U,V)}$, the activity f leads from u to a point in $\overline{RWA_l^{\mathrm{M}}(U,V)}$, without passing through *Under*.

Let f' be an activity witnessing the fact that $f(\delta) \notin RWA_l^{\mathrm{M}}(U,V)$. If U is never reached by f before time δ, the activity obtained by starting with f and then switching to f' from time δ is a witness for $u \notin RWA_l^{\mathrm{M}}(U,V)$ (contradiction). If instead f reaches U at time $\delta' < \delta$, it also holds $f(\delta') \in V$. Then, let $D = \{\delta' \mid f(\delta') \in V\} \neq \emptyset$ and let $\delta^* = \inf D$. For all $\delta' < \delta^*$, it holds $f(\delta') \in \overline{U}$. If $\delta^* \in D$ then $f(\delta^*) \in V$, and f is a witness to the fact that $u \notin RWA_l^{\mathrm{M}}(U,V)$ (contradiction).

Finally, if $\delta^* \notin D$, let $\bar{\delta}$ be any time when f visits U. This time must be strictly greater than δ^*. By definition of δ^*, there exists another time between δ^* and $\bar{\delta}$ where f visits V, proving once again that $u \notin RWA_l^{\mathrm{M}}(U,V)$ (contradiction). We conclude that $u \notin RWA_l^{\mathrm{m}}(\overline{Over}, U)$, and the thesis.

[$\supseteq$] Let $u \notin RWA_l^{\mathrm{M}}(U,V)$. It is immediate that $u \notin Under$. We prove that $u \notin Over \setminus RWA_l^{\mathrm{m}}(\overline{Over}, U)$. If $u \notin Over$, we are done. Hence, assume that $u \in Over$. Since $u \notin RWA_l^{\mathrm{M}}(U,V)$, there is an activity $f \in Adm(\langle l, u\rangle)$ such that for all $\delta \geq 0$ either *(a)* $f(\delta) \notin U$, or *(b)* there exists $\delta' \leq \delta$ such that $f(\delta') \in V$. We distinguish two cases:

- First, assume that the activity f never reaches U (and hence, *Under*). By assumption *(ii)*, there exists $\delta' \geq 0$ such that $f(\delta') \notin Over \setminus Under$. Since $f(\delta') \notin Under$, we conclude $f(\delta') \notin Over$. As a consequence, it holds $u \in RWA_l^{\mathrm{m}}(\overline{Over}, U)$, and we are done.
- Otherwise, let $D_U = \{\delta \geq 0 \mid f(\delta) \in U\} \neq \emptyset$ and $\delta_U = \inf D_U$. There can be two cases: first assume $\delta_U \in D_U$; by *(b)* there exists $\delta' \leq \delta_U$ with $f(\delta') \in V$. This implies that f reaches V (and hence $\overline{Over}$) at time δ' while remaining in $\overline{U}$ up until δ' (included). As a consequence, $u \in RWA_l^{\mathrm{m}}(\overline{Over}, U)$ and we are done.

  Next, assume $\delta_U \notin D_U$. Let $D_V = \{\delta \mid f(\delta) \in V\}$. We have $D_V \neq \emptyset$ due to $D_U \neq \emptyset$ and property *(b)* above. Let $\delta_V = \inf D_V$. If $\delta_V < \delta_U$, there exists a time between δ_V and δ_U when f reaches V (and hence $\overline{Over}$). Since f remains in $\overline{U}$ until δ_U, we can conclude that $u \in RWA_l^{\mathrm{m}}(\overline{Over}, U)$.

  Otherwise, $\delta_V \geq \delta_U$. For all δ' such that $f(\delta') \in U$, $\delta_V \leq \delta'$ by *(b)* and the fact that $\delta_V = \inf D_V$. As a consequence, since all possible intermediate points between δ_U and δ_V cannot belong to U, and $\delta_U = \inf D_U$, no such point exists, i.e., $\delta_V = \delta_U$.

  Now, if $\delta_V \in D_V$, then it immediately follows that $u \in RWA_l^{\mathrm{m}}(\overline{Over}, U)$. Otherwise, there are elements of D_V arbitrarily close to δ_V. Since V is a polyhedron and f is differentiable, there exists $\delta' > \delta_V$ such that $f(\delta) \in V \subseteq \overline{Over}$ for all $\delta_V < \delta \leq \delta'$. Therefore, at all times up to δ' (included), f remains in $\overline{U} \cup \overline{Over}$, once again we obtain that $u \in RWA_l^{\mathrm{m}}(\overline{Over}, U)$. ∎

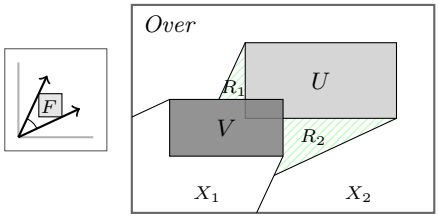


Figure 4: Relationship between RWA^{M} and RWA^{m}.

EXAMPLE 4. *An example of the application of Theorem 2 is depicted in Figure 4, where U and V are the gray boxes and Over is the outer box, excluding V. The set $RWA_l^{\mathrm{m}}(\overline{Over}, U)$ can be divided in two areas: area X_1 contains the points that may reach V (which is a part of $\overline{Over}$) while avoiding U, and area X_2 contains the points that may exit Over through its top and right sides. Following Equation 4, we remove X_1 and X_2 from Over, and we are left with the region $U \setminus V$ and the two regions R_1 and R_2, whose points are forced to enter U while avoiding V, as requested by $RWA_l^{\mathrm{M}}(U,V)$.*

4.1 Computing a Suitable Over-Approximation

Theorem 2 leaves us with one problem: We need to compute a polyhedron *Over* satisfying the assumptions of the theorem. To this purpose, we introduce the following notions.

Given a polyhedron G and a convex polyhedron F, the *positive pre-flow* operator $G \swarrow_{>0} F$ is defined as follows:

$$G \swarrow_{>0} F = \{u - \delta c \mid u \in G, c \in F, \delta > 0\}.$$

Intuitively, $G \swarrow_{>0} F$ contains the points that may reach G via a straight trajectory of non-zero length whose slope is in F. We write $G \swarrow_{>0}$ as an abbreviation for $G \swarrow_{>0} \mathit{Flow}(l)$.

For a (not necessarily convex) polyhedron G and a convex polyhedron F, we say that G is *bounded w.r.t.* F if for all $p \in G$ and all $c \in F$ there exists a constant $\delta \geq 0$ such that $p + \delta c \notin G$. Intuitively, G is bounded w.r.t. F if all straight lines starting from G and whose slope belongs to F eventually exit from G. The relationship between this definition of boundedness and the notion of l-boundedness is explored in Section 4.2.

We define the operator RU (for *Remove Unbounded*) that, given a polyhedron G, removes some convex regions of G that are not l-bounded, in such a way that the resulting set is l-bounded, and every point that was l-bounded in G belongs to the resulting set. Let B be the subset of $[\![G]\!]$ containing the convex polyhedra that are bounded w.r.t. $cl(\mathit{Flow}(l))$. We set

$$\mathrm{RU}(G) = \bigcup_{P \in B} P \cup \bigcup_{P \in [\![G]\!] \setminus B} \left(P \setminus P \swarrow_{>0}\right). \tag{5}$$

The following result summarizes the main properties of the RU operator and it is proved in Section 4.2.

THEOREM 3. *For all polyhedra G, the following hold:* (i) $\mathrm{RU}(G)$ *is l-bounded, and* (ii) $\mathit{bounded}_l(G) \subseteq \mathrm{RU}(G)$.

Given two polyhedra U and V, define $\mathit{Under} = U \setminus V$ and

$$\mathit{Over} = \mathit{Under} \cup \mathrm{RU}(\overline{U} \cap \overline{V}).$$

We prove that *Under* and *Over* satisfy the two assumptions of Theorem 2. Theorem 3 ensures that $\mathit{Over} \setminus \mathit{Under}$ is l-bounded. The following lemma proves the other assumption.

LEMMA 2. *It holds $\mathit{RWA}_l^{\mathrm{M}}(U, V) \subseteq \mathit{Over} \subseteq \overline{V}$.*

PROOF. For the first inclusion, let $u \in \mathit{RWA}_l^{\mathrm{M}}(U,V)$. If $u \in \mathit{Under} = U \setminus V$, we are done. Otherwise, $u \in \overline{U} \cup V$. Moreover, by definition of $\mathit{RWA}^{\mathrm{M}}$, $u \in \overline{V}$ (and hence $u \in \overline{U} \cap \overline{V}$) and for all activities $f \in \mathit{Adm}(\langle l, u \rangle)$ there exists $\delta \geq 0$ such that $f(\delta) \in U$. Hence, u is l-bounded in $\overline{U} \cap \overline{V}$. By property *(ii)* of Theorem 3, $u \in \mathrm{RU}(\overline{U} \cap \overline{V}) \subseteq \mathit{Over}$.

For the second inclusion, let $u \in \mathit{Over}$. If $u \in \mathit{Under} = U \setminus V$, clearly $u \notin V$. Otherwise, $u \in \mathrm{RU}(\overline{U} \cap \overline{V}) \subseteq \overline{U} \cap \overline{V} \subseteq \overline{V}$, and we are done. ∎

Section 4.3 shows how to effectively compute $\mathrm{RU}(\cdot)$, and hence *Over*, using basic operations on polyhedra. Moreover, $\mathit{RWA}_l^{\mathrm{m}}(\cdot, \cdot)$ is shown to be computable in [6, 5]. Therefore, we can compute $\mathit{RWA}_l^{\mathrm{M}}(U, V)$ using equation (4). In turn, this allows us to compute $\mathit{CPre}^{\mathrm{R}}(\cdot)$ using Lemma 1.

THEOREM 4. *For all polyhedral sets of states A, $\mathit{CPre}^{\mathrm{R}}(A)$ is computable.*

Notice that the above result provides no guarantee of termination for the global fixpoint (2). In particular, it does not imply semi-decidability of the reachability control problem, as fixpoint (2) may not be reached within ω iterations of $\mathit{CPre}^{\mathrm{R}}$.

4.2 On Bounded and Thin Polyhedra

The objective of this section is to prove the properties of the $\mathrm{RU}(\cdot)$ operator pertaining l-boundedness, which are stated by Theorem 3. Since l-boundedness is hard to directly reason about, we relate it to *geometric* boundedness, i.e., boundedness w.r.t. straight-line activities.

Let us first recall the following lemma, which is an adaptation of Lemma 4.1 in [1], and states that any point reached by an admissible trajectory can be reached with a straight-line admissible trajectory as well.

LEMMA 3 ([1]). *For all points $p \in \mathit{Inv}(l)$, activities $f \in \mathit{Adm}(\langle l, p \rangle)$ and $\delta > 0$, there exists $c \in \mathit{Flow}(l)$ such that $f(\delta) = p + \delta c$.*

The following is a trivial observation.

PROPOSITION 1. *If F is a convex polyhedron containing the origin, then no polyhedron is bounded w.r.t. F.*

We say that a polyhedron G is *thin w.r.t.* F if for all $p \in G$, $c \in F$, and $\delta > 0$, it holds $p + \delta c \notin G$. Intuitively, G is bounded (resp., thin) w.r.t. F if all straight lines starting from G and whose slope belongs to F eventually (resp., immediately) exit from G. The relationships between the geometric concepts defined in this section and the notions of l-thin and l-bounded are summarized in Figure 5.

Obviously, being thin w.r.t. F implies being bounded w.r.t. F. Moreover, being l-thin implies being thin w.r.t. $\mathit{Flow}(l)$, since straight-line activities are a special case of general activities. The following lemma shows that the converse also holds.

LEMMA 4. *For all convex polyhedra P, if P is thin w.r.t. $\mathit{Flow}(l)$ then P is l-thin.*

PROOF. Assume that P is not l-thin. Then, there exists a point $p \in P$, an activity $f \in \mathit{Adm}(\langle l, p \rangle)$ and a time $\delta > 0$ such that $f(\delta) \in P$. By Lemma 3, there exists $c \in \mathit{Flow}(l)$ such that $f(\delta) = p + \delta c \in P$. Hence, P is not thin w.r.t. $\mathit{Flow}(l)$. ∎

We shall now show with the following lemma that all points of G that are removed by $\mathrm{RU}(G)$ are not l-bounded in G (i.e., $\mathrm{RU}(G)$ does not "remove too much").

LEMMA 5. *If a convex polyhedron P is not bounded w.r.t. $cl(\mathit{Flow}(l))$ then each point in $P \cap P \swarrow_{>0}$ is not l-bounded in P.*

Next, we show that the result of $\mathrm{RU}(G)$ is l-bounded (i.e., $\mathrm{RU}(G)$ does not "remove too little"). In order to obtain this result (stated as Lemma 8), we need a few preliminary lemmata.

First, we show that if the origin does not belong to the topological closure of the flow, then there is a flow direction u such that all possible flows advance in the direction u by at least $|u|$ for each time unit. We denote by $\mathbf{0}$ the origin, i.e., the point whose coordinates are 0.

LEMMA 6. *Assume $\mathbf{0} \notin cl(\mathit{Flow}(l))$. Then there exists $u \in cl(\mathit{Flow}(l))$ such that for all $v \in \mathit{Flow}(l)$ the scalar projection of v onto u is at least $|u|$ (i.e., $\frac{u \cdot v}{|u|} \geq |u|$, where $\cdot$ denotes the inner product).*

The following fact is obvious, since straight lines are a special case of activities.

PROPOSITION 2. *If a polyhedron is l-bounded, then it is bounded w.r.t. $\mathit{Flow}(l)$.*

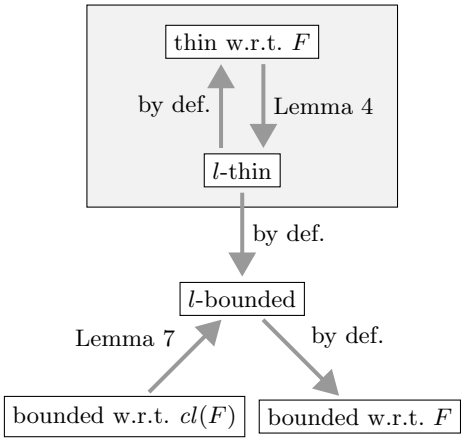


Figure 5: Relationships between properties of convex polyhedra. Arrows represent implications and $F = Flow(l)$.

Being bounded w.r.t. $Flow(l)$ is necessary but not sufficient for a polyhedron to be l-bounded, as shown by the following example.

EXAMPLE 5. *Consider the unbounded polyhedron P shown on the r.h.s. of Figure 6. The dashed contour of F (on the l.h.s. of the figure) indicates that F (i.e., $Flow(l)$) is topologically open, so that its extremal directions $(1,0)$ and $(0,1)$ are not proper (i.e., they do not belong to F). It turns out that P is bounded w.r.t. $Flow(l)$, because all straight lines whose slope belongs to F eventually exit from it, but it is not l-bounded. The figure shows an activity that remains forever in P. Its slope approaches asymptotically the extremal direction $(1,0)$.*

Lemma 7 presents a sufficient condition for being l-bounded.

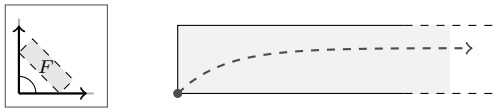


Figure 6: On the right, a polyhedron which is bounded w.r.t. $Flow(l)$ but not l-bounded, and an activity that remains forever in it.

LEMMA 7. *If a polyhedron is bounded w.r.t. $cl(Flow(l))$ then it is l-bounded.*

PROOF. Let $F = cl(Flow(l))$. By Proposition 1, F does not contain the origin. By Lemma 6, there exists $u \in F$ such that for all $v \in Flow(l)$ it holds $u \cdot v \geq |u|^2$.

Let G be a polyhedron which is bounded w.r.t. F, and let $p \in G$ and $f \in Adm(\langle l, p\rangle)$. For all $\delta \geq 0$, it holds $\dot{f}(\delta) \in Flow(l)$. From the above argument, the vector projection of $\dot{f}(\delta)$ on the direction u has length at least $|u|$. Hence, for each time unit, the activity f advances in the direction u by at least $|u|$. Since G is bounded w.r.t. $\{u\}$, we obtain the thesis. ∎

The following lemma lifts l-boundedness from convex polyhedra to general polyhedra.

LEMMA 8. *Let G be a polyhedron such that each $P \in [\![G]\!]$ is l-bounded. Then, G is l-bounded.*

PROOF OF THEOREM 3. *(i)* $\mathrm{RU}(G)$ *is l-bounded.* For a convex polyhedron P, the set $P \setminus (P \swarrow_{>0})$ is l-thin, as may be easily verified from the definitions. Hence, each convex polyhedron in $[\![\mathrm{RU}(G)]\!]$ is either bounded w.r.t. $cl(Flow(l))$ or l-thin. Since each l-thin polyhedron is l-bounded by definition, and by Lemma 7, we obtain that each convex polyhedron in $[\![\mathrm{RU}(G)]\!]$ is l-bounded. By Lemma 8, $\mathrm{RU}(G)$ is l-bounded.

(ii) $bounded_l(G) \subseteq \mathrm{RU}(G)$. Let $p \in bounded_l(G)$. Then there must be at least one convex polyhedron $P \in [\![G]\!]$ with $p \in P$. If P is bounded w.r.t. $cl(Flow(l))$ then, by equation (5), $p \in \mathrm{RU}(G)$. If, on the other hand, P is not bounded w.r.t. $cl(Flow(l))$, by Lemma 5 we have that $P \cap P \swarrow_{>0}$ is not l-bounded in P and, *a fortiori*, not l-bounded in G . Therefore, $p \in P \setminus P \swarrow_{>0}$ and, by equation (5), $p \in \mathrm{RU}(G)$. Hence the conclusion. ∎

4.3 Computing the RU Operator

As shown in the previous section, the operator RWA^{M} requires the computation of the operator RU (*Remove Unbounded*) defined by equation (5). In order to compute the operator RU, we must be able to (*i*) compute the positive-preflow $P \swarrow_{>0} F$ of a convex polyhedron P w.r.t. another convex polyhedron F, and (*ii*) collect, for any polyhedron G, the convex polyhedra $P \in [\![G]\!]$ which are bounded w.r.t. the convex polyhedron F. In the remainder of the section we shall show how these two operations can be efficiently implemented employing a canonical representation of convex polyhedra.

Any convex polyhedron admits a finite representation, in terms of *generators*. The generator representation consists in three finite sets of *points*, *closure points*, and *rays*, that generate all points in the polyhedron by linear combination. More precisely, for each convex polyhedron $P \subseteq \mathbb{R}^n$ there exists a triple (V_P, C_P, R_P) such that V_P, C_P, and R_P are finite sets of vectors in $\mathbb{R}^n$, and $x \in P$ if and only if it can be written as

$$\sum_{v \in V_P} \alpha_v \cdot v + \sum_{c \in C_P} \beta_c \cdot c + \sum_{r \in R_P} \gamma_r \cdot r, \qquad (6)$$

where all coefficients α_v, β_c and γ_r are non-negative reals, $\sum_{v \in V_P} \alpha_v + \sum_{c \in C_P} \beta_c = 1$, and there exists $v \in V_P$ such that $\alpha_v > 0$. We call the triple (V_P, C_P, R_P) a *generator system* for P.

Intuitively, the elements of V_P are the proper vertices of the polyhedron P, the elements of C_P are vertices of the topological closure of P that do not belong to P, and each element of R_P represents a direction of unboundedness (or infinity) of P.

4.3.1 Computing the Pre-Flow operator

Notice first that the positive pre-flow of P w.r.t. F is equivalent to the *positive post-flow* of P w.r.t. $-F$:

$$P \swarrow_{>0} F = P \nearrow_{>0} -F = \{x + \delta \cdot y \mid x \in P, y \in -F, \delta > 0\}.$$

The following recent result shows how to efficiently compute the positive post-flow $P \nearrow_{>0} F$ of a convex polyedron P w.r.t. another convex polyhedron F, using the generator representation. For two sets of points A and B, the *Minkowski sum* $A \oplus B$ is $\{a + b \mid a \in A, b \in B\}$.

THEOREM 5. *[7] Given two convex polyhedra P and F, let (V_P, C_P, R_P) (resp., (V_F, C_F, R_F)) be a generator system for P (resp., F). The triple $(V_P \oplus V_F, C_P \cup V_P, R_P \cup V_F \cup C_F \cup R_F)$ is a generator system for $P \not\nearrow_0 F$.*

4.3.2 Testing for boundedness w.r.t. the flow

For a convex polyhedron P, let $O_P = (\{\mathbf{0}\}, \emptyset, R_P)$ denote its *characteristic cone*, i.e., the closed polyhedron generated by the origin $\mathbf{0}$ and all the rays of P. The following theorem shows how we can effectively and efficiently test whether P is bounded w.r.t. F.

THEOREM 6. *For all convex polyhedra P and F, P is bounded w.r.t. F iff $O_P \cap F = \emptyset$.*

PROOF. [$\Rightarrow$] By hypothesis, for all $p \in P$ and for all $c \in F$ there exists $\delta \geq 0$ such that $p + \delta \cdot c \notin P$. By Proposition 1 we have that $\mathbf{0} \notin F$. Let $c \in F$, we show that $c \notin O_P$. Assume by contradiction that $c \in O_P$, we can write $c = 1 \cdot \mathbf{0} + \sum_{r \in R_p} \beta_r r = \sum_{r \in R_p} \beta_r r$. Now, let $x \in V_p$ be a vertex of P, we show that for all $\gamma \geq 0$ the point $x' = x + \gamma c$ belongs to P. Indeed, we have

$$x' = x + \gamma c = 1 \cdot x + \gamma \sum_{r \in R_p} \beta_r r = 1 \cdot x + \sum_{r \in R_p} \gamma \beta_r r.$$

Therefore, $x' \in P$, i.e. P is not bounded w.r.t. F, contradicting the hypothesis.

[$\Leftarrow$] Assume by contradiction that $c \in F \cap O_P$. By the decomposition theorem for convex polyhedra [18], since O_P is the characteristic cone of P, there exists a non-empty convex polyhedron P' such that $P = P' \oplus O_P$. Moreover, since $c \in O_P$, also $\delta c \in O_P$ for all $\delta \geq 0$. We can then conclude that for all $p' \in P'$, it holds $p' + \delta c \in P$ for all $\delta \geq 0$. Therefore, P is not bounded w.r.t. $\{c\}$ and *a fortiori* w.r.t. F. ∎

5. EXPERIMENTS WITH PHAVER+

We implemented the algorithms described in the previous sections on top of the open-source tool PHAVer[4] [11]. The following experiments were performed on an Intel Xeon (2.80GHz) PC.

The maze example. A vehicle navigates in an environment whose shape is depicted in Figure 8(a), by taking 90-degree left or right turns: the possible directions are thus North (N), South (S), West (W) and East (E). One time unit (say, second) must pass between two changes of direction, while the vehicle speed is 2 unit of length per second. The corridors of the maze are 1 unit wide, so that the vehicle can never u-turn without hitting a wall. The goal consists in reaching a target area positioned along the topmost corridor. We tested our implementation on progressively more complex mazes, by increasing downwards the number of corridors (the angle between consecutive corridors is 90-degrees). For instance, Figure 8(a) shows the shape of the maze with 5 corridors.

The LHA modeling the system with two corridors has one location for each direction, where the derivative of the position variables (x and y) are set according to the corresponding direction. Figure 7 shows the LHA fragment related to the N location. The variable t represents a clock ($\dot{t} = 1$), used to enforce a one-time-unit delay between turns. Each change of direction is modeled by a controllable transition (solid arrows in Figure 7) enabled when $t \geq 1$. The maze walls are modeled by uncontrollable transitions (dashed arrows in Figure 7). They are enabled when the variables x and y identify an invalid position (i.e., when the vehicle hits a wall) and lead to the special *Abort* location.

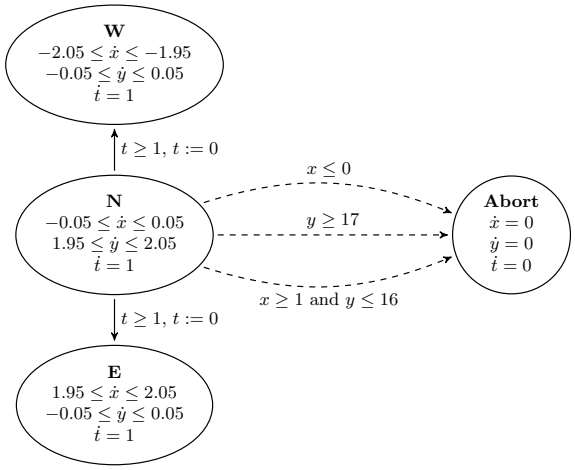


Figure 7: Fragment of the LHA modeling the maze with two corridors. The goal consists in reaching $\{2 \leq x \leq 2.5, y = 17\}$.

The maze example described above features a non-deterministic flow, allowing some uncertainity on the exact direction taken by the vehicle. A deterministic version has also been considered, where environmental disturbances are disallowed. This version can be obtained by, e.g., replacing the differential equations for x and y in location N with the differential constraints $\dot{x} = 0$ and $\dot{y} = 2$.

Figure 8(a) shows the cross-section of the solution for $t = 0$, in the case of a maze with two corridors and the vehicle initially going along the *North* direction: the gray areas A and B in the l.h.s. of the figure contain the points that can reach the target T in the case of deterministic flow. If the vehicle is located in A, it can reach the target by turning *East* and then *North* again. Notice that the area A covers only half the width of the vertical corridor. In fact, if the vehicle is located in the other half of the corridor, when turning *East* it will be too close to the target and it will not be able to take the second turn towards the target in time. The area A ends 2 units of length before the north wall, as beyond that the vehicle cannot avoid hitting the wall before being able to turn *East*. Finally, the points in the area B are trivially winning, as they can reach the target by proceeding *North*.

The solution in the corresponding non-deterministic case is shown in the r.h.s. of Figure 8(a). Notice that the winning region A' is contained in the region A and similarly B' is contained in B. Due to the uncertainty in the vehicle direction, both winning regions become gradually smaller as we move away from the target T.

We also experimented with a three-dimensional version of the maze. In addition to the vehicle directions of the 2D case, in the 3D version the vehicle can perform 90-degree turns upwards (U) and downwards (D). The resulting LHA

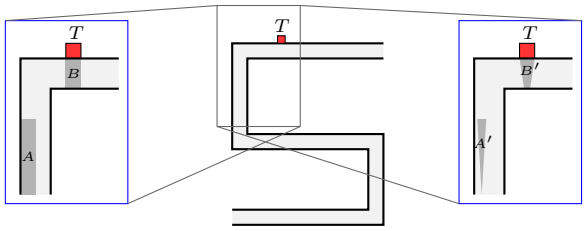
[4]A binary pre-release of our implementation, called PHAVer+, can be downloaded at `http://people.na.infn.it/mfaella/phaverplus`.

includes two additional locations to move up or down, and one additional continuous variable z for the position of the vehicle along the third dimension, for a total of four variables. The flows associated to each location in the non-deterministic case are shown in Table 1.

N	**W**	**E**
$-0.05 \leq \dot{x} \leq 0.05$ $1.95 \leq \dot{y} \leq 2.05$ $\dot{z} = 0$	$-2.05 \leq \dot{x} \leq -1.95$ $-0.05 \leq \dot{y} \leq 0.05$ $\dot{z} = 0$	$1.95 \leq \dot{x} \leq 2.05$ $-0.05 \leq \dot{y} \leq 0.05$ $\dot{z} = 0$
S	**U**	**D**
$-0.05 \leq \dot{x} \leq 0.05$ $-2.05 \leq \dot{y} \leq -1.95$ $\dot{z} = 0$	$-0.05 \leq \dot{x} \leq 0.05$ $-0.05 \leq \dot{x} \leq 0.05$ $\dot{z} = 2$	$-0.05 \leq \dot{x} \leq 0.05$ $-0.05 \leq \dot{y} \leq 0.05$ $\dot{z} = -2$

Table 1: Flows associated to each location in the 3D version of maze. In all locations it holds $\dot{t} = 1$.

The table in Figure 8(b) shows the run time in seconds for the four different versions of the maze of increasing size, in terms of number of corridors. Although still limited in scope, the results show that the proposed approach is practical, at least for relatively small problems.



(a) Structure of the maze and controllable region for the deterministic (left) and non-deterministic (right) case.

# of corridors	2D	2D NDet	3D	3D NDet
2	0.4	1.1	1.3	4.2
3	0.8	9.6	2.1	25.4
4	1.3	15.5	3.6	47.7
5	3.4	149.5	8.9	337.0
6	5.1	216.2	13.1	394.9

(b) Performance in seconds.

Figure 8: The maze example.

6. REFERENCES

[1] R. Alur, T. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. Softw. Eng.*, 22:181–201, March 1996.

[2] R. Alur, T. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *38th IEEE Symp. Found. of Comp. Sci.*, pages 100–109. IEEE Computer Society Press, 1997.

[3] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli. Effective synthesis of switching controllers for linear systems. *Proceedings of the IEEE*, 88(7):1011–1025, 2000.

[4] A. Balluchi, L. Benvenuti, T. Villa, H. Wong-Toi, and A. Sangiovanni-Vincentelli. Controller synthesis for hybrid systems with a lower bound on event separation. *Int. J. of Control*, 76(12):1171–1200, 2003.

[5] M. Benerecetti, M. Faella, and S. Minopoli. Automatic synthesis of switching controllers for linear hybrid automata. Technical report, Università di Napoli "Federico II", 2011. arXiv:1103.4584.

[6] M. Benerecetti, M. Faella, and S. Minopoli. Revisiting synthesis of switching controllers for linear hybrid systems. In *Proc. of the 50th IEEE Conf. on Decision and Control*. IEEE, 2011.

[7] M. Benerecetti, M. Faella, and S. Minopoli. Towards efficient exact synthesis for linear hybrid systems. In *GandALF 11: 2nd Int. Symp. on Games, Automata, Logics and Formal Verification*, volume 54 of *Elec. Proc. in Theor. Comp. Sci.*, 2011.

[8] P. Bouyer, T. Brihaye, and F. Chevalier. O-minimal hybrid reachability games. *Logical Methods in Computer Science*, 6, 2010.

[9] L. de Alfaro, M. Faella, T. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *CONCUR 03: Concurrency Theory. 14th Int. Conf.*, volume 2761 of *Lect. Notes in Comp. Sci.*, pages 144–158. Springer, 2003.

[10] L. de Alfaro, T. Henzinger, and R. Majumdar. Symbolic algorithms for infinite-state games. In *CONCUR 01: Concurrency Theory. 12th Int. Conf.*, Lect. Notes in Comp. Sci. Springer, 2001.

[11] G. Frehse. PHAVer: Algorithmic verification of hybrid systems past hyTech. In *Proc. of Hybrid Systems: Computation and Control (HSCC), 8th International Workshop*, volume 3414 of *Lect. Notes in Comp. Sci.*, pages 258–273. Springer, 2005.

[12] T. Henzinger. The theory of hybrid automata. In *11th IEEE Symp. Logic in Comp. Sci.*, pages 278–292, 1996.

[13] T. Henzinger, B. Horowitz, and R. Majumdar. Rectangular hybrid games. In *CONCUR 99: Concurrency Theory. 10th Int. Conf.*, volume 1664 of *Lect. Notes in Comp. Sci.*, pages 320–335. Springer, 1999.

[14] T. Henzinger, P. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Proc. of the 27th annual ACM symposium on Theory of computing*, STOC '95, pages 373–382. ACM, 1995.

[15] O. Maler. Control from computer science. *Annual Reviews in Control*, 26(2):175–187, 2002.

[16] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *12th Annual Symp. on Theor. Asp. of Comp. Sci.*, volume 900 of *Lect. Notes in Comp. Sci.* Springer, 1995.

[17] P. Ramadge and W. Wonham. Supervisory control of a class of discrete-event processes. *SIAM Journal of Control and Optimization*, 25:206–230, 1987.

[18] A. Schrijver. *Theory of linear and integer programming*. John Wiley and Sons, 1986.

[19] C. Tomlin, J. Lygeros, and S. Shankar Sastry. A game theoretic approach to controller design for hybrid systems. *Proc. of the IEEE*, 88(7):949–970, 2000.

[20] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *36th IEEE Conf. on Decision and Control*, pages 4607 – 4612, San Diego, CA, 1997. IEEE.