Non-Convex Invariants and Urgency Conditions on Linear Hybrid Automata

Stefano Minopoli and Goran Frehse

VERIMAG, Centre Équation - 2, avenue de Vignate, 38610 GIÉRES
{stefano.minopoli,goran.frehse}@imag.fr

Abstract. Linear hybrid automata (LHAs) are of particular interest to formal verification because sets of successor states can be computed exactly, which is not the case in general for more complex dynamics. Enhanced with urgency, LHA can be used to model complex systems from a variety of application domains in a modular fashion. Existing algorithms are limited to convex invariants and urgency conditions that consist of a single constraint. Such restrictions can be a major limitation when the LHA is intended to serve as an abstraction of a model with urgent transitions. This includes deterministic modeling languages such as Matlab-Simulink, Modelica, and Ptolemy, since all their transitions are urgent. The goal of this paper is to remove these limitations, making LHA more directly and easily applicable in practice. We propose an algorithm for successor computation with non-convex invariants and closed, linear urgency conditions. The algorithm is implemented in the open-source tool PHAVer, and illustrated with an example.

1 Introduction

Linear Hybrid Automata (LHA) are discrete automata enhanced with real-valued variables and linear constraints [16]. Despite their syntactical simplicity, they admit a rich variety of behaviors. In LHA, the evolution of the variables over time is governed by differential inclusions, called *flows*, which can be simple intervals such as $\dot{x} \in [1, 2]$, or more complex linear constraints over the derivatives such as the conservation law $\dot{x} + \dot{y} = 0$. Changes of the discrete state admit arbitrary linear updates of the variables. For example, LHA can model discrete-time affine systems, a widely used class of control systems, by using discrete updates of the form $x^+ = Ax + b$. LHA can even generate chaotic behavior that can be observed in real-life production systems [23]. The chaos can be due to switching flows [9] or due to updates of the discrete state, with which one can model piecewise affine maps such as the tent map [10].

Linear Hybrid Automata belong to the very few classes of hybrid systems for which set-based successor computations can be carried out exactly [1]. This makes them prime candidates for formal verification. LHA can serve as abstractions of systems that require not only timed behavior but quantitative information, e.g., to capture accumulation effects. The LHA abstraction can then be verified using model checkers such as HyTech [15] or PHAVer [13]. If the abstraction is conservative, verifying it implies that the real system satisfies the specification; if the abstraction is an approximation that is not entirely conservative, its verification helps to find bugs and identify pertinent test cases.

In model-based design, the basis for building LHA abstractions is often an existing model, given in formats like Matlab-Simulink/Stateflow [19, 18] or Modelica [20], which are the de-facto standard in many industries. Like the academic formalism Ptolemy [8], the semantics of these models are deterministic. In particular, a discrete transition is taken as soon as it is enable, which is also referred to as urgent or as-soon-as-possible (ASAP) semantics. This can pose a problem when trying to build a corresponding LHA model, since LHA transitions do not force the system to change state when they are enabled. In particular, if the derivatives of the system happen to be zero when the guard is enabled, the system may remain forever at that state. The only way to circumvent this problem is to add a clock to the controller model and periodically test (with a self-loop transition) whether the constraint is satisfied or not. This is a formally correct and conservative way to model such a system, and it even corresponds quite closely to actual behavior of process controllers, which periodically sample the sensors and set actuators. But it can tremendously increase the computational complexity of the verification task: The clock ticks introduce discrete state changes at a rate much higher than the time constants of the system, multiplying the number of sets of states that need to be computed. A computationally more efficient abstraction can be obtained if one adds *urgency* conditions to the LHA formalism. Declaring certain states of the controller as urgent prevents time from elapsing, and one can now construct an LHA abstraction (or approximation) of deterministic transitions.

Existing algorithms for set-based successor computations of LHA require urgency conditions to either be independent of the continuous variables [15] or consist of a single constraint [13], which can be quite restrictive in practice. In this paper, we propose an algorithm to compute successor states for arbitrary, non-convex, closed urgency conditions. To be able to do so, we also propose an algorithm for computing successor states for general non-convex invariants, for which so far no algorithm is available. Related work is discussed in more detail for non-convex invariants in Sect. 2.4 and for urgency in Sect. 3.5.

The proposed algorithms are implemented in the open-source tool PHAVer on the SpaceEx tool platform [12]. The tool as well as all examples from this paper are available for download at `spaceex.imag.fr`. Detailed proofs are available in a technical report [21].

In the next section, we recall the basics on LHA and then propose our post operator for non-convex invariants. In Sect. 3, we propose our post operator for urgency conditions and make the connection to urgent transitions. The computation of reachable states with these operators is illustrated by examples in Sect. 4. The paper terminates with some conclusions in Sect. 5.

2 Linear Hybrid Automata with Non-Convex Invariants

In this section, we give the syntax and the semantics description of a particular case of *Linear Hybrid Automata (LHA)*, where it is possible to define, for each location, a non-convex invariant.

2.1 Definition and Semantics

We first need to define some notation. 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 sake of clearity, given a convex polyhedron P, we will write $\widehat{P}$ to direct explicit that P is convex. For a general (i.e., not necessarily convex) polyhedron $G \subseteq \mathbb{R}^n$, we denote by $cl(G)$ its topological closure. Given an ordered set $X = \{x_1, \ldots, x_n\}$ of variables, a *valuation* is a function $v : X \to \mathbb{R}$. Let $\mathit{Val}(X)$ denote the set of valuations over X. There is an obvious bijection between $\mathit{Val}(X)$ and $\mathbb{R}^n$, allowing us to extend the notion of (convex) polyhedron to sets of valuations. We denote by $\mathit{CPoly}(X)$ (resp., $\mathit{Poly}(X)$) the set of convex polyhedra (resp., polyhedra) on X. 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 discrete transition. Arithmetic operations on valuations are defined in the straightforward way. An *activity* over X is a function $f : \mathbb{R}^{\geq 0} \to \mathit{Val}(X)$ that is continuous on its domain and differentiable except for a finite set of points. Let $\mathit{Acts}(X)$ denote the set of activities over X. The *derivative* $\dot{f}$ of an activity f is defined in the standard way and it is a partial function $\dot{f} : \mathbb{R}^{\geq 0} \to \mathit{Val}(\dot{X})$.

A *Linear Hybrid Automaton* $H = (\mathit{Loc}, X, \mathit{Lab}, \mathit{Edg}, \mathit{Flow}, \mathit{Inv}, \mathit{Init})$ consists of the following:

- a finite set Loc of *locations*,
- a finite set $X = \{x_1, \ldots, x_n\}$ of real-valued *variables*, A *state* is a pair $\langle l, v \rangle$ of a location l and a valuation $v \in \mathit{Val}(X)$;
- a finite set of labels Lab,
- a finite set Edg of *discrete transitions* that describes instantaneous changes of locations, in the course of which variables may change their value. Each transition $(l, \alpha, \eta, l') \in \mathit{Edg}_c$ consists of a *source location* l, a *target location* l', a label $\alpha \in \mathit{Lab}$, and a *jump relation* $\eta \in \mathit{Poly}(X \cup X')$, that specifies how the variables may change their value during the transition. The *guard* is the projection of η on X and describes the valuations for which the transition is enabled;
- a mapping $\mathit{Flow} : \mathit{Loc} \to \mathit{CPoly}(\dot{X})$ 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 $\mathit{Inv} : \mathit{Loc} \to \mathit{Poly}(X)$, called the *invariant*;
- a mapping $\mathit{Init} : \mathit{Loc} \to \mathit{Poly}(X)$, contained in the invariant, defining the *initial states* of the automaton.

The set of states of H is $S = \mathit{Loc} \times \mathit{Val}(X)$. Moreover, we use the shorthand notations $\mathit{InvS} = \bigcup_{l \in \mathit{Loc}} \{l\} \times \mathit{Inv}(l)$ and $\mathit{InitS} = \bigcup_{l \in \mathit{Loc}} \{l\} \times \mathit{Init}(l)$. Given a set of states A and a location ℓ, we denote by $A \downharpoonright_{\ell}$ the projection of A on ℓ, i.e. $A \downharpoonright_{\ell} = \{v \in \mathit{Val}(X) \mid \langle \ell, v \rangle \in A\}$.

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

Given a state $s = \langle l, v \rangle$, we set $\mathit{loc}(s) = l$ and $\mathit{val}(s) = v$. An activity $f \in \mathit{Acts}(X)$ is called *admissible from* s if *(i)* $f(0) = v$ and *(ii)* for all $\delta \geq 0$, if $\dot{f}(\delta)$ is defined then $\dot{f}(\delta) \in \mathit{Flow}(l)$. An activity is *linear* if there exists a constant slope $c \in \mathit{Flow}(l)$ such that, for all $\delta \geq 0$, $\dot{f}(\delta) = c$. We denote by $\mathit{Adm}(s)$ the set of activities that are admissible from s.

Runs Given two states s, s', and a transition $e \in \mathit{Edg}$, there is a *discrete step* $s \xrightarrow{e} s'$ with *source* s and *target* s' iff *(i)* $s, s' \in \mathit{InvS}$, *(ii)* $e = (\mathit{loc}(s), \alpha, \eta, \mathit{loc}(s'))$, and *(iii)* $(\mathit{val}(s), \mathit{val}(s')[X'/X]) \in \eta$, where $\mathit{val}(s')[X'/X]$ is the valuation in $\mathit{Val}(X')$ obtained from s' by renaming each variable in X with the corresponding primed variable in X'. Whenever condition *(iii)* holds, 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 $f \in \mathit{Adm}(s)$ iff *(i)* $s \in \mathit{InvS}$, *(ii)* for all $0 < \delta' \leq \delta$, $(\langle l, f(\delta') \rangle) \in \mathit{InvS}$, and *(iii)* $s' = \langle \mathit{loc}(s), f(\delta) \rangle$. For technical convenience, we admit timed steps of duration zero. A special timed step is denoted by $s \xrightarrow{\infty, f}$ and represents the case when the system follows the activity f forever. This is allowed only if for all $\delta \geq 0$, $(\langle l, f(\delta) \rangle) \in \mathit{InvS}$. A *run* is a sequence

$$r = s_0 \xrightarrow{\delta_0, f_0} s_0' \xrightarrow{e_0} s_1 \xrightarrow{\delta_1, f_1} s_1' \xrightarrow{e_1} s_2 \cdots s_n \cdots \tag{1}$$

of alternating timed and discrete steps, such that either the sequence is infinite, or it ends with a timed step of the type $s_n \xrightarrow{\infty, f}$.

If the run r is finite, we define $\mathit{len}(r) = n$ to be the length of the run, otherwise we set $\mathit{len}(r) = \infty$. Given a hybrid automaton H, the set $\mathit{Runs}(H)$ contains all the possible runs induced by the automaton H.

2.2 Reachability

Given a state $s \in S$ and a hybrid automaton H with initial set of states *Init*, s is said to be *reachable* in H if there exists a finite run $r = s_0 \xrightarrow{\delta_0, f_0} s_0' \xrightarrow{e_0} s_1 \xrightarrow{\delta_1, f_1} s_1' \xrightarrow{e_1} s_2 \cdots s_n$, such that $s_0 \in \mathit{Init}$ and $s_n = s$. In a natural way, we can extend the concept of reachability to the valuations: considering the run r explicited before, if $s_0 = (\langle l, u \rangle)$ and $s_n = (\langle l', v \rangle)$, we can say that the valuation v is reachable from the valuation u. The *reachability problem* for the automaton

H consists in the computation of the set of states

$$\mathit{Reach}(H) = \big\{s \in S \,\big|\, \exists r \in \mathit{Runs}(H) : \mathit{len}(r) \neq \infty, r = s_0 \xrightarrow{\delta_0, f_0} \cdots s_n, \\ \text{where } s_0 \in \mathit{Init} \text{ and } s_n = s\big\}.$$

Classically, the algorithm that computes the set $\mathit{Reach}(H)$ is a fixed-point procedure, over all the locations $l \in \mathit{Loc}$, based on the *continuous post operator* and on the *discrete post operator*: given a set of states $S' \subseteq S$, the first one operator is used to compute the set of states reachable from S' by following an admissible trajectory, while the second one operator is used to compute the set of states reachable from S' via discrete transitions. Notice that the computation of the discrete post operator is not affected by the nature of the invariants, so we focus on the continuous post operator. The formal definitions are as follows:

Definition 1 (Post operators). *Given an hybrid automaton H, a location $\ell \in \mathit{Loc}$, a set of valuations $P, I \subseteq \mathit{Inv}(\ell)$, the* continuous post operator $\mathit{Post}_\ell(P, I)$ *contains the set of all valuations $v \in \mathit{Val}(X)$ reachable from some $u \in P$ without leaving I:*

$$\mathit{Post}_\ell(P, I) = \big\{v \in \mathit{val}(X) \big|\, \exists u \in P, f \in \mathit{Adm}(\langle l, u\rangle) \text{ and } \delta \geq 0 : \\ \forall 0 < \delta' \leq \delta, f(\delta') \in I \quad \text{and } f(\delta) = v\big\}. \quad (2)$$

The discrete post operator $\mathit{Post}_\varepsilon(P)$ *contains the set of all valuations $v \in \mathit{Val}(X)$ reachable from some $u \in P$ by taking the discrete transition $\varepsilon = (\ell, \eta, \ell')$:*

$$\mathit{Post}_\varepsilon(P) = \big\{v \in \mathit{val}(X) \big|\, \exists u \in P, (u, v[X'/X]) \in \eta \text{ and } v \in \mathit{Inv}(\ell')\big\}.$$

From these operators on valuations we obtain the continuous and discrete post operators for a set of states S by iterating over all locations and transitions:

$$\mathit{Post}_c(S) = \bigcup_{\ell \in \mathit{Loc}} \{\ell\} \times \mathit{Post}_\ell(S\downharpoonright_\ell, \mathit{Inv}(\ell)), \quad \mathit{Post}_d(S) = \bigcup_{(\ell,\alpha,\eta,\ell') \in \mathit{Edg}} \{\ell'\} \times \mathit{Post}_\varepsilon(S\downharpoonright_\ell).$$

Note that definition (2) is valid regardless whether I is convex or not. It differs slightly from the classic definition in that we do not require that $P \subseteq I$. This trick is used in the next section to apply the operator iteratively to convex partitions of a non-convex invariant. In this case, I is a convex subset of the invariant but P is not necessarily a subset of I. For the sake of clarity, we will denote by $\mathit{Post}_\ell(P, I)$ the continuous post operator when I is convex and by $\mathit{ncPost}_\ell(P, I)$ when I is non-convex.

The following simple algorithm is used by tools such as HyTech and PHAVer to compute the reachable states. Starting from the initial states, it computes the continuous and discrete post in alternation until a fixed point is reached. Note that this is a semi-algorithm, since it may not terminate. The algorithm computes the sequence $S_0 = \mathit{Post}_c(\mathit{InitS})$, and $S_{k+1} = S_k \cup \mathit{Post}_c(\mathit{Post}_d(S_k))$. It terminates if $S_{k+1} = S_k$, with the result that $\mathit{Reach}(H) = S_k$.

2.3 Computing the Continuous Post Operator with Nonconvex Invariants

In this section we first recall how the continuous post operator is computed, when the invariant is a convex polyhedron, and why it is not possible to use the same way when the invariant is non-convex. Then we give a sound and complete procedure that, given a non-convex invariant I and an initial set of valuation $P \subseteq I$, computes the continuous post operator $ncPost_\ell(P, I)$. Given a linear hybrid automaton H, it is well known that the continuous post operator, on a location $l \in Loc$, a convex invariant $I = Inv(\ell)$, a flow $F = Flow(\ell)$ and a set of initial valuations $P \subseteq Inv(\ell)$, is given by:

$$Post_\ell(P, I) = (P\nearrow_F) \cap I, \tag{3}$$

where $P\nearrow_F$ are valuations on straight line trajectories starting in P with constant derivative $\dot{x} = c$ for any $c \in F$:

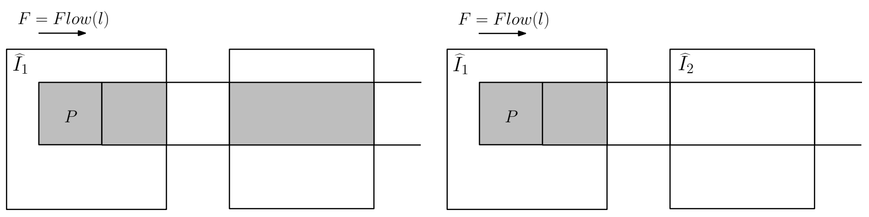
$$P\nearrow_F = \{x' \mid x \in P, c \in F, t \in \mathbb{R}^{\geq 0}, x' = x + ct\}. \tag{4}$$

The operator (4) is straightforward to compute for polyhedral sets, and is available in computational geometry libraries such as the Parma Polyhedra Library (PPL) [2]. The result of (4) is a convex polyhedron if P, F are convex polyhedra and F is closed and bounded; otherwise, it is the union of P with a convex polyhedron [7].

The natural question now is: what happens if (3) is applied when I is a non-convex polyhedron? Example 1 illustrates the answer with a simple case.

Example 1. Given a LHA with non-convex polyhedra invariants $H = (Loc, X, Edg, Flow, Inv, Init)$, and fix $l \in Loc$, $I = \widehat{I_1} \cup \widehat{I_2} = Inv(\ell)$, $P \subseteq I$, and $F = Flow(\ell)$, Figure 1 shows the comparison between the correct result of $ncPost_\ell(P, I)$ and the result obtained by forcing the usage of the classical post operator also with non-convex invariant (that is by computing $P\nearrow_F \cap I$). The gray area in Figure 1(a) contains all the valuations coming from (3). Notice that, in the resulting set there are also valuations belonging to $\widehat{I_2}$. But, according to the definition of continuous post, this is not correct because all the valuations reached during the evolution of the time have to remain in the invariant: in this case, for all the admissible activities in P, the only way to reach $\widehat{I_2}$, is to cross the area between $\widehat{I_1}$ and $\widehat{I_2}$, that is clearly not in the invariant and then the valuations in $Post_\ell(P, I) \cap \widehat{I_2}$ do not belong to $ncPost_\ell(P, I)$. The gray area of Figure 1(b) contains, instead, all the valuations that properly belong to the $ncPost_\ell(P, I)$: starting from any valuations in P, the evolution can proceed only until the right border of $\widehat{I_1}$ is reached, and there is no way to reach $\widehat{I_2}$ without crossing $\overline{I}$.

Example 1 can also give an intuition about how to tackle the issues of the classical approach: in order to compute $ncPost_\ell(P, I)$, it is necessary to consider not the global I, but the single convex polyhedra in $[\![I]\!]$, in a kind of fix-point procedure.



(a) $(P\nearrow_F) \cap I$. (b) The correct result.

Fig. 1. Comparison between $Post_\ell(P, I)$ and $ncPost_\ell(P, I)$, with non-convex invariant $I = Inv(\ell) = \widehat{I}_1 \cup \widehat{I}_2$.

Considering again Figure 1, the first step is to compute $P_1 = P\nearrow_F \cap \widehat{I}_1$. Once obtained the result, we need to check if there exists a valuation $u \in P_1$ such that, by following some admissible activity $f \in Adm(\langle \ell, u \rangle)$, it is possible to reach a valuation $v \in \widehat{I}_2$, while the system always remains in the invariant (i.e. while avoiding $\overline{I}$). In the case depicted in Figure 1, does not exist such a valuation (indeed, in order to reach $\widehat{I}_2$ via an admissible activity, it is always necessary to cross $\overline{I}$).

Before giving the fixed point characterization of $ncPost_\ell$, we need to introduce some extra notation (some of them similar to operators defined in [7]). The topological closure of a set G is denoted by $cl(G)$. Given polyhedra A and B, their *boundary* is

$$bndry(A, B) = (cl(A) \cap B) \cup (A \cap cl(B)). \tag{5}$$

Clearly, $bndry(A, B)$ is nonempty only if A and B are adjacent to one another or they overlap; otherwise, it is empty.

Definition 2 (Potential entry). *Given a location ℓ and convex polyhedra A and B, the potential entry region from A to B denotes the set of points on the boundary between A and B that may reach B by following some linear activity in location ℓ, while always remaining in $A \cup B$:*

$$pentry_\ell(A, B) = \big\{p \in bndry(A, B) \mid \exists q \in A, \delta \geq 0 \text{ and } c \in Flow(\ell) : \\ p = q + \delta \cdot c \text{ and for all } 0 \leq \delta' < \delta,\ q + \delta' \cdot c \in A\big\}. \tag{6}$$

We call the above set the "potential" entry because it may happen that, even though $pentry_\ell(A, B)$ is not empty, the system is not able to reach valuations in B starting from a valuation in A (see Example 2, Fig. 2(c), Fig. 2(f)). The following Lemma gives us a way to effectively compute the potential entry region.

Lemma 1. *Given a location ℓ and convex polyhedra A and B, let $F = Flow(\ell)$, the potential entry region from A to B can be computed by:*

$$pentry_\ell(A, B) = bndry(A, B) \cap A\nearrow_F .$$

Proof ($\subseteq$). Let $p \in \mathit{pentry}_\ell(A,B)$, by definition of potential entry we have that $p \in \mathit{bndry}(A,B)$ and there exists a valuation $q \in A$, $c \in F$ and a time $\delta \geq 0$, such that $p = q + \delta \cdot c$ and for all $0 \leq \delta' < \delta$, it holds that $q + \delta' \cdot c \in A$. The last one, recalling that $q \in A$ and the time elapse-definition, allows us to write that for all $0 \leq \delta' < \delta$, it holds that $q + \delta' \cdot c \in A\nearrow_F$. Again, by definition of time elapse, if for all $0 \leq \delta' < \delta$, then $q + \delta' \cdot c \in A\nearrow_F$ and also $p = q + \delta \cdot c \in A\nearrow_F$. Hence, $p \in \mathit{bndry}(A,B)$ and $p \in A\nearrow_F$ trivially implies that $p \in \mathit{bndry}(A,B) \cap A\nearrow_B$.

[$\supseteq$] Let $p \in \mathit{bndry}(A,B) \cap A\nearrow_F$, by definition of time-elapse we have that there exists a valuation $q \in A$, a point $c \in F$ and a time $\delta \geq 0$ such that $p = q + \delta \cdot c$ or, equivalently there exists an activity $f \in \mathit{Adm}(\langle l, u\rangle)$ such that $p = f(\delta) \in \mathit{bndry}(A,B)$. Due to the convexity of A and the definition of boundary, we have that for all $0 \leq \delta' < \delta$, $f(\delta') \in A$. The last one, coupled with $f(\delta) \in \mathit{bndry}(A,B)$, allows us to conclude $p \in \mathit{pentry}_\ell(A,B)$. □

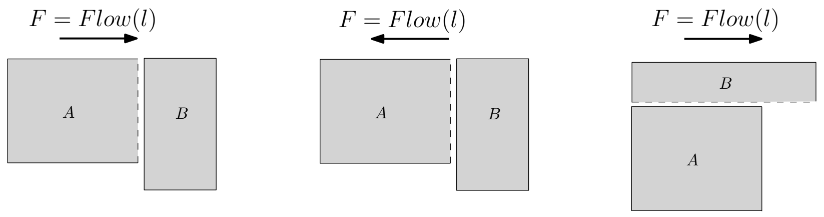
From Lemma 1 follows the following Corollary:

Corollary 1. *If $A \subseteq B$, then $A \subseteq \mathit{pentry}_\ell(A,B) \subseteq \mathit{cl}(A)$.*

Example 2. Figure 2 shows two convex polyhedra A and B whose boundary is non empty (A and B are adjacent), where flow is represented by an arrow. Considering Figure 2(a), it is easy to check that the flow allows to reach valuations on the boundary between A and B starting from a valuation belongs to A. Figure 2(d) shows the computation of $\mathit{pentry}_\ell(A,B)$, obtained by performing the intersection between $\mathit{bndry}(A,B)$ and $A\nearrow_F$. Considering instead the case depicted in Figure 2(b) (same as the previous one except for the flow), there is no way to reach valuations belong to $\mathit{bndry}(A,B)$ starting from a valuation $u \in A$: according to Lemma 1, Figure 2(e) shows that the intersection between the boundary and the post-flow is empty. Figure 2(c) shows a case where the polyhedron B is not closed and then by following the flow, the system can never reach B, even if the starting valuation is on the top border of A. Notice that (see Figure 2(f)), even if B is not reachable from A, we have that $\mathit{pentry}_\ell(A,B) \neq \emptyset$: this clearifies why we denote this set as "potential".

Now we are ready to give a way to correctly compute the continuous post operator when the invariants could be non-convex. Given a LHA H and let $l \in \mathit{Loc}$, $I = \mathit{Inv}(\ell)$, $F = \mathit{Flow}(\ell)$ and $P \subseteq I$, as Example 1 suggests, the idea is to build incrementally the sets of reachable valuations by considering each time a single convex component $\widehat{I}' \in [\![I]\!]$ instead of considering the entire invariant I. The procedure starts by finding, for all $\widehat{I}' \in [\![I]\!]$ and $\widehat{P}' \in [\![P]\!]$, the potential entry from $\widehat{P}'$ to $\widehat{I}'$. Once obtained the set $\mathit{pentry}_\ell(\widehat{P}', \widehat{I}')$, the procedure computes the classical continuous post operator on $\mathit{pentry}_\ell(\widehat{P}', \widehat{I}')$ and $\widehat{I}'$. The procedure is applied recursively by building the sequence $W_0 \subseteq W_1 \subseteq \ldots W_{i-1} = W_i$ of the sets of the reachable valuations, with $W_0 = P$, and ends when no new valuation can be added to a set. When this happens, we have that $\mathit{ncPost}_\ell(P,I) = W_i$.

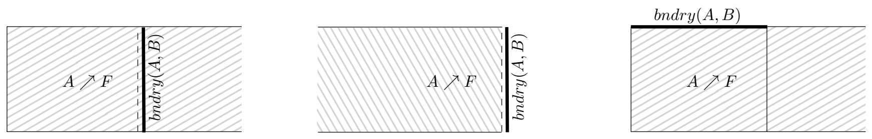
Before formalizing we informally explain, by Figure 3, why the procedure needs to compute the potential entry sets $\mathit{pentry}_\ell(\widehat{W}', \widehat{I}')$, instead of simply performing the intersection between $\widehat{W}'$ and $\widehat{I}'$. Consider the step that build the



(a) Case 1: the flow allows to reach B from $bndry(A, B)$.

(b) Case 2: the flow does not allow to reach B from $bndry(A, B)$.

(c) Case 3: the flow does not allow to reach B from $bndry(A, B)$.



(d) $bndry(A, B) \cap A \nearrow_F \neq \emptyset$: potential entry from A to B.

(e) $bndry(A, B) \cap A \nearrow_F = \emptyset$: entry from A to B impossible.

(f) $bndry(A, B) \cap A \nearrow_F \neq \emptyset$, but the system can not reach B from A.

Fig. 2. The computation of potential entry from A to B involves computing the boundary of A and B (as shown in Figs. 2(a),2(b),2(c)), and identifying states reachable on that boundary (see, respectively, Figs. 2(d),2(e),2(f))

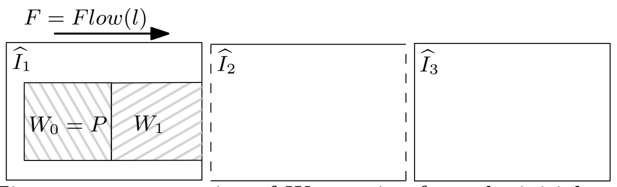
set $\widehat{W}_3$, as depicted in Figure 3(b), and suppose that the procedure, instead of using $Post_\ell(pentry_\ell(\widehat{W}_2, \widehat{I}_3), \widehat{I}_3)$ would use $Post_\ell(\widehat{W}_2 \cap \widehat{I}_3, \widehat{I}_3)$. Due to the fact that $\widehat{I}_2$ is right open, also $\widehat{W}_2$ is right open and then the initial set $\widehat{W}_2 \cap \widehat{I}_3$ would be empty as well as $Post_\ell(\widehat{W}_2 \cap \widehat{I}_3, \widehat{I}_3)$.

Notice that, even if the initial set of valuations $pentry_\ell(\widehat{W}_2, \widehat{I}_3)$ is not in the invariant convex component $\widehat{I}_3$, the set $Post_\ell(pentry_\ell(\widehat{W}', \widehat{I}')$ is non-empty. We have that $pentry_\ell(\widehat{W}_2, \widehat{I}_3) \subseteq I$ that is the starting valuations have to belong to the global non-convex invariant I, not necessary to the convex component $\widehat{I}_3 \in [\![I]\!]$.

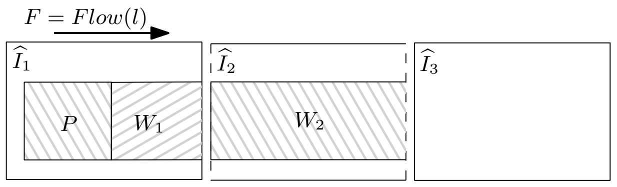
The formal relationship between the fixed-point procedure described above and the computation of the continuous post operator, when the invariant is non-convex, is given by the following theorem:

Theorem 1. *Given a location $\ell \in Loc$ and sets $P \subseteq Inv(\ell)$, $I = Inv(\ell)$, $ncPost_\ell(P, I)$ is the smallest fixed point of the sequence $W_0 = P$,*

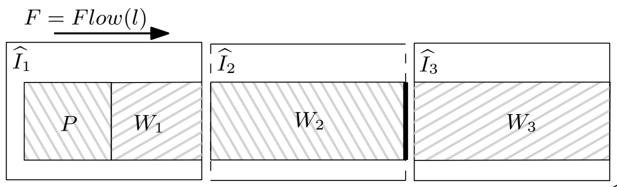
$$W_k = \bigcup_{\widehat{W}' \in [\![W_{k-1}]\!]} \bigcup_{\widehat{I}' \in [\![I]\!]} Post_\ell(pentry_\ell(\widehat{W}', \widehat{I}'), \widehat{I}').$$



(a) First step: computation of W_1 starting from the initial set P.



(b) Second step: computation of W_2 starting from $\mathit{pentry}_\ell(W_1, \widehat{I}_2)$.



(c) Last step: computation of W_3 starting from $\mathit{pentry}_\ell(W_2, \widehat{I}_3)$.

Fig. 3. The central role of $\mathit{pentry}_\ell(W_2, \widehat{I}_3)$ in the computation of $\mathit{ncPost}_\ell(P, I)$.

Moreover, the above sequence reaches the fixed point in at most $n = \big|[\![I]\!]\big|$ *steps, that is* $W_{n+1} = W_n$.

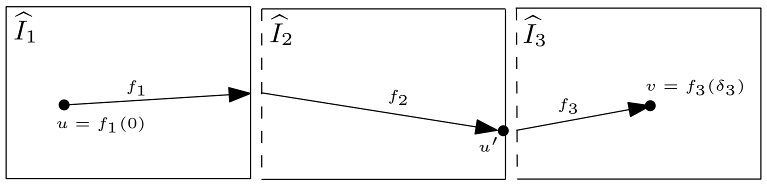
We split the proof of Theorem 1 into several lemmas. Given a polyhedron I and two valuations u and v, assuming that v is reachable from u via an admissible activity that always remains in I (i.e. always avoids $\overline{I}$), we denote by $d(u, v, I)$ the minimum number of convex polyhedra in $[\![I]\!]$ in which the system must remain in order to reach v from u via any admissible activity f. Formally:

$$d(u, v, I) = \min\{n > 0 \mid \exists f \in \mathit{Adm}(\langle \ell, u \rangle), \delta \geq 0, \widehat{I_1}, \ldots \widehat{I_n} \in [\![I]\!] : \\ f(\delta) = v \text{ and } \forall 0 \leq \delta' \leq \delta \; \exists j \in \{1, \ldots, n\} : f(\delta') \in \widehat{I_j}\}.$$

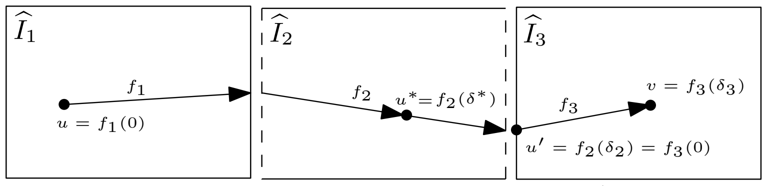
When there is no activity that can reach v from u avoiding $\overline{I}$, we write $d(u, v, I) = \infty$. Hence either $d(u, v, I) \leq \big|[\![I]\!]\big|$ or $d(u, v, I) = \infty$. Now, we define a slightly different version of ncPost_ℓ that take into account only valuations v such that the system, in order to reach v, always remains in a fixed number of convex polyhedra in the invariant. Formally, given a location l and fixed $I = \mathit{Inv}(\ell)$, $P \subseteq I$ and $i \leq |[\![I]\!]|$,

$$\mathit{ncPost}_\ell(P, I, i) = \big\{v \in \mathit{ncPost}_\ell(P, I) | \exists u \in P : d(u, v, I) \leq i\big\}.$$

Note that for all $i \leq j$, $\mathit{ncPost}_\ell(P, I, i) \subseteq \mathit{ncPost}_\ell(P, I, j)$.



(a) From u to u' the system never touches $\widehat{I}_3$.



(b) From u to u' the system touches also $\widehat{I}_3$.

Fig. 4. Cases (a) and (b) of the proof of Lemma 3.

A fundamental property of LHA is that if there is an activity that goes from u to v inside the invariant, there is also a sequence of linear activities that does the same. Moreover, each linear activity is contained within one convex polyhedron of $[\![I]\!]$ and hence the connecting points between any two consecutive linear activities lie on the boundary between two polyhedra in $[\![I]\!]$. The following formalization is a reformulation of Lemma 2.2 in [25] given as Lemma 5 in [7]:

Lemma 2. [7] *Let u and v be valuations, and I a polyhedron. If $d(u,v,I) = i < \infty$, then there is a sequence of linear activities $f_1, \dots, f_i$, delays $\delta_0, \dots, \delta_i$, and convex polyhedra $\widehat{I}_1, \dots, \widehat{I}_i \in [\![I]\!]$ such that* (i) *$f_1 \in Adm(\langle l, u\rangle)$,* (ii) *$f_{i-1}(\delta_{i-1}) = v$,* (iii) *for all $j < i$ it holds $f_j(\delta_j) \in bndry(\widehat{I}_j, \widehat{I}_{j+1})$ and $f_{j+1} \in Adm(\langle l, f_j(\delta_j)\rangle)$, and* (iv) *for all $j \leq i$ and $0 < \delta' < \delta_j$ it holds $f_j(\delta') \in \widehat{I}_j$.*

Now we are ready to give two lemma that prove Theorem 1.

Lemma 3. *For all locations ℓ, polyhedra P and I, and $i \geq 1$, it holds*

$$ncPost_\ell(P, I, i) \subseteq W_i.$$

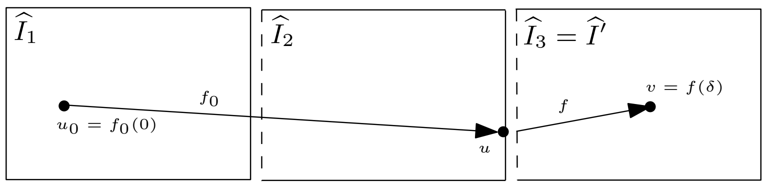
Proof. Let v be a valuation belonging to $ncPost_\ell(P, I, i)$, by definition we have that there exists a valuation $u \in P$, an activity $f \in Adm(\langle \ell, u\rangle)$ and a time $\delta \geq 0$ such that $f(\delta) = v$, $d(u, v, I) = i$ and, for all $0 < \delta' \leq \delta$, $f(\delta') \in I$.

Now, we proceed by induction on $i \geq 1$. For the base case, if $i = 1$ then $d(u, v, I) = 1$. Recalling the definition, $d(u, v, I) = 1$ means that in order to reach valuation v from u the system remains always in just one convex component of $[\![I]\!]$. Hence, there exists a convex polyhedron $\widehat{I'} \in [\![I]\!]$ such that, for all $0 \leq \delta' \leq \delta$, it holds that $f(\delta') \in \widehat{I'}$. As a consequence, $f(0) = u \in P \cap \widehat{I'}$ and, by Corollary 1, we obtain $u \in pentry_\ell(P, \widehat{I'})$. From the valuation u that belongs to the potential entry from P to $\widehat{I}'$, it is possible to reach the valuation v via the activity f by always remaining in $\widehat{I}'$. But this is the definition of post operator, and then we can write $v \in Post_\ell(pentry_\ell(P, \widehat{I}')) \subseteq W_1$, ending the base case. For the inductive step, we consider $i \geq 2$. If the valuation v also belongs to $ncPost_\ell(P, I, i-1)$ then the thesis trivially holds by inductive hypothesis. Otherwise, by Lemma 2, there exists a sequence of activities $f_1, \ldots, f_i$, of convex component $I_1, \ldots, I_i \in [\![I]\!]$ and times $\delta_1, \ldots, \delta_i$, such that from valuation $u = f_1(0)$ it is possible to reach the valuation $u' = f_{i-1}(\delta_{i-1})$, and $u' = f_{i-1}(\delta_{i-1}) \in bndry(\widehat{I}_{i-1}, \widehat{I}_i)$. Moreover, Lemma 2 also says that by following activities $f_1, \ldots, f_{i-1}$, the system always remains in the convex components $I_1, \ldots, I_{i-1}$ and in $bndry(\widehat{I}_{i-1}, \widehat{I}_i)$. By definition of boundary, we have either (a) $u' \in \widehat{I}_{i-1} \cap cl(\widehat{I}_i)$ (and then the system always remains in $I_1, \ldots, I_{i-1}$) or (b) $u' \in \widehat{I}_i \cap cl(\widehat{I}_{i-1})$ (and then the system always remains in $I_1, \ldots, I_i$. Considering case (a), since the system always remains in $I_1, \ldots, I_{i-1}$, it is clear that $d(u, u', I) = i-1$ and then, by inductive hypothesis, $u' \in W_{i-1}$. Hence, $u' \in W_{i-1} \cap cl(\widehat{I}_i)$, that means that there exists a convex polyhedron $\widehat{W}' \in [\![W_{i-1}]\!]$ such that $u' \in \widehat{W}' \cap cl(\widehat{I}_i) \subseteq \widehat{W}' \nearrow_F$ and clearly $u' \in bndry(\widehat{W}', \widehat{I}_i)$. Considering the last two conditions and Lemma 1, we have that $u' \in pentry_\ell(\widehat{W}', \widehat{I}_i)$. Now, from the valuation $u' \in pentry_\ell(\widehat{W}', \widehat{I}_i)$, it is possible to reach the valuation v via the activity f_i without leaving the invariant $\widehat{I}_i$, allowing us to write $v \in Post_\ell(pentry_\ell(\widehat{W}', \widehat{I}_i)) \subseteq W_i$, and this ends the first case (see Figure 4(a) to better understand the reasoning). Case (b), otherwise, is when $u' \in \widehat{I}_i \cap cl(\widehat{I}_{i-1})$ and then in order to reach the valuation u' from u the system always remains in $I_1, \ldots, I_i$. By Lemma 2, for all $0 < \delta^* < \delta_{i-1}$, the valuation $u^* = f_{i-1}(\delta^*)$ belong to $\widehat{I}_{i-1}$ and is reachable from u while the system always remains in $\widehat{I}_1, \ldots, \widehat{I}_{i-1}$. Hence, by definition, $d(u, u^*, I) = i-1$ and then $u^* \in ncPost_\ell(P, I, i-1)$. By inductive hypothesis, we have also that $u^* \in W_{i-1}$, that is there exists $\widehat{W}' \in W_{i-1}$ such that $u^* \in \widehat{W}'$. Moreover, u' can be reached from any of the u^* via the activity f_{i-1}. But this means that $u' \in \widehat{W}' \nearrow_F$ and, recalling that $u' \in \widehat{I}_i$, this means that $u' \in pentry_\ell(\widehat{W}', \widehat{I}_i)$ (Lemma 1). Now, by using again Lemma 2, there exists the activity f_i and the time δ_i such that (i) $f_i(0) = u'$, (ii) $f_i(\delta_i) = v$, and (iii) for all $0 < \delta' < \delta$, $f_i(\delta') \in \widehat{I}_i$. By combining conditions i, ii and iii, we have that $v \in Post_\ell(pentry_\ell(\widehat{W}', \widehat{I}_i), \widehat{I}_i) \subseteq W_i$. This ends the second case (see Figure 4(b) to better understand the reasoning) and the entire proof. □

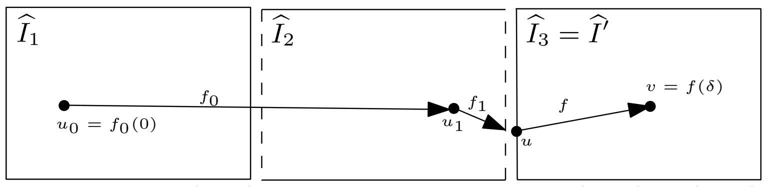
Lemma 4. *For all locations ℓ, polyhedra P and I, and $i \geq 1$, it holds*

$$W_i \subseteq ncPost_\ell(P, I, i).$$

Proof. We proceed by induction on $i \geq 1$. For the base case, let $v \in W_1$ and recalling that $W_0 = P$, by definition there exist convex polyhedra $\widehat{P}' \in [\![P]\!]$ and $\widehat{I}' \in [\![I]\!]$ such that $v \in Post_\ell\big(pentry_\ell(\widehat{P}', \widehat{I}'), \widehat{I}'\big)$. By definition of post operator, there exists a valuation $u \in pentry_\ell(\widehat{P}', \widehat{I}')$, an activity $f \in Adm(\langle l, u\rangle)$ and time $\delta \geq 0$ such that $f(\delta) = v$ and for all $0 < \delta' \leq \delta$, $f(\delta') \in \widehat{I}'$. But the last



(a) Case (a): $u \in \widehat{I}_{i-1} = \widehat{I}_2$. Subcase $(a1)$ occours when $\widehat{I}_1 = \widehat{I}'$ or $\widehat{I}_2 = \widehat{I}'$, otherwise we are in subcase $(a2)$.



(b) Case (b): $u \in \widehat{I}_i = \widehat{I}_3$. Subcase $(b1)$ occours when $\widehat{I}_1 = \widehat{I}'$ or $\widehat{I}_2 = \widehat{I}'$, otherwise we are in subcase $(b2)$.

Fig. 5. The cases in the proof of Lemma 4.

one implies $d(u, v, I) = 1$ that, together with the other conditions, allows us to write $v \in \mathit{ncPost}_\ell(P, I, 1)$ and this concludes the base case. For the inductive step, let $i \geq 2$ and $v \in W_i$. If the valuation v also belongs to W_{i-1}, then the thesis trivially holds for the inductive hypothesis. Otherwise, there exists convex polyhedra $\widehat{W}' \in W_{i-1}$ and $\widehat{I}' \in [\![I]\!]$ such that $v \in \mathit{Post}_\ell(\mathit{pentry}_\ell(\widehat{W}', \widehat{I}'), \widehat{I}')$ By definition of post operator, there exists a valuation $u \in \mathit{pentry}_\ell(\widehat{W}', \widehat{I}')$, an activity $f \in \mathit{Adm}(\langle l, u\rangle)$ and time $\delta \geq 0$ such that $f(\delta) = v$ and for all $0 < \delta' \leq \delta$, $f(\delta') \in \widehat{I}'$. From $u \in \mathit{pentry}_\ell(\widehat{W}', \widehat{I}')$ we have either (a) $u \in \widehat{W}' \cap cl(\widehat{I}')$ or (b) $u \in cl(\widehat{W}') \cap \widehat{I}'$. Case (a), that is $u \in \widehat{W}'$, trivially implies that $u \in W_{i-1}$ and then, by inductive hypothesis, $u \in \mathit{ncPost}_\ell(P, I, i-1)$. By definition of post operator, there exists a valuation $u_0 \in P$, an activity $f_0 \in \mathit{Adm}(\langle l, u_0\rangle)$ and time $\delta_0 \geq 0$ such that $f(\delta_0) = u$, for all $0 < \delta' \leq \delta_0$, it holds $f(\delta') \in I$ and $d(u_0, u, I) = i - 1$. Now we can distinguish two subcases: $(a1)$ when the convex polyhedron $\widehat{I}'$ is one of the $i-1$ components of the invariant in which the system remains, and $(a2)$ when the system never reaches $\widehat{I}'$ during the activity f_0 that leads from u_0 to u. Figure 5(a) graphically depicts this case and the corresponding subcases. In the case $(a1)$ v belongs to $\mathit{ncPost}_\ell(P, I, i-1)$: one can reach v from u by concatenation f_0 and f and the system does not exit the $i-1$ (including $\widehat{I}'$) components of the invariant. So $v \in \mathit{ncPost}_\ell(P, I, i-1)$, which directly implies that $v \in \mathit{ncPost}_\ell(P, I, i)$, and we are done for the subcase $(a1)$. In the case $(a2)$, it is clear that $d(u_0, v, I) = i$ (the system reaches for the first time $\widehat{I}'$) and, recalling that $f(\delta) = v$, we can conclude that $v \in \mathit{ncPost}_\ell(P, I, i)$, and this conclude the subcase $(a2)$. In the case (b), we have that $u \in cl(\widehat{W}') \cap \widehat{I}'$. By definition of potential entry there exists a valuation $u_1 \in \widehat{W}'$, an activity $f_1 \in \mathit{Adm}(\langle l, u_1\rangle)$ and time $\delta_1 \geq 0$ such that $f_1(\delta_1) = u$ and for all $0 \leq \delta' < \delta_1$,

$f_1(\delta') \in \widehat{W}'$. But $u_1 \in \widehat{W}'$ implies that $u_1 \in W_{i-1}$ and by inductive hypothesis we can write $u_1 \in ncPost_\ell(P, I, i-1)$. Now, by definition of the post operator, there exists a valuation $u_0 \in P$, an activity $f_0 \in Adm(\langle l, u^* \rangle)$ and time $\delta_0 \geq 0$ such that $f_0(\delta_0) = u$, for all $0 < \delta' \leq \delta_0$ it holds that $f_0(\delta') \in I$ and $d(u_0, u_1, I) = i-1$. The last one means that in order to reach the valuation u_1 from the valuation u_0 (via the activity f_0) the system always remains in $i-1$ convex components of $[\![I]\!]$. Now, two different subcases can happen: $(b1)$ $\widehat{I}'$ is one of the $i-1$ visited convex component or $(b2)$ otherwise. Figure 5(b) graphically depicts this case and the corresponding subcases. Consider the subcase $(b1)$ and recall that for all $0 \leq \delta' < \delta_1$, $f_1(\delta') \in \widehat{W}'$, clearly we have that $f_1(\delta') \in W_{k-1}$ and, by inductive hypothesis, $f_1(\delta') \in ncPost_\ell(P, I, i-1)$. Hence, in order to reach the valuation $u = f_1(\delta_1)$ from u_0 the system always remains in the same $i-1$ convex (included $\widehat{I}'$) and then $d(u_0, u, I) = i-1$. Then, from u to v (via the activity f) the system always remains in $\widehat{I}'$ and then $d(u_0, v, I) = i-1$. Hence, $v \in ncPost_\ell(P, I, i-1) \subseteq ncPost_\ell(P, I, i)$, and this ends subcase $(b1)$. In the subcase $(b2)$, we have that $d(u_0, u, I) = i-1$. Then in order to reach v from u (via activity f), the system remains in $\widehat{I}'$ (never reached before) and then $d(u_0, v, I) = i$ and hence $v \in ncPost_\ell(P, I, i)$ that ends the case and the entire proof. □

Finally, we are able to easly prove Theorem 1, by using Lemma 3 and Lemma 4 as follows:

Proof of Theorem 1 First, notice that give two valuation u and v, where $u, v \in ncPost_\ell(P, I)$, then by definition of post operator $d(u, v, I) \leq \left|[\![I]\!]\right|$. Then trivially holds that let $n = \left|[\![I]\!]\right|$, we have that $ncPost_\ell(P, I) = ncPost_\ell(P, I, n)$.

Now, by Lemmas 3 and 4, for all $k \geq n$, $ncPost_\ell(P, I) = W_k$. This means that the sequence reaches the fixed point after at most n iterations. □

2.4 Related Work

In [17], the author shows a different approach in order to tackle non-convex invariants. The proposed algorithm to compute the reachable set is built only for closed convex invariants, but this is not a restriction because (closed) non-convex invariants can be modeled by splitting locations. This means that starting from an automaton A with non-convex invariants, it is necessary to build an equivalent automaton B whose locations have only convex invariants: this is done by taking, for each location of A, the exact convex covering Q of the corresponding invariant and then, for each convex component $\widehat{Q} \in Q$, by adding a location to B whose associated (convex and closed) invariant is $\widehat{Q}$. Therefore, this approach does not work with non-closed invariants and needs a postprocessing phase in order to build the automaton B. Our approach tries to overcome these limitations: the recahibility analysis is directly done by using the $ncPost_\ell$ operator, allowing the usage of non-closed invariants and avoiding the hidden process of building a new automaton.

The operator $ncPost_\ell$ has an interesting relation with a group of operators used in safety control problems. In particular with the RWA_l^{m} operator defined in [7] (also known under other names as *Reach* [24], *Unavoid_Pre* [3] and *flow_avoid* [25]). Informally, given a location l and two sets of variable valuations U and V, $RWA_l^{\mathrm{m}}(A, B)$ contains the set of valuations from which there exists an activity that reaches A while avoiding $B \cap \overline{A}$. Formally:

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

Informally, we can express $ncPost_\ell(P, I)$ as the set of valuations that, starting from P, may reach I while avoiding $\overline{I}$. The main difference is given by the fact that RWA_l^{m} does not take into account a starting set of valuations, while the computation of $ncPost_\ell(P, I)$ depends on the initial set P. This is reflected also in how the operators are computed: in the case of $RWA_l^{\mathrm{m}}(A, B)$, the computation is given by a backward procedure that starts from the target set of valuations A and by taking all the valuations that avoid B. In the opposite, $ncPost_\ell(P, I)$ is computed by a forward procedure that starts from the initial set of valuations P and by taking all the valuations that avoid $\overline{I}$.

3 Linear Hybrid Automata with Urgency

In this section, we extend LHA with a so-called *urgency condition* each location. The urgency condition impedes time elapse, i.e., no continuous activities continue from a valuation that satisfies the condition. As we will see later, there is a connection between urgency conditions on locations and urgent semantics on transitions.

3.1 Definition and Semantics

We denote by $SPoly(X)$ the subset of $\mathbb{R}^X$ that can be obtained by finite disjunction of closed convex polyhedron. A *Linear Hybrid Automaton with Urgency (LHAU)* $H = (Loc, X, Lab, Edg, Flow, Inv, Urg, Init)$ consists of a LHA defined in Sect. 2 and a mapping $Urg : Loc \to SPoly(X)$, called *urgency condition*. To designate the urgent states, we use the shorthand $UrgS = \bigcup_{l \in Loc} \{\ell\} \times Urg(\ell)$.

Urgent transitions In our definition, the urgency condition is defined for each location. An alternative approach, popular mainly because of its syntactical simplicity, is to designate each discrete transition as urgent or not. This is also referred to as *as-soon-as-possible (ASAP) transitions*. Urgent transitions can easily be translated to an urgency condition: Let $Edg_U \subseteq Edg$ be the set of urgent transitions. Then the equivalent urgency condition is the union of the outgoing guards,

$$Urg(\ell) = \{u \mid \exists(\ell, \eta, \ell') \in Edg_U : (u, v) \in \eta\}.$$

Semantics The urgency conditions affect only the timed steps, while the definition of discrete step remains the same as for LHA. Given a state $s = \langle l, v\rangle$, we define $loc(s) = l$ and $val(s) = v$. In order to give the semantics of timed-steps for $LHAU$ we define, for an activity $f \in Adm(s)$, the *Switching Time* of f in l, denoted by $SwitchT(f, U)$, as the value $\delta \geq 0$ such that, for all $0 \leq \delta' < \delta$, $f(\delta') \notin U$ and $f(\delta) \in U$. When for all $\delta \geq 0$ it holds that $f(\delta) \notin U$, we write $SwitchT(f, U) = \infty$. Informally, the switching time of an activity f in the location l specifies the maximum amount of time δ such that the system, by following the activity f, is allowed to remain in the location l.

Given two states s, 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)* there exists the timed step $s \xrightarrow{\delta,f} s'$ in the LHA without urgency conditions, and *(ii)* $\delta \leq SwitchT(f, Urg(loc(s)))$. The special timed step $s \xrightarrow{\infty,f}$, which represents the case when the system follows the activity f forever, is allowed only if, for all $0 \leq \delta' \leq \delta$, $\langle loc(s), f(\delta')\rangle \in InvS$ and $SwitchT(f, Urg(loc(s))) = \infty$.

3.2 Parallel Composition

One attractive feature of urgency is that a model can be decomposed for cases where this is not possible without urgency. Consider the example of an automaton for the plant and an automaton for the controller. Without urgency, the controller automaton can in general not prevent time elapse in the plant automaton, unless an additional clock is introduced and that clock is sampled periodically, see also remarks in Sect. 1. We give a brief formal definition of parallel composition with urgency for the case where both automata range over the same variables. For a definition including input and output variables (as used in SpaceEx) see [11]. The key here is that the urgency condition of the composition is the union of the urgency conditions of the operands.

Definition 3 (Parallel composition). *Given linear hybrid automata with urgency H_1, H_2 with $H_i = (Loc_i, X, Lab_i, Edg_i, Flow_i, Inv_i, Urg_i, Init_i)$, their* parallel composition *is the LHAU $H = (Loc_1 \times Loc_2, X, Lab_1 \cup Lab_2, Edg, Flow, Inv, Urg, Init)$, written as $H = H_1 \| H_2$, where*

- $((l_1, l_2), \alpha, \eta, (l_2', l_2')) \in Edg$ *iff*
  - $\alpha \in Lab_1 \cap Lab_2$, *for* $i = 1, 2$, $(l_i, \alpha, \eta_i, l_i') \in Edg_i$, *with* $\eta = \eta_1 \cap \eta_2$, *or*
  - $\alpha \notin Lab_1$, $l_2' = l_2$, *and* $(l_1, \alpha, \eta, l_1') \in Edg_1$, *or*
  - $\alpha \notin Lab_2$, $l_1' = l_1$, *and* $(l_2, \alpha, \eta, l_2') \in Edg_2$ *;*
- $Flow(l_1, l_2) = Flow_1(l_1) \cap Flow_2(l_2)$;
- $Inv(l_1, l_2) = Inv_1(l_1) \cap Inv_2(l_2)$;
- $Urg(l_1, l_2) = Urg_1(l_1) \cup Urg_2(l_2)$;
- $Init(l_1, l_2) = Init_1(l_1) \cap Init_2(l_2)$.

3.3 Reachability

The discrete post operator for the class of $LHAU$ is trivially the same of the classical one, while the continuous one, that we call *Urgent Continuous Post Operator*, changes due to the extra condition induced by the operator $SwitchT$:

Definition 4 (Urgent continous post). *Given a linear hybrid automaton with urgency H, a location $\ell \in Loc$, and a set of valuations $P \subseteq Inv(\ell)$, let $I = Inv(\ell)$, and $U = Urg(\ell)$. The* urgent continuous post operator $UPost(P, I, U)$ *is defined as:*

$$UPost(P, I, U) = \Big\{ v \in val(X) \Big| \exists u \in P, f \in Adm(\langle \ell, u \rangle), \delta \geq 0 : \\ f(\delta) = v, \text{ for all } 0 < \delta' \leq \delta, f(\delta') \in I, \text{ and } \delta \leq SwitchT(f, U) \Big\}.$$

Notice that, by definition, $UPost_\ell(P, I, U) \subseteq ncPost_\ell(P, I)$.

3.4 Computing the Urgent Continuous Post Operator

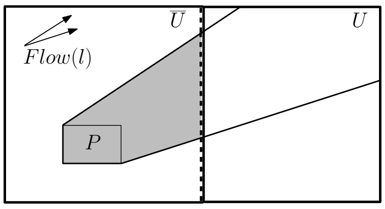
We now derive a construction of the urgent post operator, starting with the post operator for non-convex invariants and adding the states that are missing.

The urgent post operator has to compute the valuations that are reachable from some set P without passing through states in the urgent set U. This includes the states that are reachable within the complement of U, so $ncPost_\ell(P \cap \overline{U}, I \cap \overline{U})$ is an underapproximation of $UPost_\ell(P, I, U)$. In the following, let $\mathbb{V}_{nc} = ncPost_\ell(P \cap \overline{U}, \overline{U})$ and $\mathbb{V}_U = UPost_\ell(P, I, U)$. The set $\mathbb{V}_{nc}$ trivially does not contain valuations that belong to U (by definition of invariant), while $\mathbb{V}_U$ also contains those valuations that touch U for the first time on a run. Indeed, while the system is allowed to remain on the boundary of an invariant for any time as the invariant is satisfied, the system can not remain on the boundary of an urgency condition because in the instant the urgency condition is meet, the system can not evolve any more, i.e., it is forced either to stop the evolution of the continuous variables or to jump in another location. Figure 6 illustrates the relationship between $ncPost_\ell(P \cap \overline{U}, I \cap \overline{U})$ (left column of the figure) and $UPost_\ell(P, I, U)$ (right column of the figure) with several examples.

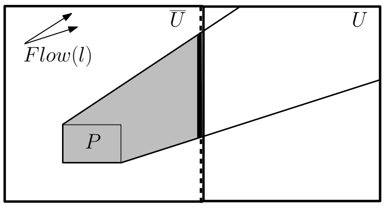
We compute $\mathbb{V}_U$ by adding (i) P itself (ii) the reachable states in $\bar{U}$ $\mathbb{V}_{nc}$, and (iii) the valuations that belong to the boundary between $\mathbb{V}_{nc}$ and U from where it is possible to reach U by following some admissible activity. This is formalized as follows:

Theorem 2. *Given a location $\ell \in Loc$ and a set $P \subseteq Inv(\ell)$, let $I = Inv(\ell)$, $U = Urg(\ell)$, $\mathbb{V}_{nc} = ncPost_\ell(P \cap \overline{U}, I \cap \overline{U})$, and $B = \bigcup_{\widehat{A}' \in [\![\mathbb{V}_{nc}]\!]} \bigcup_{\widehat{U}' \in [\![U]\!]} pentry_\ell(\widehat{A}', \widehat{U}' \cap I)$. Then*

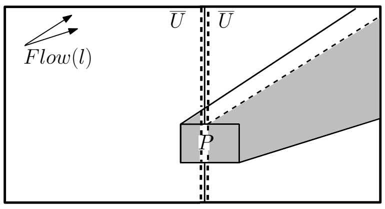
$$UPost_\ell(P, I, U) = P \cup \mathbb{V}_{nc} \cup B.$$



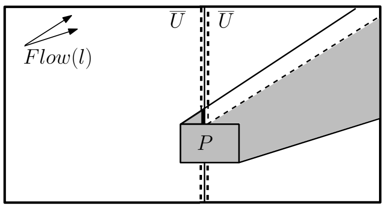
(a) $\mathbb{V}_{nc}$ with $U = \{x \geq d\}$.



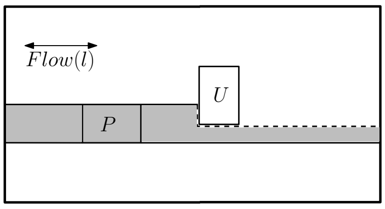
(b) $\mathbb{V}_U$ contains the boundary of $\mathbb{V}_{nc}$ with U.



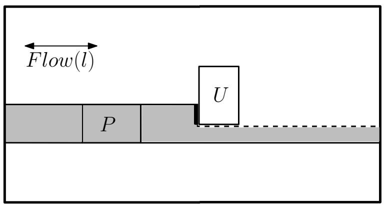
(c) $\mathbb{V}_{nc}$ with $U = \{x = d\}$.



(d) $\mathbb{V}_U$ contains $P \cap U$ and the boundary of $\mathbb{V}_{nc}$ with U.



(e) $\mathbb{V}_{nc}$ with flow tangential to U.



(f) $\mathbb{V}_U$ contains only the reachable part of the boundary of $\mathbb{V}_{nc}$ with U.

Fig. 6. The urgent post states $\mathbb{V}_U = UPost_\ell(P, I, U)$ can be obtained from $\mathbb{V}_{nc} = ncPost_\ell(P \cap \overline{U}, I \cap \overline{U})$ plus a part of the boundary between $ncPost_\ell(P \cap \overline{U}, I \cap \overline{U})$ and U. Here, the invariant I is defined to be *true*. The dashed lines identify the non-closed borders.

Proof. [$\subseteq$] Let $v \in UPost_\ell(P, I, U)$, by definition there exists a valuation $u \in P$, an admissible activity $f \in Adm(\langle \ell, u \rangle)$ and a real value $\delta \geq 0$ such that $f(\delta) = v$, for all $0 < \delta' \leq \delta$ it holds that $f(\delta') \in I$, and $\delta \leq SwitchT(f, U)$. Moreover, due to the fact that $P \subseteq I$, we have also that $f(0) \in I$ and then, for all $0 \leq \delta' \leq \delta$, we have that $f(\delta') \in I$. Now, we can distinguish three different cases on $SwitchT(f, U)$:

1. If $SwitchT(f, U) = 0$, then $\delta = 0$. By definition of $SwitchT$, we have that $u = v = f(0) \in U$, and then $v \in (P \cap U)$, and the thesis holds.
2. If $0 \neq \delta < SwitchT(f, U)$, then for all $0 \leq \delta' \leq \delta$ it holds $f(\delta') \in \overline{U}$. The latter, coupled with the conditions on the invariant, gives us $f(\delta') \in I \cap \overline{U}$, and then $v \in A$.

3. If $0 \neq \delta = \mathit{SwitchT}(f, U)$ then, for all $0 \leq \delta' < \delta$ it holds that $f(\delta') \in \overline{U}$, and $f(\delta) \in U$. Moreover, for all $0 \leq \delta' < \delta$, $f(\delta')$ also belongs to $\mathbb{V}_{nc}$ ($f(\delta')$ is always in the invariant $I \cap \overline{U}$). This means that, starting from a valuation in $\widehat{A}' \in [\![\mathbb{V}_{nc}]\!]$ it is possible to reach $f(\delta)$ in $\widehat{U}' \in [\![U]\!]$, and by definition of potential entry we have that $v = f(\delta) \in \mathit{pentry}_\ell(\widehat{A}', \widehat{U}' \cap I) = B$.

[$\supseteq$] Considering the valuation v that belongs to the r.h.s. of the formula in Theorem 2, we can distinguish three cases, based on at what disjunct among A, B or $(P \cap U)$ the valuation v belongs.

1. If $v \in (P \cap U)$, then there exists an activity $f \in \mathit{Adm}(\langle \ell, v \rangle)$ and $\delta = 0$ such that $v = f(\delta) \in P \cap I \cap U$. Hence $f(0)$ is in the invariant I and $\mathit{SwitchT}(f, U) = 0$ and we can conclude that $v \in \mathit{UPost}_\ell(P, I, U)$.
2. If $v \in A$, then by definition of $\mathit{ncPost}_\ell(P \cap \overline{U}, I \cap \overline{U})$, there exists a valuation $u \in P \cap \overline{U}$, an activity $f \in \mathit{Adm}(\langle \ell, u \rangle)$ and time $\delta \geq 0$ such that, for all $0 \leq \delta' \leq \delta$, we have $f(\delta') \in I \cap \overline{U}$. This clearly implies that $\mathit{SwitchT}(f, U) > \delta$, and we can conclude that $v \in \mathit{UPost}_\ell(P, I, U)$.
3. If $v \in B$ then, by definition of B, there exists $\widehat{A}' \in [\![\mathbb{V}_{nc}]\!]$ and $\widehat{U}' \in [\![U \cap I]\!]$ such that $v \in \mathit{pentry}_\ell(\widehat{A}', \widehat{U}' \cap I)$. Using the potential entry definition, we have that there exists a valuation $u \in \widehat{A}'$, an activity $f \in \mathit{Adm}(\langle \ell, u \rangle)$ and a time $\delta \geq 0$ such that $v = f(\delta) \in \mathit{bndry}(\widehat{A}', \widehat{U}' \cap I)$ and for all $0 \leq \delta' < \delta$, it holds that $f(\delta') \in \widehat{A}'$. The last one implies, by definition of $\mathbb{V}_{nc}$ (i.e. post operator), that (i) for all $0 \leq \delta' < \delta$, $f(\delta') \in I \cap \overline{U}$. Moreover, the fact that $U \in \mathit{SPoly}(X)$ and $f(\delta) \in \mathit{bndry}(\widehat{A}', \widehat{U}' \cap I)$ implies that $f(\delta) \in \mathit{cl}(\widehat{A}') \cap \widehat{U}' \cap I$ and then (ii) $f(\delta) \in \widehat{U}' \cap I$. By coupling conditions (i) and (ii) we have that (iii) $\delta = \mathit{SwitchT}(f, U)$. But conditions (i), (ii) and (iii) are the definition of the urgent post operator, allowing us to write $v \in \mathit{UPost}_\ell(P, I, U)$.

□

3.5 Related Work

A general class of hybrid automata with urgency conditions is described in detail in [22], but without giving a way to compute the continuous post operator that takes into account the extra constraints coming from the urgency. In that work, the authors define the *Time Can Progress (tcp)* predicate that roughly speaking specifies, for each location $l \in \mathit{Loc}$, the maximum sojourn time at l, which may depend on the values of the variables when entering the location. There are two main differences between our urgency condition and the *tcp* predicate. First of all, an urgency condition on location l defines when the system is constraint to exit from l, while *tcp* describes exactly the opposite. Then, informally speaking, we can say that in order to model, by using our formalism, a *tcp* predicate associated to a location l, we need to attach to l an urgency condition that is the complement of *tcp*. In [22] the authors also describe several kind of policies on urgency. One of them, called *Synchronous Scheduling Policy*, exactly describes our approach: the location must be left as soon as possible an edge becomes enabled (the urgency condition is meet, in our case). This policy is realized by setting the *tcp* predicate equal to the complement of the disjunction of the outgoing guards from the related location. This means that, by using our urgency condition, in order to model multi-outgoing asap transition from a location ℓ, we have to define $Urg(l)$ as the disjunction of the outgoing guards transitions. This give us another motivation to choose non-convex polyhedra to define the urgency condition: by this way, we are able to easily model a system with multiple outgoing transition with asap guards. Notice that the semantics specified in [22] requires that in the exact moment that a location ℓ is entered, the *tcp* associated to ℓ must be satisfied. In our framework we relax this constraint, by allowing to jump in ℓ even if its urgency condition is already satisfied: in this case, the system must exit from ℓ instantaneously. The rational behind this choice coming from the usual asap transitions semantics: considering a location ℓ_1 with an outgoing transition that must be taken in the exact moment that a variable assumes a fixed value. Even if the target location ℓ_2 has another asap outgoing transition whose guard is already satisfied (and hence the system is constrained to exit from ℓ_2 without time progress), the system is still allowed to jump to ℓ_2 and escape from it instantaneously. For this motivation, our $UPost^c_{\ell_2}(P, I, U)$ operator also contains all the valuations that at entering time (valuations that belong to P) already meet the urgency condition.

Our definition of urgency condition is closer to the *stopping condition* defined in [14] (in the context of timed automata), that is a predicate on the clock-variables of an automaton, associated to each location, which allows passing of time in the location as long as the stopping condition is false. The urgency in hybrid automata is also described in the *Computational Interchange Format for Hybrid Systems (CIF)* (see [5]), accepted as standard for modelling hybrid systems. As in our case, the time is allowed to progress until the urgency condition is false. The difference is that, similarly to the *tcp* predicate, *CIF* requires that in the exact moment that a location is entered, the valuation of the variables have to satisfies the condition.

Urgent locations In the classic LHA model checker HyTech, a transition can be designated as urgent by adding the keyword ASAP [17]. But this is restricted to transitions without guard constraints locally, as well as in the composed model (see [15, 17] for more details). This is equivalent to having *urgent locations*, i.e., locations in which time progress is not allowed. The real-time verification tool UPPAAL [6] similarly features urgent locations and urgent channels (synchronization labels) that can be used only on transitions without guard constraints. Urgent locations are semantically equivalent to adding an extra variable t, with dynamics $\dot{t} = 1$, that is set to zero when the location is entered and by attaching the invariant $t = 0$ to the location. It is easy to check that this technique can not be applied if we want to model even just a slightly more complex urgency condition. For example, consider a system with a variable x and a location, say ℓ_1, where the derivative of x can be zero, and the location must be left when $x = 0$. There is no way to model this behaviour just using invariants: by defining $(x < 0) \vee (x > 0)$ as invariant, the system can never reach the state $x = 0$. In the other hand, by relaxing the invariant in order to allow the system to reach a state with $x = 0$, the system is allowed to remain in location ℓ_1 always.

In previous versions of our model checker PHAVer, transitions could be designated as urgent, but only if the guard consists of a single constraint, locally as well as in the composed model [13]. This restriction was imposed because it suffices to be able to compute the urgent post using the standard post operator for convex invariants.

Almost ASAP In [26] the authors propose a relaxed semantics on asap transition in the context of the timed automaton, for the so called *almost asap* by delay δ. In practice, they define the *guard enlargement*, that means that transitions can be taken also with δ time delay. The rationale behind this approach is that no hardware can guarantee that a transition will always be taken in the exact moment as defined in theory. We could define a similar approach, not only on clock variables, in a simple but opposite way: it is enough to define the urgency condition by narrowing all the constraints by a quantity that is equal to the maximum variation of the variable in the time δ.

4 Examples

The following examples shall illustrate the application of the non-convex and urgent post operators. The first example is a set of test cases for the implementation. The second example is a small case study modeling a batch reactor system.

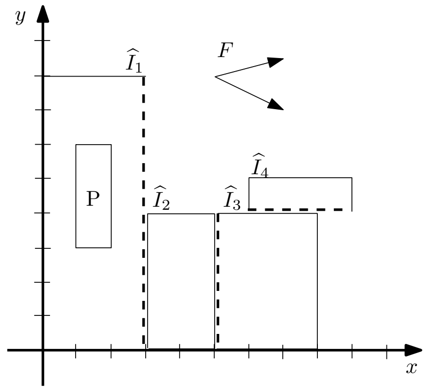
4.1 Test Cases

Figure 7 depicts two test cases. The first one (Figure 7(a)) is used to check cases (a) and (b) of Lemma 3: the two cases are verified because one time there is a potential entry all inside $\widehat{I}_2$ (by going from $\widehat{I}_1$ to $\widehat{I}_2$), and one time there is a potential entry that is not inside $\widehat{I}_3$ (by going from $\widehat{I}_2$ to $\widehat{I}_3$). Notice that in the situation shown in Figure 7(a), all the founded potential entry really are entries. This test case is described as follows:

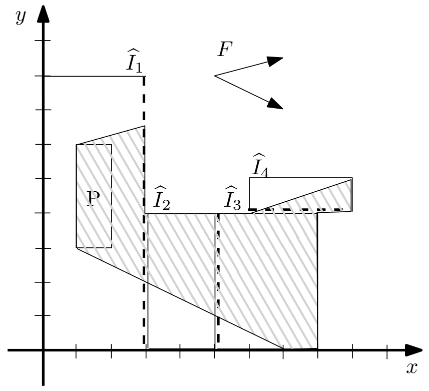
- Init set $P \equiv x \geq 1$ *and* $x \leq 2$ *and* $y \geq 3$ *and* $y \leq 6$;
- Invariant $Inv = \widehat{I}_1 \cup \widehat{I}_2 \cup \widehat{I}_3 \cup \widehat{I}_4$;
- $\widehat{I}_1 \equiv 0 \leq x < 3$ *and* $0 \leq y \leq 8$;
- $\widehat{I}_2 \equiv 3 \leq x \leq 5$ *and* $1 \leq y \leq 4$;
- $\widehat{I}_3 \equiv 5 < x \leq 8$ *and* $0 \leq y \leq 4$;
- $\widehat{I}_4 \equiv 6 \leq x \leq 9$ *and* $4 < y \leq 7$;
- Flow $F \equiv \dot{x} = 1$ *and* $-0.5 \leq \dot{y} \leq 0.5$.

The set of all the reachable valuations is described as follows:

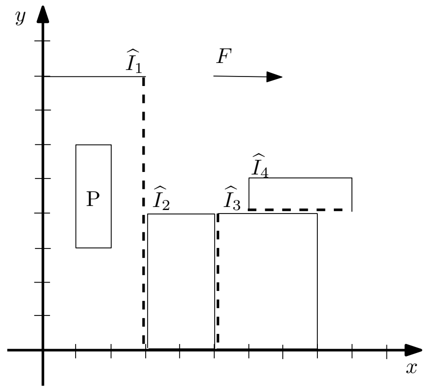
$$\big(1 \leq x < 3 \text{ and } -0.5 \cdot x + 3.5 \leq y \leq 0.5 \cdot x + 5.5\big)||$$



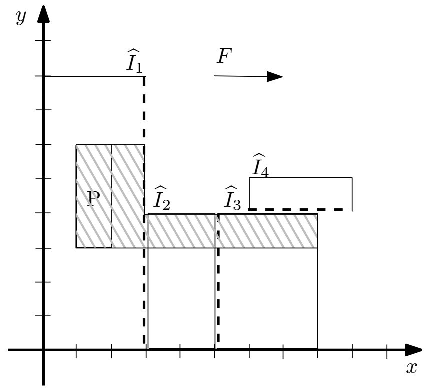
(a) Test Case 1: used to stress Lemma 3 cases (a) and (b).



(b) Test Case 1 Result.



(c) Test Case 2: used to stress the potential entry definition.



(d) Test Case 2 Result.

Fig. 7. Two Test Cases with the results.

$$\big(3 \leq x \leq 5 \text{ and } -0.5 \cdot x + 3.5 \leq y \leq 4\big)||$$

$$\big(5 < x \leq 8 \text{ and } y \geq -0.5 \cdot x + 3.5 \text{ and } 0 \leq y \leq 4\big)||$$

$$\big(6 < x \leq 9 \text{ and } 4 < y \leq 0.5 \cdot x + 1\big).$$

The second test case (Figure 7(c)) is built with the same invariant shape of the first one (and then it is usuful to stress the two cases of Lemma 3) but the real aim here is to see what happens when there is a non-empty potential entry that actually it is not a real entry: here, due to the nature of the flow (it is constant and described by $\dot{x} = 1$, that is no vertical movement is allowed) when the system reach the convex component $\widehat{I}_3$, even if it is possible to identify a potential entry to $\widehat{I}_4$, the flow is such that it is impossible to enter in $\widehat{I}_4$ from $\widehat{I}_3$.

This test case is described as follows:

- Init set $P \equiv x \geq 1$ *and* $x \leq 2$ *and* $y \geq 3$ *and* $y \leq 6$;
- Invariant $Inv = \widehat{I_1} \cup \widehat{I_2} \cup \widehat{I_3} \cup \widehat{I_4}$;
- $\widehat{I_1} \equiv 0 \leq x < 3$ *and* $0 \leq y \leq 8$;
- $\widehat{I_2} \equiv 3 \leq x \leq 5$ *and* $1 \leq y \leq 4$;
- $\widehat{I_3} \equiv 5 < x \leq 8$ *and* $0 \leq y \leq 4$;
- $\widehat{I_4} \equiv 6 \leq x \leq 9$ *and* $4 < y \leq 7$;
- Flow $F \equiv \dot{x} = 1$ *and* $\dot{y} = 0$.

The set of all the reachable valuations is described as follows:

$$\big(1 \leq x < 3 \text{ and } 3 \leq y \leq 6\big)||$$
$$\big(3 \leq x \leq 5 \text{ and } 3 \leq y \leq 4\big)||$$
$$\big(5 < x \leq 8 \text{ and } 3 \leq y \leq 4\big)||.$$

4.2 Batch Reactor

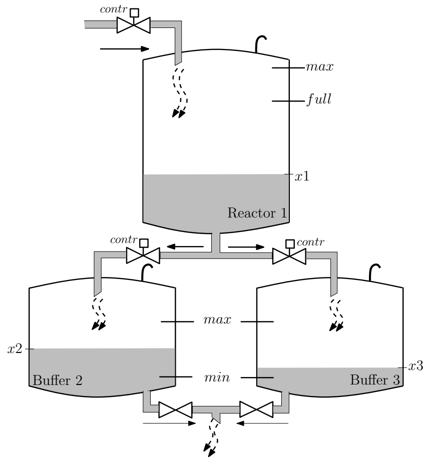
To showcase the algorithm an its implementation, we present a modular model of a batch-reactor system, which is a variation of the case study in [4]. It shall illustrate the following points:

- urgency conditions simplify the modeling process greatly because models can be better decomposed and each module can be simpler;
- urgent and non-urgent transitions arise naturally in practice, as abstractions that reflect the degree of determinism and knowledge about the system;
- urgent transitions with more than one guard constraint also arise naturally, illustrating how limiting the restrictions of HyTech and PHAVer can be;
- the requirement that urgency conditions be closed is of lesser importance in practice, since individual guard constraints need not be closed.

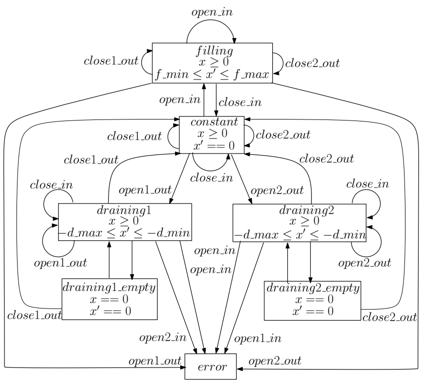
The batch reactor is comprised of a reactor R1 and two buffer tanks B2,B3 connected by pipes. The reactor is used to create a product that is then made available to a consumer in the two buffer tanks, see the schematic in Fig. 8(a). A controller measures the fill levels in the reactor and the buffers, and opens and closes valves connecting the reactor to the buffers in order to produce and deliver the product to the consumer. The specification is to verify that neither buffer ever becomes empty, and that none of the tanks overflows.

We now present the LHA models.

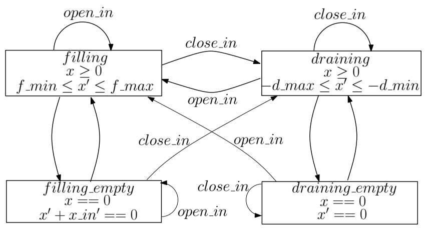
Controller The controller automaton is shown in Fig. 8(d). The opening and closing of valves is modeled by synchronization labels. In the production step, the reactor is filled with educts (raw materials) coming from the outside. Details on the filling and reaction process itself are omitted since they are irrelevant to this example, but it does take a certain amount of time and produces an uncertain amount of product. This is modeled by the fact that the controller ends the filling process when the reactor level $x_1 \in [x_{1,full}, x_{1,max}]$, which is accomplished with the invariant $x_1 \leq x_{1,max}$ and a non-urgent transition with



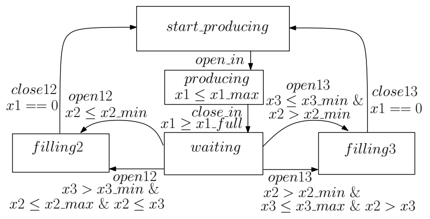
(a) Schematic of the Batch-Reactor System.



(b) Automaton Model of the Reactor.



(c) Automaton Model of the Buffers.



(d) Automaton Model of the Controller.

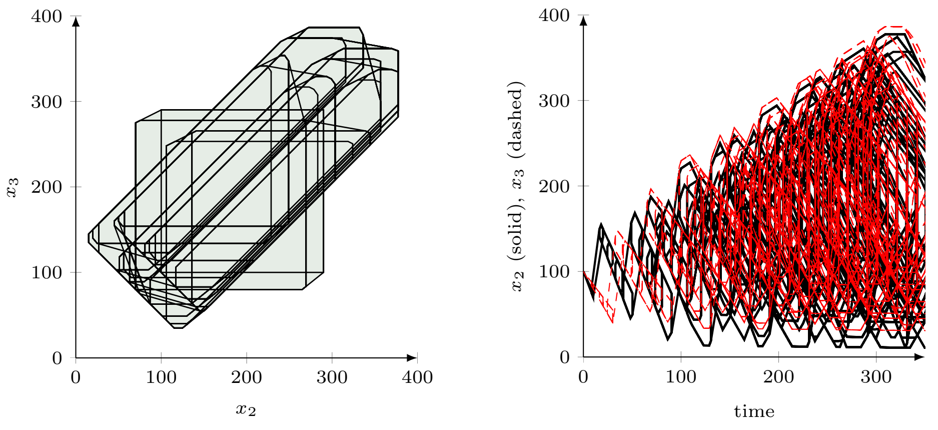
Fig. 8. Batch Reactor System: Schematic and Automata Models.

label *close_in* and guard condition $x_1 \geq x_{1,full}$. When the product is ready, the controller decides whether to fill buffer B2, buffer B3, or wait. The controller decides which buffer to fill using the following simple criteria:

– To avoid overflow, never start filling a buffer above a given maximum level.
– To avoid empty buffers, fill a buffer below a given minimum level.
– If the above is met, fill the buffer with the lower level.
– To be deterministic, prioritize B2.

All transitions for filling buffers B2 and B3 are urgent. The if-then-else structure of the criteria leads to guards with more than one constraint, some of which are strict inequalities. Thanks to the urgency, the controller model requires no clocks or while-loops.

Reactor The reactor automaton is shown in Fig. 8(b). The locations of the reactor correspond to the different combinations of open and closed valves. If the tank is empty, the dynamics of the locations *draining*$_i$ lead to a deadlock, so locations *draining*$_i$*-empty* with $\dot{x}_1 = 0$ are included to allow time to elapse. The



(a) reachable buffer levels x_2 and x_3 for safe parameter values

(b) decreasing the min. reactor outflow by 1% eventually leads to an empty buffer B3

Fig. 9. The evolution of the continuous variables of the batch reactor example, starting from location ℓ_1 with initial values $x_1 = 0$, $x_2 = 100$ and $x_2 = 100$

model is simplified using the assumptions that the reactor must not be filled and drained at the same time (a common requirement in chemical engineering), and that only one of the buffers is filled at any given time. An error location is included so that violations of these assumptions can be detected. The transitions are not set as urgent in this automaton; the urgency in the composed system results from the controller.

Buffers The buffers are modeled each as an instantiation of the automaton shown in Fig 8(c). The outflow of the buffers is determined by the consumer, and therefore only known within the bounds $[-d_{i,max}, -d_{i,min}]$ (there is no valve to control outflow). The inflow is determined is equal to the outflow of the reactor. This leads to the dynamics $\dot{x}_i = [-d_{i,max}, -d_{i,min}] - \dot{x}_1$. Note that $\dot{x}_1$ is negative when the buffer is filling, so $\dot{x}_i$ is augmented by $-\dot{x}_1$ in this dynamics. Again, the transitions are not set as urgent; the urgency in the composed system results from the controller.

The specification was verified using SpaceEx/PHAVer (an implementation of the PHAVer reachability algorithm built on the SpaceEx platform). The inflow and outflow rates were set nondeterministically to be within intervals; the models incl. parameter values are available at `spaceex.imag.fr`. The computation of the complete reachable states shown in Fig. 9(a) takes 3.0 s and 24 MB of memory on a standard laptop. Finding the fixed point takes a total of 178 continuous post operations.

5 Conclusions

Linear Hybrid Automata stand out in the hybrid systems domain because sets of successor states can be carried out exactly. Available algorithms are require convex invariants and single-constraint urgency conditions. In this paper we propose algorithms that can handle non convex invariants and (closed) non-convex urgency conditions. The practical impact is that this extension can be used in order to model system in which transitions have to be taken as soon as possible. This is a common feature in several commercial tools used as de-facto standard in the industry (for example in the automotive context) such as Matlab/Simulink or Modelica. We formally proved the correctness and the termination of the proposed procedures, which are based on two operators: the first one is the classical continuous post operator $Post_\ell$ and the other one (defined here) is the so-called potential entry operator $pentry_\ell$. To the best of our knowledge, the proposed solutions represent the first sound and complete procedures for the task in the literature. We extended the tool PHAVer with our procedures and illustrated the results on an example.

The results from this paper will serve as the basis for future work on hybrid automata with more complex (piecewise affine and nonlinear) dynamics. The approach will need to be adapted since the specialized successor computations for these classes, e.g., in [12], only handle closed sets.

Bibliography

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

[2] R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008.

[3] 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.

[4] Nanette Bauer, Stefan Kowalewski, Guido Sand, and Thomas Löhl. A case study: Multi product batch plant for the demonstration of control and scheduling problems. In Sebastian Engell, Stefan Kowalewski, and Janan Zaytoon, editors, *ADPM'00*, pages 383–388. Shaker, 2000.

[5] D.A. Beek, M.A., Reniers, R.R.H., Schiffelers, and J.E. Rooda. Foundations of a compositional interchange format for hybrid systems. In *HSCC'07*, volume 4416 of *LNCS*, pages 587–600. Springer, 2007.

[6] G. Behrmann, A. David, and K. Larsen. A tutorial on uppaal. In *Formal Methods for the Design of Real-Time Systems*, volume 3185 of *LNCS*, pages 200–236. Springer, 2004.

[7] M. Benerecetti, M. Faella, and S. Minopoli. Automatic synthesis of switching controllers for linear hybrid systems: Safety control. *TCS: Theoretical Computer Science*, 493:116–138, 2012.

[8] Joseph T Buck, Soonhoi Ha, Edward A Lee, and David G Messerschmitt. *Ptolemy: A framework for simulating and prototyping heterogeneous systems.* Ablex Publishing Corporation, 1994.

[9] Christopher Chase, Joseph Serrano, and Peter J Ramadge. Periodicity and chaos from switched flow systems: contrasting examples of discretely controlled continuous systems. *Automatic Control, IEEE Transactions on*, 38(1):70–83, 1993.

[10] Pierre Collet and Jean Pierre Eckmann. *Iterated maps on the interval as dynamical systems*, volume 1. Springer, 1980.

[11] Alexandre Donzé and Goran Frehse. Modular, hierarchical models of control systems in spaceex. In *Control Conference (ECC), 2013 European*, pages 4244–4251. IEEE, 2013.

[12] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *CAV 11: Proc. of 23rd Conf. on Computer Aided Verification*, pages 379–395, 2011.

[13] Goran Frehse. PHAVer: algorithmic verification of hybrid systems past HyTech. *STTT*, 10(3):263–279, 2008.

[14] B. Gebremichael and Frits F. Vaandrager. Specifying urgency in timed i/o automata. In *IEEE Int. Conf. Software Engineering and Formal Methods*, SEFM '05, pages 64–74, 2005.

[15] T. A. Henzinger, Pei-Hsin Ho, and H. Wong-Toi. Hytech: the next generation. In *Proc. IEEE Real-Time Systems Symposium (RTSS '95)*, page 56. IEEE Computer Society, 1995.

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

[17] Pei-Hsin Ho. *Automatic Analysis of Hybrid Systems*. PhD thesis, Cornell University, August 1995. Technical Report CSD-TR95-1536.

[18] MathWorks. Mathworks stateflow: Design and simulate state machines, September 2012. `http://www.mathworks.fr/products/simulink/`.

[19] MathWorks. Mathworks simulink: Simulation et model-based design, March 2014. `www.mathworks.fr/products/simulink`.

[20] Sven Erik Mattsson, Hilding Elmqvist, and Martin Otter. Physical system modeling with modelica. *Control Engineering Practice*, 6(4):501–510, 1998.

[21] Stefano Minopoli and Goran Frehse. Non-convex invariants and urgency conditions on linear hybrid automata. Technical Report TR-2014-4, Verimag, April 2014. `www-verimag.imag.fr`.

[22] Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. An approach to the description and analysis of hybrid systems. In *Hybrid Systems*, pages 149–178. Springer, 1993.

[23] JPM Schmitz, DA Van Beek, and JE Rooda. Chaos in discrete production systems? *Journal of Manufacturing Systems*, 21(3):236–246, 2002.

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

[25] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *IEEE Conf. Decision and Control*, pages 4607 – 4612. IEEE, 1997.

[26] Martin Wulf, Laurent Doyen, and Jean-Franois Raskin. Almost asap semantics: From timed models to timed implementations. In *HSCC'04*, volume 2993 of *LNCS*, pages 296–310. Springer, 2004.