Theoretical Informatics and Applications Will be set by the publisher
Informatique Théorique et Applications

FORMAL LANGUAGE PROPERTIES OF HYBRID SYSTEMS WITH STRONG RESETS *

Thomas Brihaye[1], Véronique Bruyère[2] and Elaine Render[3]

Abstract. We study hybrid systems with strong resets from the perspective of formal language theory. We define a notion of hybrid regular expression and prove a Kleene-like theorem for hybrid systems. We also prove the closure of these systems under determinisation and complementation. Finally, we prove that the reachability problem is undecidable for synchronized products of hybrid systems.

1991 Mathematics Subject Classification. 68Q68, 68Q45.

Keywords and phrases: hybrid systems with strong resets, formal language theory

* *The research of the third author was supported by a Modnet grant. This work has been partially supported by a grant from the National Bank of Belgium, and by the grant 2.4530.02 of the Belgian Fund for Scientific Research (F.R.S.-FNRS)*

[1] Institute of Mathematics
University of Mons
20, Place du Parc
7000 Mons
Belgium; e-mail: thomas.brihaye@umons.ac.be

[2] Institute of Computer Science
University of Mons
20, Place du Parc
7000 Mons
Belgium; e-mail: veronique.bruyere@umh.ac.be

[3] School of mathematics
University of Manchester
Oxford Road
Manchester
United Kingdom; e-mail: e.render@maths.man.ac.uk

1. Introduction

Nowadays more and more real-life systems are (automatically) controlled by computer programs. It is of capital importance to know whether the programs governing these systems are correct; these questions gave rise to the theory of verification. In order to model real-life systems, various extensions of finite automata, such as *timed automata* [4] and *hybrid systems* [29], have been studied. Together with these models, various (temporal) logics, such as LTL [42] and CTL [24, 43], have been considered, which capture properties of the systems in which we are interested. For instance, an important question is to know whether the system can reach some prohibited states. This question is known as *the reachability problem*. Let $\mathcal{C}$ be a class of mathematical models and Φ be a class of properties (expressed in a given logic), then one can naturally ask the following question: Given $M \in \mathcal{C}$ and $\varphi \in \Phi$ is it true that "$M \models \varphi$?". This question is known as the *model-checking problem*.

Timed automata. Timed automata [3, 4] (see also [7] for a survey) are now a well-established formalism for the modeling and analysis of timed systems in the verification community. Roughly speaking timed automata are finite state automata enriched with clocks and clock constraints. A large number of positive and interesting theoretical results have been obtained on timed automata. Among them, is the decidability of the reachability problem [3] and the decidability of the model-checking problem for the quantitative temporal logic TCTL [1]. In parallel with these theoretical results, efficient verification tools have been implemented and successfully applied to industrially relevant case studies [30, 38]. Timed automata have also been studied from the perspective of formal language theory. In this framework surprising and negative results have quickly occurred. Indeed, in [4], the authors proved the undecidability of both the *universality problem* and the *language inclusion problem*. These negative results are due in some sense to the non-closure under complement of timed automata. Nevertheless, formal language properties of timed automata have been studied in various papers and by various authors: Kleene-like theorem [8, 16], definition of the determinisable subclass of event-clock timed automata [5, 25], power of ϵ-transitions [11], decidability problems regarding determinisability, complementability, and ϵ-transitions [15, 28, 45].

Hybrid systems. Hybrid systems [29] extend timed automata by allowing more complex continuous dynamics. Indeed, continuous variables are not restricted to clocks but can evolve as polynomial or exponential function. In the wide class of hybrid systems, the reachability problem is in general undecidable. In this context, an interesting question is to identify expressive classes of hybrid systems which are decidable (for example with respect to the reachability problem). Classes with this property include: *linear hybrid automata* [2], *(initialized) rectangular automata* [31], *weighted (or priced) timed automata* [6, 10], and *o-minimal hybrid systems* [19, 36].

In order to obtain decidability of the reachability problem, serious restrictions on either the continuous dynamics or the discrete transitions must be imposed.

O-minimal hybrid systems are a particularly interesting class of models since they admit very rich continuous dynamics but impose severe restrictions on the discrete transitions. Various nice positive results have been obtained in this class: existence of a finite bisimulation [20,36], decidability of the reachability problem [19], extension to games [12] and to weighted systems [13]. *Hybrid systems with strong resets* [14] extend o-minimal hybrid systems since they remove the o-minimality hypothesis but keep the restrictions on the discrete transitions.

Our contribution. In this article, we study hybrid systems with strong resets from the perspective of formal language theory. To the best of our knowledge, this study has not been done.

Our first result is a Kleene-like theorem based on the new notion of hybrid regular expressions. This first result is obtained thanks to two key results (Propositions 3.3 and 3.4) which show that the behaviour of a hybrid system $\mathcal{H}$ can be described by a (classical) finite automaton on an alphabet composed of couples (a, P) where a is an event letter of $\mathcal{H}$ and P is some piece of a finite partition of the time. This description using finite automata is made possible by the strong restrictions on the discrete transitions of the system $\mathcal{H}$.

From these key propositions we derive a series of nice positive results for the class of hybrid systems, all in contrast with the case of timed automata. Nondeterministic and deterministic models are equivalent in this class; we also find that we have closure under boolean operations. Since the reachability problem is known to be decidable for hybrid systems, it follows that the universality problem for this class is also decidable. Finally we study the synchronized product of several o-minimal hybrid systems. Synchronized product is an important notion in the context of verification; it may be seen as a generalization of the cartesian product. The synchronized product is not necessarily an o-minimal hybrid system. We prove the negative result that the reachability problem becomes undecidable as soon as seven o-minimal hybrid systems are synchronized.

Organization of the paper. In Section 2, we define and illustrate the notion of a hybrid system. We introduce *hybrid regular expressions* and prove a hybrid version of the Kleene Theorem in Section 3. In Section 4, we consider the closure properties of hybrid languages. Finally in Section 5 we consider synchronized products of o-minimal hybrid systems.

2. Preliminaries

In this section, we start with some basic definitions related to first order logic (our presentation is largely inspired from [21]). Then we recall the notion of hybrid system with strong resets and its canonical transition system. Finally we introduce the concept of timed languages accepted by hybrid systems.

In first-order logic, a *first-order structure* $\mathcal{M} = \langle M, (R_i)_{i\in I}, (f_j)_{j\in J}, (c_k)_{k\in K}\rangle$ consists of a domain M (some set), a family of relations $(R_i)_{i\in I}$ on M, a family of functions $(f_j)_{j\in J}$ on M and a family of constant $(c_k)_{k\in K}$ of M. The relation

R_i is a subset of M^{n_i} and f_j is a function from M^{n_j} to M. For instance the ordered group of real numbers $\mathcal{M}_R = \langle \mathbb{R}, \leq, +, 0 \rangle$ is a first-order structure where the domain is $\mathbb{R}$ (the set of real numbers) equipped with the usual order relation $\leq \subseteq \mathbb{R}^2$, the usual addition function $+ : \mathbb{R}^2 \to \mathbb{R}$ and the real constant 0. In order to define *first-order formulae*, we first need to define the notion of *term* of a given structure $\mathcal{M}$. In addition to the symbols of relations, functions and constants, we also use a countable set of *variables* $x, y, z, \ldots$, the usual *connectives* $\vee$ (or), $\wedge$ (and), $\neg$ (not), the *quantifiers* $\forall$ (for all), $\exists$ (there exists) and the symbol $=$ (equal). The terms are defined by the following grammar:

$$t ::= \ c \mid x \mid f(t, \ldots, t),$$

where c is a constant, x is a variable and f is a n-ary function. The formulae are defined by the following grammar:

$$\varphi ::= \ t_1 = t_2 \mid R_i(t_1, \ldots, t_n) \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \neg\varphi \mid \exists x\, \varphi \mid \forall x\, \varphi,$$

where the t_i's are terms, R_i is a n-ary relation and x is a variable. An example of first order formula in $\langle \mathbb{R}, \leq, +, 0 \rangle$ is given by

$$\forall x\, \forall y \ \ x + y = y + x.$$

The above formula asserts that the addition is a commutative operation. Let $\varphi(x_1, \ldots, x_n)$ be a first-order formula of the structure $\mathcal{M}$, where the variable x_i's are not under the scope of a quantifier, and $m_1, \ldots, m_n$ be elements of M, when we write $\mathcal{M} \models \varphi(m_1, \ldots, m_n)$, we mean that the formula $\varphi(x_1, \ldots, x_n)$ is true when the (free) variables x_i's are replaced by the elements m_i's of M. We say that a subset $S \subseteq M^n$ is *first-order definable in* $\mathcal{M}$ if there exists a first-order formula $\varphi(x_1, \ldots, x_n)$ such that

$$S = \{(m_1, \ldots, m_n) \mid \mathcal{M} \models \varphi(m_1, \ldots, m_n)\}.$$

For instance, the set of non-negative real numbers is definable in $\mathcal{M}_R$ by the first-order formula $\psi(x)$ given by $x \geq 0$. We say that a function is first-order definable when its graph is a first-order definable set. In the sequel of the paper, when we speak of a formula, we always mean a first-order formula and when we say that a subset or a function is definable over $\mathcal{M}$, we always mean it is first-order definable over $\mathcal{M}$. General references for first-order logic are [23, 33, 34, 39, 44].

Let us now formalize our notion of *dynamical system*.

Definition 2.1. A *dynamical system* is a pair $(\mathcal{M}, \gamma)$ where:

- $\mathcal{M} = \langle M, <, 0, \ldots \rangle$ is an expansion of a totally ordered structure with a symbol of constant 0,
- $\gamma : M^{k_1} \times M \to M^{k_2}$ is a function definable in $\mathcal{M}$, for some fixed k_1, $k_2 \in \mathbb{N}$.

The function γ is called the *dynamics* of the dynamical system. More generally, we may consider the case where γ is defined on definable subsets of $\mathcal{M}$, that is $\gamma : V_1 \times V \to V_2$ with $V_1 \subseteq M^{k_1}$, $V \subseteq M$ and $V_2 \subseteq M^{k_2}$ being definable subsets.

Classically when $\mathcal{M}$ is the ordered field of reals, we see V as the time, $V_1 \times V$ as the space-time, V_2 as the output space and V_1 as the input space. We keep this terminology in the more general context of a structure $\mathcal{M}$.

Example 2.2. Consider $(\mathcal{M}, \gamma)$ where each point of the plane has two possible behaviors: "to go to the right" or "to go up" (see Figure 1). More precisely we have that $\mathcal{M} = \langle \mathbb{R}, <, +, 0 \rangle$ and $\gamma : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^2$ is defined as follows.

$$\gamma(x_1, x_2, p, t) = \begin{cases} (x_1 + t, x_2) & \text{if } p \geq 0, \\ (x_1, x_2 + t) & \text{if } p < 0. \end{cases}$$

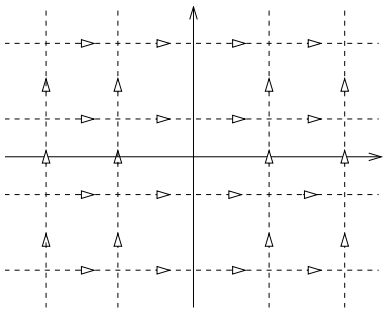

FIGURE 1. A "non-deterministic" dynamical system.

We are now ready to define the key structure for this paper, a *hybrid system with strong resets*[1].

Definition 2.3. Given $\mathcal{M}$ an ordered structure, a *hybrid system with strong resets on* $\mathcal{M}$ is given by $\mathcal{H} = (Loc, \Sigma, Edg, Dyn, Inv, \mathcal{G}, \mathcal{R})$, where:

- Loc is a finite set of locations (discrete states),
- Σ is a finite alphabet of events,
- $Edg \subseteq Loc \times \Sigma \times Loc$ is a finite set of edges,
- Dyn assigns continuous dynamics to each location (there exist some fixed definable subsets V, V_1, V_2 such that for each $l \in Loc$, $Dyn(l) = \gamma_l$ where $\gamma_l : V_1 \times V \to V_2$ is a definable function), i.e. $(\mathcal{M}, \gamma_l)$ *is a dynamical system*,
- Inv assigns to each location $l \in Loc$ a definable subset $Inv(l)$ of V_2 called *invariant*,
- $\mathcal{G}$ assigns to each edge $e \in Edg$ a definable subset $\mathcal{G}(e)$ of V_2 called *guard*,

[1]In the sequel of the paper, when we speak of a hybrid system, we always mean a hybrid system with strong resets.

- $\mathcal{R}$ assigns to each edge $e \in Edg$ a definable subset $\mathcal{R}(e)$ of V_2 called *reset*.

To define the semantics of a given hybrid system $\mathcal{H}$, we define the *canonical transition system* $T^{can}_{\mathcal{H}} = (Q_{\mathcal{H}}, \rightarrow)$ with:

- set of states $Q_{\mathcal{H}}$ is $Loc \times V_2$,
- a transition relation $\rightarrow$ given by:

$$(l_1, y_1) \rightarrow (l_2, y_2) \Leftrightarrow \exists\ e \in Edg,\ \exists (l_1, y_1') \in Loc \times V_2 \text{ such that}$$
$$(l_1, y_1) \xrightarrow{t} (l_1, y_1') \xrightarrow{a} (l_2, y_2)$$

where $(l_1, y_1) \xrightarrow{t} (l_1, y_1')$ is a *continuous transition* with $t \geq 0$ such that $\exists x \in V_1, \exists \delta_1 \in V$,

$$y_1 = \gamma_{l_1}(x, \delta_1),\ y_1' = \gamma_{l_1}(x, \delta_1 + t) \text{ and } \gamma_{l_1}(x, \delta) \in Inv(l_1)\ \forall \delta, \delta_1 \leq \delta \leq \delta_1 + t$$

and where $(l_1, y_1') \xrightarrow{a} (l_2, y_2)$ is a *discrete transition* such that

$$e = (l_1, a, l_2),\ y_1' \in \mathcal{G}(e) \text{ and } y_2 \in \mathcal{R}(e).$$

In our definition of continuous transition, t is the *duration* spent at location l_1. Note that $t \in M^+$, where $M^+ = \{x \in M \mid x \geq 0\}$. A *canonical run* in a hybrid system is a run of its canonical transition system.

Remark 2.4. Note that our definition of hybrid system allows only resets as defined in [36], that is, discrete transitions are *memoryless*: each variable must be non-deterministically reset into some definable subset.

Below is an example of a hybrid system inspired by the thermostat example of [2].

Example 2.5. The temperature of a room has to be kept between m and M degrees. The room is equipped with a *thermostat* which senses the temperature and turns a heater on and off. The temperature is governed by two differential equations. We denote the temperature by the variable x. When the heater is off, the temperature decreases according to the function $x(t) = \theta e^{-Kt}$; when the heater is on, the temperature increases according to the function $x(t) = \theta e^{-Kt} + h(1 - e^{-Kt})$, where t is the time, θ the initial temperature, and h and K are parameters for the heater and the room. This situation is described by the hybrid system of Figure 2.

The thermostat is a hybrid system definable in $\langle \mathbb{R}, <, +, \cdot, 0, 1, e^x \rangle$ as soon as h and K are definable constants. Making this precise in view of Definition 2.3:

- $Loc = \{l_0, l_1\}$,
- $\Sigma = \{\mathsf{on}, \mathsf{off}\}$,
- $e_1 = (l_0, \mathsf{on}, l_1) \in Edg$, $e_2 = (l_1, \mathsf{off}, l_0) \in Edg$,
- $Dyn(l_0) = \gamma_0$ where $\gamma_0 : \mathbb{R}^2 \rightarrow \mathbb{R}$ is given by $\gamma_0(\theta, t) = \theta e^{-Kt}$,
  $Dyn(l_1) = \gamma_1$ where $\gamma_1 : \mathbb{R}^2 \rightarrow \mathbb{R}$ is given by $\gamma_1(\theta, t) = \theta e^{-Kt} + h(1 - e^{-Kt})$,

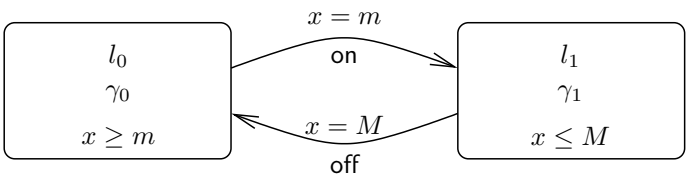

FIGURE 2. Thermostat

- $Inv(l_0) = \{x \in \mathbb{R} \mid x \geq m\}$, $Inv(l_1) = \{x \in \mathbb{R} \mid x \leq M\}$,
- $\mathcal{G}(e_1) = \{m\}$, $\mathcal{G}(e_2) = \{M\}$,
- $\mathcal{R}(e_1) = \{m\}$, $\mathcal{R}(e_2) = \{M\}$.[2]

In the figures throughout this paper we distinguish between guards and resets as follows: Given $e \in Edg$, we denote the associated guard by $y \in \mathcal{G}(e)$ and the associated reset by $y := \mathcal{R}(e)$.

In order to define the notion of o-minimal hybrid system as introduced in [36], we first recall the notion of o-minimality [41].

Definition 2.6. A totally ordered structure $\mathcal{M} = \langle M, <, \ldots \rangle$ is *o-minimal* if every definable subset of M is a finite union of points and open intervals (possibly unbounded). A hybrid system is *o-minimal* if the underlying structure is o-minimal.

In an o-minimal structure, the definable subsets of M are thus the simplest possible: the ones which are definable with parameters in $\langle M, < \rangle$. This assumption implies that definable subsets of M^n (in the sense of $\mathcal{M}$) admit very nice structure theorems like *Cell decomposition* and *uniform finiteness* [35] (see also [27] for a good introduction to o-minimality and an extensive bibliography). The following are examples of o-minimal structures.

Example 2.7.
- The ordered group of rationals $\langle \mathbb{Q}, <, +, 0 \rangle$.
- The ordered field of reals $\langle \mathbb{R}, <, +, \cdot, 0, 1 \rangle$.
- The ordered field of reals with exponential function $\langle \mathbb{R}, <, +, \cdot, 0, 1, e^x \rangle$ [47].

For more interesting examples of o-minimal structures see [27].

We now consider the language of timed words accepted by a given (o-minimal or not) hybrid system $\mathcal{H}$. For this, we need to define notions of *timed word* and *timed language*.

Definition 2.8. A *timed word* w over a finite alphabet Σ is a finite sequence

$$(a_1, t_1)(a_2, t_2) \ldots (a_n, t_n)$$

of pairs of letters $a_i \in \Sigma$ and $t_i \in M^+$. The *length* of w is equal to n and its *duration* is equal to $\Sigma_{i=1}^n t_i$. The *timed empty word* is of the form $(\epsilon, 0)$ with length and duration equal to 0.

A *timed language* over Σ is a set of timed words over Σ.

[2] On Figure 2 the resets are not explicitly mentioned since they have no effect on x.

Remark 2.9. Notice that an equivalent notion of timed word is given by the usual definition $(a_1, t_1)(a_2, t_1 + t_2) \dots (a_n, \sum_{i=1}^{n} t_i)$ as proposed for timed automata [4], instead of the definition $(a_1, t_1)(a_2, t_2) \dots (a_n, t_n)$ here given. Due to the strong resets condition it is preferable to use the latter one.

In order to consider timed languages accepted by hybrid systems we must introduce an acceptance condition to the definition of the hybrid system $\mathcal{H}$. First we define the initial condition, $\mathsf{Init} \subseteq Loc \times V_2$ such that $\mathsf{Init} = \cup_{l \in Loc}(l, \mathsf{Init}_l)$ and with each Init_l definable (with possibly $\mathsf{Init}_l = \emptyset$). The termination condition is given in a similar way by $\mathsf{Term} \subseteq Loc \times V_2$.

Definition 2.10. A timed word $w = (a_1, t_1)(a_2, t_2) \dots (a_n, t_n)$ is accepted by a hybrid system $\mathcal{H}$ with initial condition Init and termination condition Term if there exists a canonical run

$$(l_0, y_0) \to (l_1, y_1) \to (l_2, y_2) \to \cdots \to (l_n, y_n)$$

such that

- each transition $(l_{i-1}, y_{i-1}) \to (l_i, y_i)$ decomposes into

$$(l_{i-1}, y_{i-1}) \xrightarrow{t_i} (l_{i-1}, y'_{i-1}) \xrightarrow{a_i} (l_i, y_i),$$

- $(l_0, y_0) \in \mathsf{Init}$,
- $(l_n, y_n) \in \mathsf{Term}$.

The set of timed words accepted by a hybrid system $\mathcal{H}$ is denoted by $L_{\mathcal{H}}$.

Note that the timing information attached to each letter from Σ in a timed word denotes the amount of time which has passed since the last discrete transition on the run through the transition system. Note also that the timed empty word is accepted by a hybrid system $\mathcal{H}$ if and only if we have $\mathsf{Init}_l \cap \mathsf{Term}_l \neq \emptyset$ for some location l in $\mathcal{H}$.

We now give a very simple example illustrating the above definitions.

Example 2.11. Consider the hybrid system $\mathcal{H}$ depicted in Figure 3. We assume that the dynamics of the two locations are given by $\gamma : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ where $\gamma(x, t) = x + t$ (this corresponds to the continuous dynamics of timed automata). The invariant of each location is the interval $[0, 1]$. The initial condition for the system is $\mathsf{Init} = (\mathsf{l}_0, \{\mathsf{0}\}) \cup (\mathsf{l}_1, \emptyset)$ and the termination condition is $\mathsf{Term} = (\mathsf{l}_0, \emptyset) \cup (\mathsf{l}_1, [\mathsf{0}, \mathsf{1}])$. Examples of timed words accepted by $\mathcal{H}$ are $(b, 0.7)$ and $(a, 1)(a, 1)(b, \frac{\pi}{4})$. It is straightforward to observe that the timed language accepted by $\mathcal{H}$ is given by:

$$L_{\mathcal{H}} = \{(a, 1)^n (b, t) \mid n \in \mathbb{N} \text{ and } t \in \mathbb{R} \text{ with } 0 \leq t < 1\}.$$

We conclude this section with some comments. The notion of (o-minimal) hybrid system presented in this section allows very rich dynamics inside each location. In contrast, a strong reset condition has to be imposed on the edges. In this

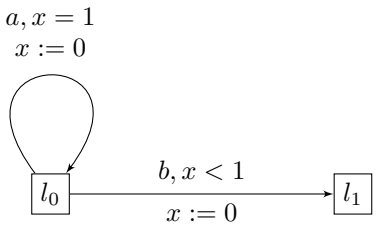

FIGURE 3. A simple hybrid system.

way, the decidability of the reachability problem for o-minimal hybrid systems is ensured (see [18, 37]) assuming the decidability of the underlying structure. Note that our definition of canonical run in a hybrid system (that is, runs alternating discrete and continuous transitions) is somewhat more limited than the standard definition. However, when this condition is relaxed, undecidability of the reachability problem for o-minimal hybrid systems is quickly reached [17].

3. Hybrid regular expressions

In this section we define the notion of hybrid regular expression, and show the equivalence between timed languages represented by hybrid regular expressions and timed languages accepted by hybrid systems. We define such regular expressions in terms of the event alphabet Σ and definable subsets of M^+, a natural when considering the memoryless reset conditions imposed on the edges of the system.

Definition 3.1. Let $\mathcal{M} = \langle M, <, 0, \ldots \rangle$ be a structure. Let $\mathcal{P} = \{P_i \mid 1 \leq i \leq n\}$ be a finite partition of M^+ into definable non-empty pieces, and let Σ be a finite alphabet of events. A *hybrid regular expression* E is a regular expression over the alphabet $\Sigma \times \mathcal{P}$, that is, defined by the following grammar

$$E ::= \emptyset \mid (a, P) \mid E \cup E \mid E \cdot E \mid E^*$$

with $a \in \Sigma$ and $P \in \mathcal{P}$.

A *hybrid regular language* is a timed language $L(E)$ *represented by* a hybrid regular expression E in the following sense[3] :

- if $E = (a, P)$ (respectively, $\emptyset$), then $L(E) = \{(a, t) \mid t \in P\}$ (respectively, $\emptyset$),
- if $E = E_1 \cup E_2$ (respectively, $E_1 \cdot E_2$, E_1^*), then $L(E) = L(E_1) \cup L(E_2)$ (respectively, $L(E_1) \cdot L(E_2)$, $L(E_1)^*$).

[3]The regular operators are defined on sets of timed words in the classical way as for finite words.

Note that we may associate a second semantics, called *abstract*, with a hybrid regular expression E. It is defined as the classical one considered over the alphabet $\Sigma \times \mathcal{P}$, that is,

- if $E = (a, P)$ (respectively, $\emptyset$), then $L_A(E) = \{(a, P)\}$ (respectively, $\emptyset$),
- if $E = E_1 \cup E_2$ (respectively, $E_1 \cdot E_2$, E_1^*), then $L_A(E) = L_A(E_1) \cup L_A(E_2)$ (respectively, $L_A(E_1) \cdot L_A(E_2)$, $L_A(E_1)^*$).

Given a hybrid regular expression E, a finite automaton accepting $L_A(E)$ is called *abstract*.

Example 3.2. Returning to the example in Figure 3, the hybrid regular expression representing the language $L_{\mathcal{H}}$ (accepted by the hybrid system $\mathcal{H}$) is easily seen to be the following

$$(a, \{1\})^* \cdot (b, [0, 1)).$$

With the hybrid regular language $L_{\mathcal{H}}$, one can thus also associate the abstract finite automaton of Figure 4 over the alphabet $\{(a, \{1\}), (b, [0, 1))\}$.

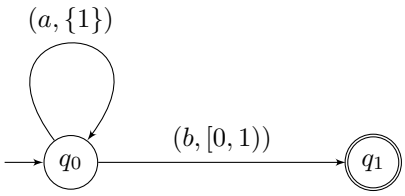

FIGURE 4. An abstract finite automaton accepting the hybrid regular expression representing the language $L_{\mathcal{H}}$.

We now state the two key results of this article, which rely heavily on the strong reset condition. The first result has already been illustrated by Figure 4. Example 3.6 below is another more involved illustration.

Proposition 3.3. *Let $\mathcal{M} = \langle M, +, <, 0, \ldots \rangle$ be an expansion of an ordered group, and let $\mathcal{H}$ be a hybrid system with initial and terminal conditions, accepting a timed language $L_{\mathcal{H}}$. Then one may define*

- *a finite alphabet $\Sigma \times \mathcal{P}$ with Σ the event alphabet of $\mathcal{H}$ and $\mathcal{P} = \{P_i \mid 1 \leq i \leq n\}$ a finite partition of M^+ definable in $\mathcal{M}$,*
- *an abstract finite automaton $\mathcal{A}_{\mathcal{H}}$ accepting a language $L = L_A(E)$ for a hybrid regular expression E over $\Sigma \times \mathcal{P}$ such that E represents the timed language $L_{\mathcal{H}}$.*

Proposition 3.4. *Let E be a hybrid regular expression. Then there exists a hybrid system accepting exactly the timed language represented by E.*

Combining the previous two propositions allows us to conclude the following.

Theorem 3.5. *A timed language L is accepted by a hybrid system if and only if L is a hybrid regular language.*

What follows is a second example to demonstrate the fundamentals of the equivalence between a hybrid system, a hybrid regular expression and the associated abstract finite automaton, (Proposition 3.3). Proposition 3.4 will be illustrated later in Example 4.3.

Example 3.6. Consider the hybrid system in Figure 5. Both locations in the system have the same dynamics, that is $\gamma(x,t) = x + t$. Similarly both locations share the same invariant, $Inv_{l_0} = Inv_{l_1} = [0,4]$. The initial condition for the system is $\mathsf{Init} = (l_0, [0,1]) \cup (l_1, \emptyset)$ and the termination condition is $\mathsf{Term} = (l_0, \emptyset) \cup (l_1, [0,4])$. For reasons of clarity, no resets have been included in the figure: all edges share the same reset $\mathcal{R} = [0,1]$.

This system accepts the hybrid regular language $L(E)$ represented by the following hybrid regular expression

$$E \quad = \quad [(b,[2,4])^+(a,[0,2])^*][(b,[2,4])(b,[2,4])^+(a,[0,2])^*]^*, \tag{1}$$

The associated abstract finite automaton is given in Figure 6.

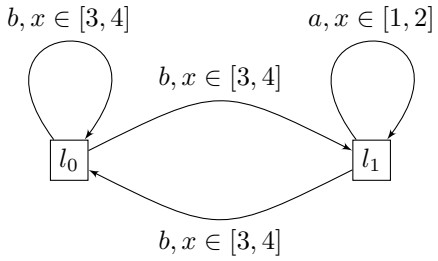

FIGURE 5. An example of a hybrid system.

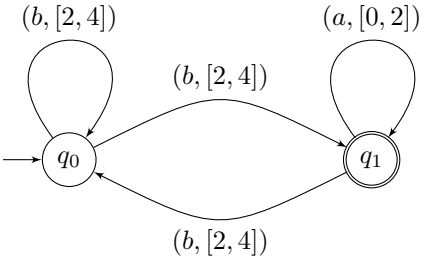

FIGURE 6. An abstract automaton for the expression (1)

We provide some intuition on this example. All incoming edges to the location l_0 of the hybrid system of Figure 5 have the same reset $[0, 1]$, and all outgoing edges have the same guard $[3, 4]$. One may check that the set of all possible durations $t \in M^+$ in a continuous transition $(l_0, y) \xrightarrow{t} (l_0, y')$ is equal to the interval $[2, 4]$. After such a continuous transition, one may perform a discrete transition $(l_0, y') \xrightarrow{b} (l_0, z)$. This situation is summarized by the loop on q_0 labeled by $(b, [2, 4])$ on the automaton of Figure 6.

The alphabet in this example is $\Sigma \times \mathcal{P}$ where $\Sigma = \{a, b\}$ and $\mathcal{P} = \{[0, 2], [2, 4]\}$. To respect Definition 3.1, we should define $\mathcal{P}$ equal to $\{[0, 2), \{2\}, (2, 4], (4, +\infty)\}$ to give a partition of $\mathbb{R}^+$.

We now prove the preceding propositions. The proof of Proposition 3.3 is based on the following observation. Assume that a location l of the system $\mathcal{H}$ has all its incoming edges with the same reset and all its outgoing edges with the same guard. Then the set of durations $t \in M^+$ which may appear in a continuous transition $(l, y_1) \xrightarrow{t} (l, y_2)$ is a set definable in $\mathcal{M}$. These sets for all locations l give rise to a finite partition of M^+ leading to the definition of $\mathcal{P}$.

Proof of Proposition 3.3. Let $\mathcal{H}$ be a hybrid system with initial condition Init and termination condition Term. We assume for now that the timed language accepted by $\mathcal{H}$ does not include the timed empty word. We perform several transformations on the system to build a finite automaton accepting a hybrid regular expression which represents exactly the timed language accepted by $\mathcal{H}$. The transformation described in (a) and (b) below is a "classical" transformation, used often in automata theory.

(a) Removing self-loops. Let $l \in Loc$ be a location in the hybrid system $\mathcal{H}$ which has a self looping edge $e = (l, a, l) \in Edg$ for some $a \in \Sigma$. Then we make a copy of the location l so that we have two identical locations l and l' (with the same dynamics and invariant as the original location), with identical incoming and outgoing edges except for the loop e. We replace the loop with two edges $e = (l, a, l')$ and $e' = (l', a, l)$ with the same guard and reset as the original edge. See Figure 7.

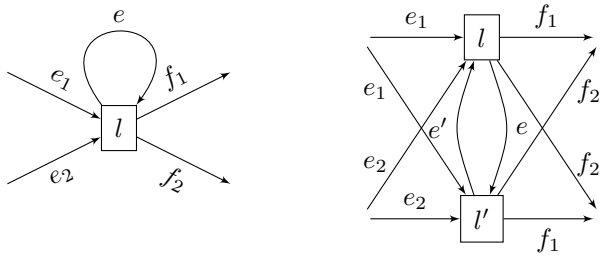

FIGURE 7. Step (a): Removing self loops

In the initial condition, (l, Init_l) is replaced by $(l, \mathsf{Init}_l) \cup (l', \emptyset)$, and in the termination condition, (l, Term_l) is replaced by $(l, \mathsf{Term}_l) \cup (l', \mathsf{Term}_l)$.

Note that after the transformation, the hybrid regular language $L_{\mathcal{H}'}$ accepted by the transformed system $\mathcal{H}'$ is equal to $L_{\mathcal{H}}$. Indeed, if a timed word $(a_1, t_1)(a_2, t_2) \dots (a_n, t_n)$ belongs to $L_{\mathcal{H}'}$ it also belongs to $L_{\mathcal{H}}$ (replace all occurrences of l' with l). Conversely any timed word of $L_{\mathcal{H}}$ also belongs to $L_{\mathcal{H}'}$. Indeed the related canonical run ρ of $\mathcal{T}^{can}_{\mathcal{H}}$ can be transformed into a canonical run ρ' of $\mathcal{T}^{can}_{\mathcal{H}'}$, such that each occurrence of l in ρ is replaced by either l or l' in ρ'; ρ' begins with l if ρ begins with l and ρ' ends with either l or l' if ρ ends with l.

(b) Normalizing guards and resets for each location. Our aim in this step is to ensure that for any given location l, all outgoing edges share the same guard, and all incoming edges share the same reset. This will be achieved in two steps, the first of which will deal with the resets. Consider a location l in $\mathcal{H}$. Let $e_1, \dots, e_n \in Edg$ be edges in the system with target location l, and let $\mathcal{R}_1, \dots, \mathcal{R}_m$ be the resets of these edges. Note that $m \leq n$ since two or more edges may share the same reset. We partition the edges by the resets into m subsets. Let the new states labeled $l_1, \dots, l_m$ be m copies of l with the same dynamics and invariant. Each location l_i, $(i = 1, \dots, m)$, has identical copies of all outgoing edges from l. The location l_i has as incoming edges copies of only those edges e_j, $j = 1, \dots, n$, where $\mathcal{R}(e_j) = \mathcal{R}_i$. See Figure 8.

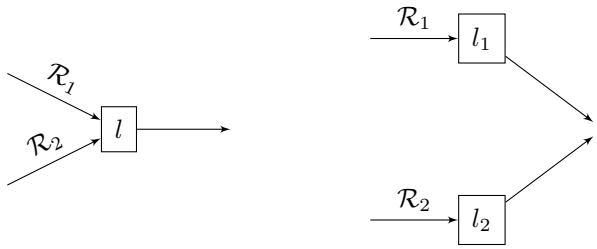

FIGURE 8. Step (b): Normalize resets for each location

In the case that $\mathsf{Init}_l \neq \emptyset$ we take an extra copy of l (additional to $l_i, i = 1, \dots, m$) denoted by l_I which has the same invariant, dynamics and outgoing edges of l, but has no incoming edges. In the initial condition, (l, Init_l) is replaced by

$$(l_I, \mathsf{Init}_l) \cup \bigcup_{1 \leq i \leq m} (l_i, \emptyset).$$

The location l_I is called *initial*. In the case that $\mathsf{Init}_l = \emptyset$, then (l, Init_l) is simply replaced by $\bigcup_{1 \leq i \leq m} (l_i, \emptyset)$.

Next we consider the termination condition. Suppose that a canonical run ρ which is accepted by $\mathcal{H}$ has a discrete transition with reset $\mathcal{R}$ as its final transition in $\mathcal{H}$, ending in a state (l, y). Hence $y \in \mathcal{R} \cap \mathsf{Term}_l$. So for each copy l_i of l such that $\mathcal{R}_i \cap \mathsf{Term}_l \neq \emptyset$ we add an extra copy of l_i denoted $l_{T,i}$ with the same invariant, dynamics and incoming edges as l_i, but with no outgoing edges. Such a location $l_{T,i}$ is called *terminal*. Then the term (l, Term_l) in the termination condition is replaced by

$$\bigcup_{1 \leq i \leq m} ((l_{T,i}, \mathsf{Term}_l) \cup (l_i, \emptyset)).$$

In this way, a canonical run that is accepted by the original system is translated into a canonical run of the transformed system which begins with an initial location (of the form l_I) and ends at a terminal location (of the form $l_{T,i}$ for some i).

It is clear from a similar argument to case (a) above that the hybrid regular language $L_{\mathcal{H}}$ is unchanged, and that given a location l, all incoming edges will share the same reset.

Note that the *removing self-loops* transformation is necessary in order to guarantee the correctness of the *normalizing guards and resets* transformation. Notice also that the latter transformation does not introduce new self-loops in the automaton.

We perform a similar operation on the outgoing edges of locations to ensure that all such edges from a given location will share the same guard. Let l be a location in our hybrid system with outgoing edges $e_1, \ldots, e_n$, with guards $\mathcal{G}_1, \ldots, \mathcal{G}_m$ with $m \leq n$. We replace location l by m copies of l labeled $l_1, \ldots, l_m$ with the same dynamics and invariants as l. For each $i = 1, \ldots, m$, l_i has incoming edges identical to those of l, and we include only outgoing edges e_j, $j = 1, \ldots, n$, such that $\mathcal{G}(e_j) = \mathcal{G}_i$.

In the case that $\mathsf{Init}_l \neq \emptyset$, we take m extra copies $l_{I,i}$, $i = 1, \ldots, m$, with no incoming edges, and replace (l, Init_l) in the initial condition with

$$\bigcup_{1 \leq i \leq m} ((l_{I,i}, \mathsf{Init}_l) \cup (l_i, \emptyset)).$$

Again, the locations $l_{I,i}$ are called initial. In the case that $\mathsf{Init}_l = \emptyset$, we simply replace (l, Init_l) by $\bigcup_{1 \leq i \leq m}(l_i, \emptyset)$.

Let $\mathcal{R}$ be the (unique) reset for all the edges that have location l as target. If $\mathsf{Term}_l \cap \mathcal{R} \neq \emptyset$, we take one extra copy of l labeled l_T with no outgoing edges, and replace (l, Term_l) in the termination condition by

$$(l_T, \mathsf{Term}_l) \cup \bigcup_{1 \leq i \leq m} (l_i, \emptyset).$$

The location l_T is called terminal. If $\mathsf{Term}_l \cap \mathcal{R} = \emptyset$, we replace (l, Term_l) by $\bigcup_{1 \leq i \leq m}(l_i, \emptyset)$.

By the same reasoning as above we have a hybrid system which accepts the same language $L_{\mathcal{H}}$ as the original system. See Figure 9.

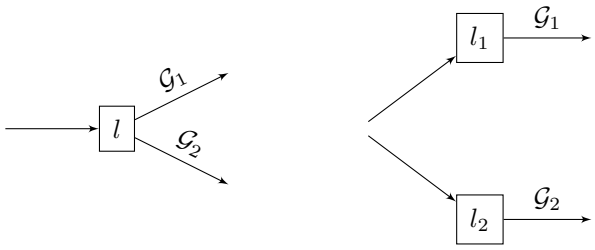

FIGURE 9. Step (b): Normalize guards for each location

(c) Relabeling locations and edges. Our aim now is to define a finite partition $\mathcal{P}$ of M^+ which encapsulates all of the timing information. The hybrid system is then replaced by a *graph* denoted $\mathcal{G}_{\mathcal{H}}$ with the same set of locations and edges as $\mathcal{H}$; locations will be labeled by pieces of $\mathcal{P}$ and edges will be labeled from Σ.

After steps (a) and (b), for any given location l in our system, all incoming edges have the same reset, say $\mathcal{R}$, and all outgoing edges have the same guard, say $\mathcal{G}$, and let l have invariant Inv. Then the set of possible time durations which may pass at location l is a definable subset of M^+:

$$T(l) = \{t \in M^+ \mid \exists \bar{x} \in V_1\ \exists \bar{y}_1, \bar{y}_2 \in V_2\ \exists \delta_1, \delta_2 \in V\ (\delta_2 = t + \delta_1) \wedge (\gamma_l(\bar{x}, \delta_1) = \bar{y}_1) \\ \wedge (\gamma_l(\bar{x}, \delta_2) = \bar{y}_2) \wedge (\bar{y}_1 \in \mathcal{R}) \wedge (\bar{y}_2 \in \mathcal{G}) \wedge \forall \delta\ \delta_1 \leq \delta \leq \delta_2\ \gamma_l(\bar{x}, \delta) \in Inv\}.$$

Recall that some locations are initial. Thus in the case of initial locations, we have instead the following set:

$$T_I(l) = \{t \in M^+ \mid \exists \bar{x} \in V_1\ \exists \bar{y}_1, \bar{y}_2 \in V_2\ \exists \delta_1, \delta_2 \in V\ (\delta_2 = t + \delta_1) \wedge (\gamma_l(\bar{x}, \delta_1) = \bar{y}_1) \\ \wedge (\gamma_l(\bar{x}, \delta_2) = \bar{y}_2) \wedge (\bar{y}_1 \in \mathsf{Init_l}) \wedge (\bar{\mathsf{y}}_2 \in \mathcal{G}) \wedge \forall \delta\ \delta_1 \leq \delta \leq \delta_2\ \gamma_l(\bar{\mathsf{x}}, \delta) \in \mathsf{Inv}\}.$$

Notice that the sets $T(l)$ and $T_I(l)$ may be empty. It is clear that there exists a finite partition $\mathcal{P}$ of M^+ into non-empty sets which respects the sets defined above since there is a finite number of locations. Each piece of $\mathcal{P}$ is a definable subset of M^+. Then the sets $T(l)$ and $T_I(l)$ are rewritten as finite (possibly empty) unions of partition pieces.

The graph $\mathcal{G}_{\mathcal{H}}$ is defined as follows. All information is removed from each location l (except for the property of being initial or terminal) and replaced with the label $T(l)$ (respectively $T_I(l)$), a union of partition pieces

of the time M^+. Similarly, we remove the resets and guards from edges $e \in Edg$ leaving only the event label from Σ.

(d) Reducing definable subsets to partition pieces and relabeling edges. This is the last step of the transformation; here we define the abstract finite automaton $\mathcal{A}_{\mathcal{H}}$ introduced in the statement of Proposition 3.3. For each location l in the graph $\mathcal{G}_{\mathcal{H}}$ with label $T = P_1 \cup \ldots \cup P_n$ (with P_i a piece of $\mathcal{P}$ for $i = 1, \ldots, n$), l is replaced by n copies of the location labeled $l_1, \ldots, l_n$ with identical incoming and outgoing edges. The label of l_i is the piece P_i. If l is initial, each copy is initial. If l is terminal, each copy of l is terminal. In this way,each location in the graph is labeled by a single partition piece. See Figure 10.

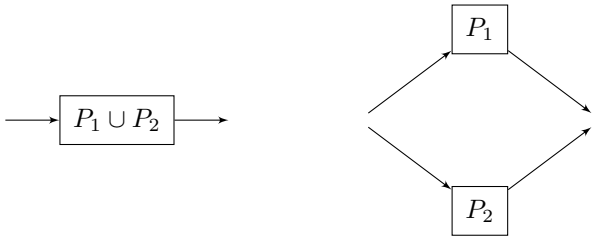

FIGURE 10. Step (d): Reduce definable sets labeling locations to partition pieces

It remains to move the labels of the locations to the edges. Let l be a location with label P. For each outgoing edge e with label $a \in \Sigma$ we relabel the edge (a, P) and remove the label P from the location. We then consider our graph as a finite automaton $\mathcal{A}_{\mathcal{H}}$ in the usual sense with initial and terminal states those locations labeled in this way in the earlier stage of the construction. Our finite automaton is labeled from $\Sigma \times \mathcal{P}$ where $\mathcal{P}$ is the finite partition of the time M^+ defined above. Below we prove that the finite automaton defined in this way accepts the language $L_A(E)$ for a hybrid regular expression E representing the timed language accepted by the original hybrid system $\mathcal{H}$.

The automaton $\mathcal{A}_{\mathcal{H}}$ accepts a language over the alphabet $\Sigma \times \mathcal{P}$ represented by a (classical) regular expression E. We must show that E seen as a hybrid regular expression represents exactly $L_{\mathcal{H}}$, the timed language accepted by $\mathcal{H}$.

We know that after steps (a) and (b), the transformed hybrid system accepts the same timed language $L_{\mathcal{H}}$ as the original hybrid system $\mathcal{H}$. So let $\mathcal{H}$ denote the transformed system, and argue directly on it.

Let w be a timed word accepted by $\mathcal{H}$, with $w = (a_1, t_1) \ldots (a_n, t_n)$ for some $n \in \mathbb{N} \setminus \{0\}$. Then there exists a canonical run ρ of $T^{can}_{\mathcal{H}}$

$$(l_0, y_0) \to (l_1, y_1) \to \cdots \to (l_n, y_n)$$

such that l_0 is an initial location with $(l_0, y_0) \in \mathsf{Init}$, l_n is a terminal location with $(l_n, y_n) \in \mathsf{Term}$, and each transition $(l_{i-1}, y_{i-1}) \to (l_i, y_i)$ decomposes into

$$(l_{i-1}, y_{i-1}) \xrightarrow{t_i} (l_{i-1}, y'_{i-1}) \xrightarrow{a_i} (l_i, y_i).$$

Consider first (a_1, t_1). There exists an edge e_1 labeled with a_1 connecting the initial location l_0 to the location l_1, and there exist $\delta \in V$, $\bar{x} \in V_1$ such that $\gamma_{l_0}(\bar{x}, \delta) \in \mathsf{Init}_{l_0}, \gamma_{l_0}(x, \delta + t_1) \in \mathcal{G}(e_1)$ and $\gamma_{l_0}(\bar{x}, \delta') \in Inv(l_0)$ for all δ' such that $\delta \leq \delta' \leq \delta + t_1$. Then it is clear that $t_1 \in T_I(l_0)$. Let P_1 be the partition piece of $T_I(l_0)$ in which t_1 appears. Then in the automaton $\mathcal{A}_\mathcal{H}$, there exists an edge connecting an initial state q_0 to some state q_1 labeled with $(a_1, P_1) \in \Sigma \times \mathcal{P}$.

Consider next (a_i, t_i), the i-th letter in the word w, where $1 < i \leq n$. In ρ we have reached a state (l_{i-1}, y_{i-1}) such that the location l_{i-1} has an outgoing edge e_i labeled with a_i and there exists $\bar{x} \in V_1, \delta \in V$ such that $\gamma_{l_{i-1}}(\bar{x}, \delta) \in \mathcal{R}(e_{i-1})$, $\gamma_{l_{i-1}}(\bar{x}, \delta + t_i) \in \mathcal{G}(e_i)$, and $\gamma_{l_{i-1}}(\bar{x}, \delta') \in Inv(l_{i-1})$ for all δ' such that $\delta \leq \delta' \leq \delta + t_i$. In the automaton $\mathcal{A}_\mathcal{H}$ we have reached a state q_{i-1} via a path labeled $(a_1, P_1) \dots (a_{i-1}, P_{i-1})$ where $t_k \in P_k$ for $1 \leq k < i$. Then by definition there exists an edge from q_{i-1} to a state q_i labeled with (a_i, P_i) where the target state of this edge was derived from the location l_i in $\mathcal{H}$.

In the particular case of (a_n, t_n), notice that the last transition labeled by a_n ends in the state (l_n, y_n) and that l_n is terminal. Thus in the automaton $\mathcal{A}_\mathcal{H}$ the corresponding state q_n is terminal.

We have shown that if $w = (a_1, t_1) \dots (a_n, t_n)$ is a timed word accepted by $\mathcal{H}$, then the word $(a_1, P_1) \dots (a_n, P_n)$ over the alphabet $\Sigma \times \mathcal{P}$ is accepted by $\mathcal{A}_\mathcal{H}$. This means that w belongs to $L(E)$, the timed language represented by the hybrid regular expression E.

The converse is proved in a similar way. Let $w = (a_1, t_1) \dots (a_n, t_n)$ be a timed word appearing in $L(E)$. Then there exists partition pieces $P_1, \dots, P_n$ (not necessarily distinct) such that $t_i \in P_i$ for $i = 1, \dots, n$, and such that the word $(a_1, P_1) \dots (a_n, P_n)$ is accepted by the finite automaton $\mathcal{A}_\mathcal{H}$. By the same kind of argument as above, we prove that the word w is accepted by $\mathcal{H}$.

Finally, if the language accepted by the original hybrid system $\mathcal{H}$ includes the timed empty word, we add this information to the hybrid regular expression E as $\emptyset^*$. It is clear that $E \cup \emptyset^*$ is still a hybrid regular expression. We also add to the automaton $\mathcal{A}_\mathcal{H}$ a new state which is both initial and terminal. □

Remark 3.7. The success of this method is due to the strong reset condition imposed by the definition of a hybrid system. Without this condition, the subsets defined in step (c) will not be defined correctly for every edge. Note also that the definable subsets which form the partition pieces of M^+ may have arbitrarily many connected components. If the underlying structure is o-minimal, the partition pieces are simple intervals.

No assumption is made about the decidability of the underlying theory $\mathcal{M}$. In the case of an undecidable theory, the preceding proof is not constructive. Nevertheless, when the theory is decidable, the proof becomes constructive.

Lemma 3.8. *If the underlying theory $\mathcal{M}$ is decidable, then the abstract finite automaton $\mathcal{A}_\mathcal{H}$ defined from the hybrid system $\mathcal{H}$ in Proposition 3.3 can be effectively constructed.*

Proof. Let $\mathcal{A}_\mathcal{H}$ be the abstract finite automaton defined in the proof of Proposition 3.3. By looking at this proof, we show that this automaton can be constructed. In step (b), it is decidable whether two resets (resp. two guards) are equal since they are definable sets. Similarly, it is also decidable whether or not the sets Init_l, Term_l, and $\mathcal{R}_i \cap \mathsf{Term}_l$ are empty. In step(c), one may decide whether $T(l)$ or $T_I(l)$ is empty, and one can construct the definable pieces of the partition $\mathcal{P}$ that are non-empty (by giving the formulae defining them). Finally each $T(l)$ and $T_I(l)$ can be written as a finite union of partition pieces since the inclusion of two definable sets is decidable. □

We now proceed to the proof of Proposition 3.4. An illustration of this proposition is given in Example 4.3 below.

Proof of Proposition 3.4. Let E be a hybrid regular expression, constructed using a finite alphabet of events Σ and a finite partition $\mathcal{P}$ of M^+ into definable subsets. Then by Proposition 3.3 there exists a finite abstract automaton $\mathcal{A}$ with edges labeled from $\Sigma \times \mathcal{P}$ which accepts $L_A(E)$. This automaton can be transformed into a hybrid system $\mathcal{H}$ with resets, invariants and guards strongly related to the partition pieces of $\mathcal{P}$. We construct the hybrid system from $\mathcal{A}$ in such a way as to accept exactly the hybrid regular language $L(E)$ represented by E. The key idea is to use only one variable x for the output space V_2 which behaves like a clock (the dynamics in each location are the same as for timed automata). For a given edge in the automaton $\mathcal{A}$ with label (a, P) for some $a \in \Sigma$ and $P \in \mathcal{P}$, we relabel the edge as follows. The event label will remain unchanged. For the guard we have $\mathcal{G} = P$ and the reset $x := 0$.

The termination condition Term of the new system $\mathcal{H}$ is equal to $\bigcup_l (l, \mathsf{Term}_l)$ with $\mathsf{Term}_l = \{0\}$ when l is a terminal state of $\mathcal{A}$ and $\mathsf{Term}_l = \emptyset$ when l is a non-terminal state of $\mathcal{A}$. Similarly the initial condition Init is equal to $(l_0, \{0\}) \cup \bigcup_{l \neq l_0} (l, \emptyset)$ where l_0 is the initial state of $\mathcal{A}$.

The last thing to consider is the invariants. Consider a location l in our system so far, and suppose it has outgoing edges with guards which are partition pieces $P_1, \ldots, P_n$ for some $n \in \mathbb{N}$. Then the invariant $Inv(l)$ of l will be the set of all $x \in M^+$ bounded by the supremum value of the partition pieces $P_1, \ldots, P_n$. Indeed, an edge with label (a, P) in the automaton $\mathcal{A}$ has an equivalent edge in the hybrid system $\mathcal{H}$ which can only be taken if the clock has a value within the partition piece P (by definition of the resets and the initial condition). The invariants specified above ensure that the system cannot simply stay at one location; it is impelled to take one of the available discrete transitions available. Notice that each invariant $Inv(l)$ is a definable set. Indeed, the partition pieces $P_1, \ldots, P_n$ are definable by $\varphi_1(x), \ldots, \varphi_n(x)$ respectively, and therefore $Inv(l)$ is definable by the formula $\phi(x) : x \geq 0 \wedge \exists y \; x \leq y \wedge (\varphi_1(y) \vee \ldots \vee \varphi_n(y))$.

It is easy to verify that the hybrid system $\mathcal{H}$ defined in this way accepts exactly the hybrid regular language $L(E)$ represented by the expression E. □

Remark 3.9. Contrarily to Proposition 3.3, one can symbolically construct the system $\mathcal{H}$ in the previous proof even if the underlying theory in undecidable. However, in this general framework, we can not check if a given label is empty or not; in particular it would not be possible to check emptiness.

Propositions 3.3 and 3.4 give a number of interesting immediate corollaries, including a pumping type lemma for hybrid regular languages. The first corollary is a consequence of the proof of Proposition 3.4. The second corollary is a consequence of the (classical) pumping lemma applied to the automaton $\mathcal{A}_{\mathcal{H}}$ defined in Proposition 3.3. We use the definition of the pumping lemma property for timed words suggested in [9].

Corollary 3.10. *Let $\mathcal{M} = \langle M, <, +, 0, \ldots \rangle$ be an expansion of an ordered group, and let $\mathcal{H}$ be a hybrid system over $\mathcal{M}$. Then $\mathcal{H}$ is equivalent in accepting power to a hybrid system with a 1-dimensional output space.*

Corollary 3.11. *Let L be a hybrid regular language. Then there exists a constant $K > 0$ such that for all timed words $u \in L$ with length (respectively, duration) of u greater than K there exists timed words v, w, z with $w \neq (\epsilon, 0)$ which satisfy*

$$u = vwz \text{ and for every integer } n \geq 0 \; vw^n z \in L.$$

We conclude this section with comparisons with some related works.

The notion of regular expression exists for timed automata [8]. The key difference is the need for extra operations in order to entirely define the language class. It is necessary to include intersection as well as an interval operator, which allows the overall duration of some subexpression to be specified. The latter operation demonstrates the obvious difference between our definition of hybrid systems and timed automata; the interval operator is a way of encoding into a regular expression the memory in the resets of the automaton. Since we insist on all discrete transitions being memoryless, this operator is not compatible with our definition.

In [48] the authors define a notion of hybrid regular expression slightly different from our notion. It involves an extra operation which uses linear inequalities. This operation is used to encode information about the resets, guards, invariants and dynamics of the system. Their definition of hybrid regular expression works for a class of linear hybrid systems (we recall that linear hybrid systems have linear guards, resets and invariants, and no strong reset condition).

In [26], the author studies the class of timed automata with a single clock which is reset at each transition (epsilon-transitions are allowed). Among other results, he proposes a notion of regular expression and obtains a Kleene theorem. These results are close to some of our results in the following sense. The timed automata studied in [26] are hybrid systems as the one defined in Proposition 3.4 except that the underlying structure is limited to the o-minimal decidable structure $\langle \mathbb{R}, <, +, \cdot, 0, 1 \rangle$. However epsilon-transitions are allowed contrarily to our model.

The regular expressions proposed in [26] are very close to ours. They are defined over a finite alphabet $\Sigma \times \mathcal{P}$ such that $\mathcal{P}$ is a partition of $\mathbb{R}^+$ into definable subsets of $\langle \mathbb{R}, <, +, \cdot, 0, 1 \rangle$, that is, into intervals with rational bounds. More details concerning the comparison of our results with those of [26] are given at the end of Section 4 where we discuss the power of epsilon-transitions in our model.

We conclude this section with a last comment concerning infinite words. Notice that an analogue of Theorem 3.5 could be obtained for hybrid systems with Büchi acceptance condition and hybrid omega regular expressions.

4. Closure properties of hybrid regular languages

In this section we consider the most natural questions related to determinism, complementation, emptiness and universality for hybrid systems where $\mathcal{M}$ is an expansion of an ordered group.

First we require a compatible definition of determinism. The following definition was originally given for timed automata [4], but is naturally extended to the hybrid systems case.

Definition 4.1. We say that a hybrid system is *deterministic* if for all locations $l \in Loc$, and for every pair of outgoing edges e_1 and e_2 from l with the same label a from Σ, the guards $\mathcal{G}(e_1)$ and $\mathcal{G}(e_2)$ are mutually exclusive.

In the case of timed automata [4], the class of deterministic timed languages is strictly included in the class of non-deterministic timed languages, and the non-deterministic class is not closed under complement. However, our characterization of hybrid regular languages using abstract finite automata allows us to conclude the following.

Theorem 4.2. *Let $\mathcal{H}$ be a non-deterministic hybrid system accepting a timed language L. Then there exists a deterministic hybrid system accepting L.*

Proof. Given a hybrid system $\mathcal{H}$ accepting a timed language $L_{\mathcal{H}}$, by Proposition 3.3 above we may define a finite alphabet $\Sigma \times \mathcal{P}$, a hybrid regular expression E over $\Sigma \times \mathcal{P}$ such that $L_{\mathcal{H}} = L(E)$ and a finite abstract automaton $\mathcal{A}_{\mathcal{H}}$ which accepts $L_A(E)$. Then there exists a deterministic finite automaton $\mathcal{A}'$ over $\Sigma \times \mathcal{P}$ accepting the same language $L_A(E)$ as $\mathcal{A}_{\mathcal{H}}$. Looking at the proof of Proposition 3.4, we easily check that the hybrid system $\mathcal{H}'$ resulting from $\mathcal{A}'$ remains deterministic. Hence we obtain in this way a deterministic hybrid system accepting exactly $L(E) = L_{\mathcal{H}}$. □

Example 4.3. We recall Example 3.6. Note that this example is non-deterministic. When determinizing the finite automaton in Figure 6, we obtain the automaton in Figure 11.

Following Proposition 3.4, we then build the deterministic hybrid system shown in Figure 12. For clarity the invariants have been left off the figure: all locations share the invariant $Inv(l_i) = [0, 4]$ for $i = 0, 1, 2$. The initial condition Init is

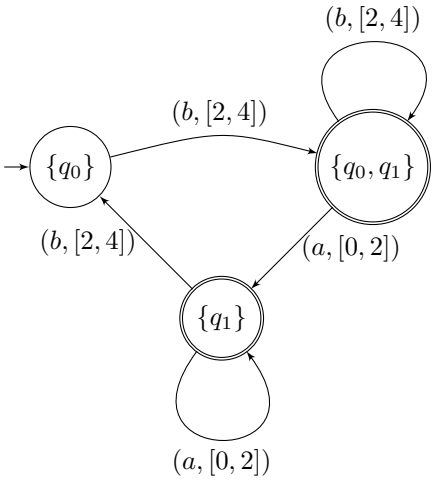

FIGURE 11. A deterministic finite automaton for the expression (1).

equal to $(l_0, \{0\}) \cup (l_1, \emptyset) \cup (l_2, \emptyset)$ and the termination condition Term is equal to $(l_0, \emptyset\}) \cup (l_1, \{0\}) \cup (l_2, \{0\})$.

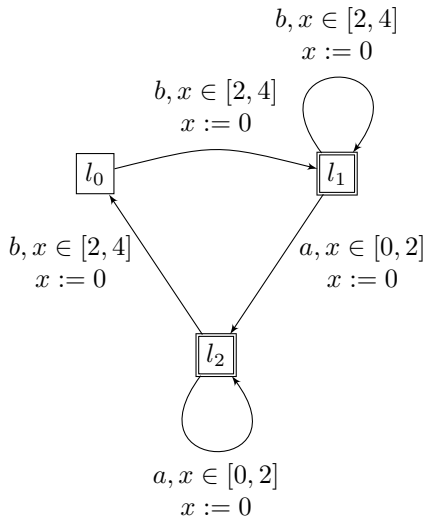

FIGURE 12. A deterministic hybrid system for the expression (1).

With the same ideas as for Theorem 4.2, we prove the closure under complement of the class of hybrid regular languages. Since this class is closed under union, it is also closed under intersection.

Theorem 4.4. *The family of hybrid regular languages is closed under boolean operations.*

Proof. We only need to prove the closure under complement. Let $\mathcal{H}$ be a hybrid system accepting a timed language $L_{\mathcal{H}}$. By Proposition 3.3 there exists a hybrid regular expression E over $\Sigma \times \mathcal{P}$ such that $L_{\mathcal{H}} = L(E)$ and a finite abstract automaton $\mathcal{A}_{\mathcal{H}}$ accepting $L_A(E)$. We consider $\mathcal{A}'$, the finite automaton accepting the complement L' of the language $L_A(E)$ over the alphabet $\Sigma \times \mathcal{P}$, and E' the related (classical) regular expression over this alphabet. This expression seen as a hybrid regular expression represents a timed language $L(E')$ which is the complement of $L(E)$. Indeed, $\mathcal{P}$ is a partition of M^+. So by applying Proposition 3.4 to the automaton $\mathcal{A}'$, we may define a hybrid system $\mathcal{H}'$ accepting exactly $L(E')$ and the theorem is proved. □

Remark 4.5. Notice that the previous proof is not constructive. Nevertheless, the hybrid system $\mathcal{H}'$ can be effectively constructed from the given hybrid system $\mathcal{H}$ as soon as the underlying theory is decidable. So in this case the family of hybrid regular languages is effectively closed under boolean operations.

It is well known [4] that in the case of timed automata, the deterministic timed languages are strictly included in the non-deterministic timed languages, and the language family is not closed under complement. Consider the timed automaton in Figure 13. This is the classical example of a timed automaton which accepts a timed language whose complement is not accepted by a timed automaton [32]. It is clear that this timed automaton is not a hybrid system as defined in this article, since the clock variable x is not reset on each edge. Indeed, x cannot be reset on the loop of location 2, though on all other edges it would be reasonable to reset the clock to zero. Hence our method cannot be applied in the more general setting of timed automata.

By Corollary 3.10, the hybrid systems which we have studied are equivalent in accepting power to hybrid systems using only one variable for the output space. In comparison with the timed automaton of Figure 13, such hybrid systems have strong reset conditions on the unique variable but in counterpart they have powerful dynamics on the locations. We have shown that the family of languages accepted by such systems are closed under complementation.

We now turn to the problems of emptiness and universality for hybrid systems. From this point forward in the paper it is necessary to suppose decidability of the underlying theory. It is already known (see [18]) that the emptiness problem is decidable for o-minimal hybrid systems as soon as the underlying o-minimal structure is decidable. This result is often mentioned as the reachability problem. The following theorem applies in a more general setting.

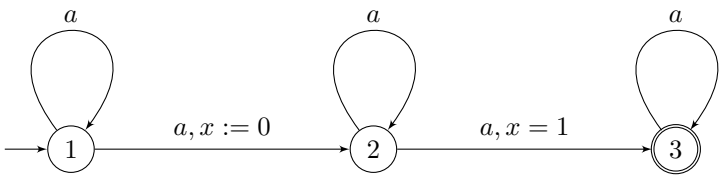

FIGURE 13. A timed automaton whose complement language is not accepted by a timed automaton.

Theorem 4.6. *Let $\mathcal{M} = \langle M, <, +, 0, \ldots \rangle$ be a decidable expansion of an ordered group. Then the emptiness problem for hybrid systems over $\mathcal{M}$ is decidable.*

Proof. Let $\mathcal{H}$ be a hybrid system accepting a timed language $L_{\mathcal{H}}$. By Proposition 3.3, we may consider a finite alphabet $\Sigma \times \mathcal{P}$ with Σ the event alphabet of $\mathcal{H}$ and $\mathcal{P}$ a finite partition of M^+ info definable non-empty subsets, and an abstract finite automaton $\mathcal{A}_{\mathcal{H}}$ accepting a language $L = L_A(E)$ for a hybrid regular expression E over $\Sigma \times \mathcal{P}$ such that $L_{\mathcal{H}} = L(E)$.

By Lemma 3.8, the automaton $\mathcal{A}_{\mathcal{H}}$ can be effectively constructed. Therefore, it is decidable whether $L_{\mathcal{H}}$ is non-empty since it is equivalent to check if there exists a word $(a_1, P_1)(a_2, P_2) \ldots (a_n, P_n)$ over $\Sigma \times \mathcal{P}$ that is accepted by $\mathcal{A}_{\mathcal{H}}$. □

From Theorems 4.6 and Remark 4.5, we have the next corollary.

Corollary 4.7. *The universality problem is decidable for hybrid systems. The language inclusion problem is decidable for hybrid systems.*

Returning to our comparison with timed automata, we note that in [4] the authors show that the universality problem and the language inclusion problem are undecidable for timed automata. On the positive side, the emptiness problem (also called the reachability problem) is decidable for timed automata.

The picture is different for hybrid systems with a decidable underlying structure since all these problems have been shown to be decidable. We recall that these systems allow rich dynamics inside their locations and a strong reset condition on the edges.

On the other hand, it should be noted that if the definition of timed automata is slightly broadened to allow irrational constraints, the reachability problem becomes undecidable [40].

We end this section with some comments. The results of this section have a strong relationship with the results proved in [26]. We have proved in the previous section that hybrid systems are equivalent in accepting power to hybrid systems using only one variable that behaves like a clock (Corollary 3.10). Timed automata with a single clock which is reset at each transition are studied in [26] with a view to answering the same questions as in this section. In the latter article, the conclusions are similar, with two main differences:

- the underlying structure is restricted to the o-minimal decidable structure $\langle \mathbb{R}, <, +, \cdot, 0, 1 \rangle$,

- in contrast, epsilon transitions resetting the clock are allowed in the considered model.

If we allow-epsilon transitions with the strong reset condition in our model of a hybrid system, power is added to the model as demonstrated by the following example. Notice that o-minimal hybrid systems with epsilon-transitions *without* the strong reset condition have an undecidable emptiness problem [17, 18].

Example 4.8. In Figure 14, we consider the hybrid system $\mathcal{H}$ with a loop labeled by the silent event ϵ. The invariants of the locations l and l' are $Inv(l) = [0, 1]$ and $Inv(l') = \{0\}$ respectively. The initial condition is $(l, \{0\}) \cup (l', \emptyset)$ and the termination condition is $(l, \emptyset) \cup (l', \{0\})$. The underlying structure is $\langle \mathbb{R}, <, +, 0, 1 \rangle$. The timed language $L_{\mathcal{H}}$ accepted by $\mathcal{H}$ is equal to

$$\{(a, n) \mid n \in \mathbb{N}\}.$$

There is no hybrid system $\mathcal{H}'$ without ϵ-transitions (definable in $\langle \mathbb{R}, <, +, 0, 1 \rangle$) such that $L_{\mathcal{H}'} = L_{\mathcal{H}}$. Otherwise by Proposition 3.3, there is a finite partition $\mathcal{P}$ into definable subsets of $\mathbb{R}^+$ and a hybrid regular expression E over $\Sigma \times \mathcal{P}$ such that $L(E) = L_{\mathcal{H}}$. By the o-minimality of the structure, the definable pieces of the finite partition are intervals with rational bounds. Hence it is impossible to have such an expression E since the structure of $L_{\mathcal{H}}$ forces an infinite number of intervals.

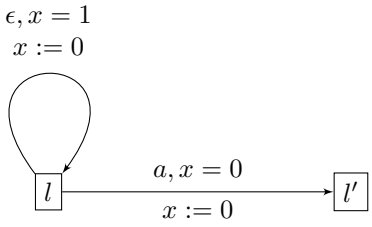

FIGURE 14. An hybrid system where the ϵ transition can not be eliminated.

Notice that if we shift from the structure $\langle \mathbb{R}, <, +, 0, 1 \rangle$ to the structure $\langle \mathbb{R}, \mathbb{Z}, <, +, 0, 1 \rangle$, the timed language $L_{\mathcal{H}}$ is then accepted by the hybrid system $\mathcal{H}'$ with a single location l such that $Inv(l) = \mathbb{R}^+$, $\mathsf{Init}_l = \mathsf{Term}_l = \{0\}$, and with a single edge $e = (l, a, l)$ labeled by $a \in \Sigma$ such that $\mathcal{G}(e) = \mathbb{N}$ and $\mathcal{R}(e) = \{0\}$. The guard $\mathcal{G}(e)$ is a definable subset of $\langle \mathbb{R}, \mathbb{Z}, <, +, 0, 1 \rangle$ (but not of the original structure).

The same kind of example is given in [26] to show that the studied model is more powerful when ϵ-transitions are allowed. In this general setting, the author is able to prove the closure under complementation. Indeed he shows that any timed automaton with ϵ-transitions and a single clock that is reset at each transition is equivalent in accepting power to such an automaton without ϵ-transition but with more general constraints: instead of forcing the clock to belong to an interval

with rational bounds, it is forced to belong to the sub-Kleene algebra generated by these intervals. In terms of our hybrid systems, this means that the underlying structure $\langle \mathbb{R}, <, +, 0, 1\rangle$ is extended with the predicate $\mathbb{Z}$. This fact has already been demonstrated in the previous example. Notice that the theory of the extended structure $\langle \mathbb{R}, \mathbb{Z}, <, +, 0, 1\rangle$ has been proved decidable in [46]. This approach could provide an alternative proof to [26, Theorem 4.5] (decidability of the universality).

5. SYNCHRONIZED PRODUCTS OF O-MINIMAL HYBRID SYSTEMS

In this section we define the notion of a synchronized product of hybrid systems. Synchronized product is an operation allowing two (or more) hybrid systems to synchronize on some discrete transitions. This operation is very useful in the verification framework because it allows the description of a large system as the composition of several communicating smaller systems that can be described more easily.

Informally, given two hybrid systems $\mathcal{H}_1$ and $\mathcal{H}_2$, the *synchronized product* means that the systems interact on joint events. So if an event a is an event of both $\mathcal{H}_1$ and $\mathcal{H}_2$, then the two systems synchronize on discrete transitions labeled by a. If a is an event of $\mathcal{H}_1$ only, then a discrete transition of $\mathcal{H}_1$ labeled by a synchronizes with a continuous transition of duration 0 in $\mathcal{H}_2$, and vice versa.

Definition 5.1. Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two hybrid systems with event alphabets Σ_1 and Σ_2 respectively. The synchronized product of $\mathcal{H}_1$ and $\mathcal{H}_2$ is the transition system $\mathcal{T}^{can}_{\mathcal{H}_1\times\mathcal{H}_2} = (Q, \to)$ defined as follows from the canonical transition systems $\mathcal{T}^{can}_{\mathcal{H}_1} = (Q_1, \to_1)$ and $\mathcal{T}^{can}_{\mathcal{H}_2} = (Q_2, \to_1)$.

- The set of states Q is $Q_1 \times Q_2$,
- The event alphabet Σ is $\Sigma_1 \cup \Sigma_2$,
- A discrete transition $(q_1, q_2) \xrightarrow{a} (q_1', q_2')$ labeled by the event $a \in \Sigma$ satisfies one among the following conditions.
  - $a \in \Sigma_1 \cap \Sigma_2$. Then $q_1 \xrightarrow{a} q_1'$ and $q_2 \xrightarrow{a} q_2'$ are discrete transitions in $\mathcal{T}^{can}_{\mathcal{H}_1}$ and $\mathcal{T}^{can}_{\mathcal{H}_2}$ respectively.
  - $a \in \Sigma_1 \setminus \Sigma_2$. Then $q_1 \xrightarrow{a} q_1'$ and $q_2 = q_2'$.
  - $a \in \Sigma_2 \setminus \Sigma_1$. Then $q_1 = q_1'$ and $q_2 \xrightarrow{a} q_2'$.
- For a continuous transition $(q_1, q_2) \xrightarrow{t} (q_1', q_2')$ with duration $t \in M^+$ to appear, $q_1 \xrightarrow{t} q_1'$ and $q_2 \xrightarrow{t} q_2'$ are continuous transitions of $\mathcal{T}^{can}_{\mathcal{H}_1}$ and $\mathcal{T}^{can}_{\mathcal{H}_2}$ respectively.
- A transition $(q_1, q_2) \to (q_1', q_2')$ appears in $\mathcal{T}^{can}_{\mathcal{H}_1\times\mathcal{H}_2}$ if there exists $\bar{q}_1 \in Q_1$, $\bar{q}_2 \in Q_2$ such that

$$(q_1, q_2) \xrightarrow{t} (\bar{q}_1, \bar{q}_2) \xrightarrow{a} (q_1', q_2'),$$

that is, there exists a continuous transition followed by a discrete transition (as specified above).

The *synchronization alphabet* of $\mathcal{H}_1$ and $\mathcal{H}_2$ is equal to $\Sigma_1 \cap \Sigma_2$, that is, the letters on which they synchronize.

We extend the definition to a product of n hybrid systems in the obvious way.

We prove that the reachability problem is in general undecidable for synchronized products of hybrid systems. In the proof we construct a simulation of a two counter machine using a synchronized product of seven hybrid systems that are o-minimal. The construction is inspired by the proof given in [40] of the undecidability of reachability in the case of timed automata with irrational constants.

Theorem 5.2. *The reachability problem is undecidable for synchronized products of hybrid systems.*

Proof.
A two counter machine. Consider a two counter machine consisting of counters C_1 and C_2 and a finite list of labeled instructions given in Table 1. Without loss of generality, we assume that the initial instruction is labeled with 1.

zero test	k : if $C_i = 0$ then go to k' else go to k''
increment	k : $C_i := C_i + 1$
decrement	k : $C_i := C_i - 1$
stop	k : STOP

TABLE 1. The possible instructions of a two-counter machine.

Our goal is to simulate the behavior of a two counter machine by using a synchronized product of hybrid systems. The underlying o-minimal decidable structure $\mathcal{M}$ is the ordered field of reals $\langle \mathbb{R}, <, +, \cdot, 0, 1 \rangle$. We require seven separate systems to achieve this aim; one system will be used to encode the instruction list for the machine, while the others will encode the evolution of the counters themselves.

Encoding the instruction list. We encode such a finite list as a hybrid system $\mathcal{H}_0$ with no variable. Note that no information about the state of the counters will be included in this system, it will simply serve as a reference for which action should be performed when. The locations are the labels k, plus some additional locations as described below. The initial condition Init is such that $\mathsf{Init}_l = \emptyset$ for each location l except for the one with label $k = 1$ where $\mathsf{Init}_l = \{0\}$. The set Σ of events consists of the actions of the counter machine, that is $\mathrm{inc}_i, \mathrm{dec}_i, \mathrm{zero}_i, \text{non-zero}_i, \mathrm{copy}_i, \mathrm{stop}$. The edges of $\mathcal{H}_0$ are depicted in Figure 15.

An extra step is added to the negative zero test for technical reasons which we shall explain later. We add a new location $\bar{k}$ for this purpose. Similarly we add a new location K which indicates the halting of the counter machine.

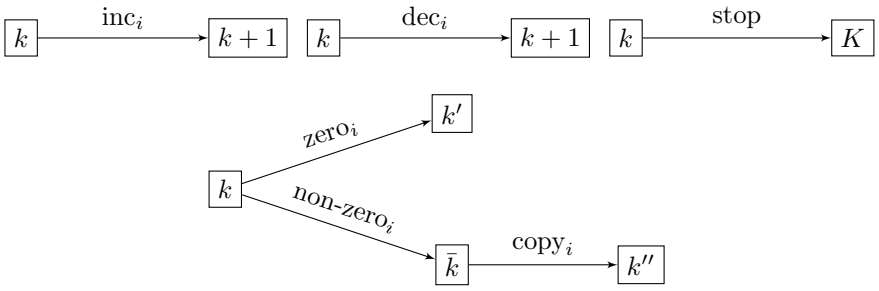

FIGURE 15. The edges of the hybrid system $\mathcal{H}_0$

Encoding the counters. Let $\delta \in [1,2] \setminus \mathbb{Q}$ be definable; we fix $\delta = \sqrt{2}$. Then we encode the integer value m_i of the counter C_i by $m_i\delta \pmod 1$, which we will denote by $\langle m_i\delta \rangle$. Since $\delta \notin \mathbb{Q}$, $\langle m_i\delta \rangle$ unambiguously encodes the value m_i (that is, if $\langle m_i\delta \rangle = \langle n_i\delta \rangle$ then $m_i = n_i$).

For each counter C_i we have three variables x_i, y_i and z_i, with $0 \leq x_i \leq \delta$, and $0 \leq y_i, z_i \leq 1$. We encode the value of the counter C_i as the value $y_i - x_i$ in the system, plus z_i a copy of y_i, which we require for technical reasons which we discuss later. See Figure 16.

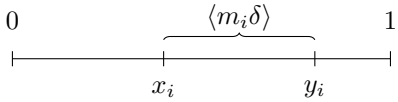

FIGURE 16. $y_i - x_i$ represents the counter value $\pmod 1$.

The encoding of the counter value as the difference of the two variables x_i and y_i must be understood to be circular, as seen in Figure 17. An alternative but equivalent understanding of the encoding of the counter value may be gained by considering the counter to be the value of y_i when the variable x_i is equal to one and we are in the distinguished location defined in the next section.

FIGURE 17. $\alpha + \beta = \langle m_i\delta \rangle$

The structure of the hybrid systems. In the previous paragraph we described the system $\mathcal{H}_0$. The other six systems are related to the six variables x_i, y_i, z_i for $i = 1, 2$. We denote them by $\mathcal{H}_i^x, \mathcal{H}_i^y, \mathcal{H}_i^z$ for $i = 1, 2$ respectively. These six hybrid systems each have only the one variable implied by their names (namely x_i, y_i and z_i respectively). We distinguish one particular location in the system $\mathcal{H}_i^x$ and denote this location l_i^x. For each of these systems, its alphabet of events is composed of some events of the system $\mathcal{H}_0$ and an additional event only used by the system. See for instance Figure 18, the additional event letter has not been indicated on the figure but is implicitly present on the transition linking the location DEC to the location l_i^x. Note that except for two edges in the system $\mathcal{H}_i^x$ the variables x_i, y_i, z_i are reset to zero each time their value reaches one.

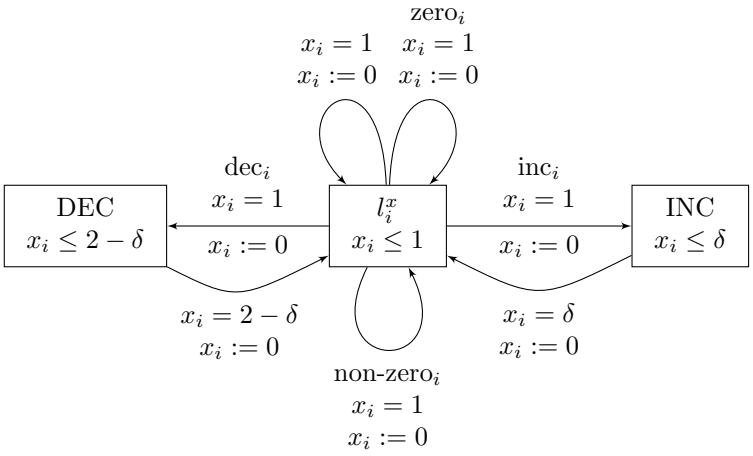

FIGURE 18. The system $\mathcal{H}_i^x$ with $\delta = \sqrt{2}$.

- Incrementing counter C_i.
  The increment step is performed using the systems $\mathcal{H}_0$ and $\mathcal{H}_i^x$. Suppose that counter C_i is encoded by $\langle m_i\delta\rangle$, that is, $\langle m_i\delta\rangle = y_i - x_i$ with $x_i = 1$ and we are in the distinguished location l_i^x of $\mathcal{H}_i^x$. On reaching an edge labeled inc$_i$ in the system $\mathcal{H}_0$ we are compelled to follow the edge from l_i^x in $\mathcal{H}_i^x$ to the location labeled INC (note x_i has been reset to 0). On returning to l_i^x we again have $x_i = 0$. We claim that the counter C_i has been incremented by 1, that is $y_i - x_i = \langle (m_i + 1)\delta\rangle$. See Figure 21 (we begin with x_i being reset to zero).
- Decrementing counter C_i.
  The decrement step is performed using the systems $\mathcal{H}_0$ and $\mathcal{H}_i^x$. On reaching an edge labeled dec$_i$ in the system $\mathcal{H}_0$ we are compelled to follow the edge from l_i^x to the location labeled DEC. On returning to l_i^x we again

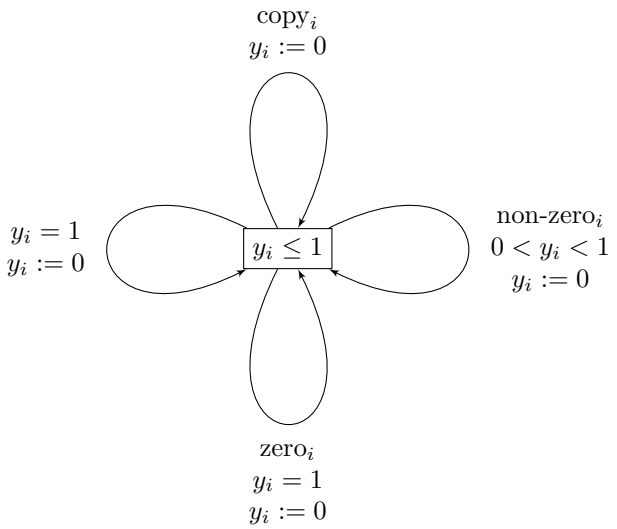

FIGURE 19. The system $\mathcal{H}_i^y$

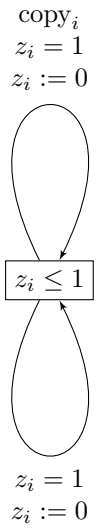

FIGURE 20. The system $\mathcal{H}_i^z$

have $x_i = 0$. We claim that the counter C_i has been decremented by 1, that is $y_i = x_i = \langle (m_i - 1)\delta \rangle$. See Figure 22.

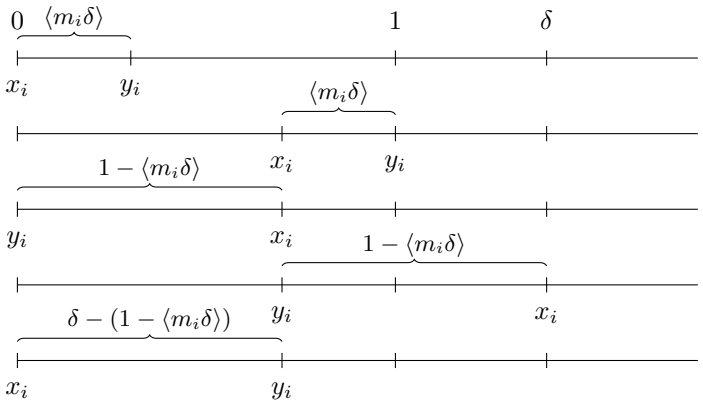

FIGURE 21. The increment step. Note that $\delta - (1 - \langle m_i\delta\rangle) = \langle (m_i + 1)\delta\rangle$.

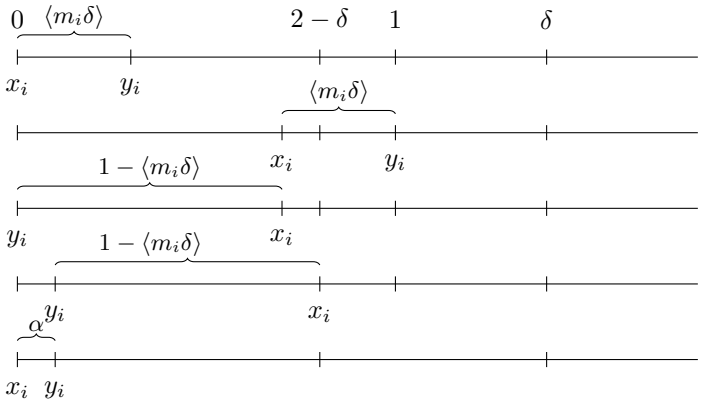

FIGURE 22. The decrement step. Note that $\alpha = 2 - \delta - (1 - \langle m_i\delta\rangle)$.

- Positive zero test on C_i.

  For this operation on the counter C_i we require the hybrid systems $\mathcal{H}_0, \mathcal{H}_i^x$ and $\mathcal{H}_i^y$. From our encoding of the value of C_i, it is clear that the value of C_i is zero if and only if $\langle m_i\delta\rangle = 0$ which is true if and only if $y_i = x_i$. In the distinguished location l_i^x of $\mathcal{H}_i^x$ with $x_i = 1$, this means that $y_i = 1$ and we are compelled to follow the edge labeled zero$_i$ in both $\mathcal{H}_i^x$ and $\mathcal{H}_i^y$ as required.
- Negative zero test on C_i.

  We require the hybrid systems $\mathcal{H}_0, \mathcal{H}_i^x, \mathcal{H}_i^y$ and $\mathcal{H}_i^z$. Note that in $\mathcal{H}_0$, each edge labeled with non-zero$_i$ is followed immediately by an edge labeled with copy$_i$. Therefore there is a synchronization between $\mathcal{H}_0, \mathcal{H}_i^x$ and

$\mathcal{H}_i^y$ (due to the event non-zero$_i$) then a synchronization between $\mathcal{H}_0, \mathcal{H}_i^y$ and $\mathcal{H}_i^z$ (due to the event copy$_i$). If the value of C_i is non-zero, our encoding implies that $x_i \neq y_i$. Equivalently, $x_i = 1$ and $0 < y_i < 1$. If the value of C_i is non zero then we synchronize the systems $\mathcal{H}_i^x$ and $\mathcal{H}_i^y$ on the event non-zero$_i$ (see Figures 18 and 19). The strong reset condition means that the variables x_i and y_i are reset to zero, and the encoded value of the counter is lost. However, recall that the variable z_i is a copy of the variable y_i, and the former will not have been reset since there is no edge labeled non-zero$_i$ in the system $\mathcal{H}_i^z$. So, on encountering next an edge labeled copy$_i$ in the system $\mathcal{H}_0$, we resynchronize the value of y_i to the stored value z_i, ensuring that no information is lost. See Figure 23.

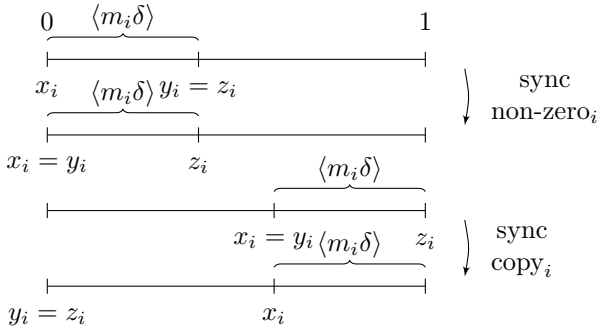

FIGURE 23. Negative zero test.

From the above discussion, one can conclude that the two counter machine halts if and only if the synchronized product of the seven hybrid systems reaches the `stop` location.

□

Consider hybrid systems such that their underlying structure is o-minimal and decidable. We can now detail the picture about the decidability of the reachability problem for the synchronized product of such systems.

In [22], the authors show that the reachability problem is decidable under the hypothesis that the synchronization alphabet is empty. In our proof, we use hybrid systems such that the synchronization alphabet is non-empty and the reachability problem becomes undecidable.

Note that when two hybrid systems $\mathcal{H}_1$ and $\mathcal{H}_2$ have the same alphabet of events which is their synchronization alphabet, their synchronized product is nothing else than a hybrid system accepting the intersection of the timed languages $L_{\mathcal{H}_1}$ and $L_{\mathcal{H}_2}$. In the proof of Theorem 5.2, the synchronization alphabet is strictly included in the event alphabet of the systems, and the resulting synchronized product is no longer a hybrid system with the strong reset condition. This ensures no contradiction with Theorem 4.4.

Therefore the borderline between decidability and undecidability of the reachability problem strongly depends on the synchronization alphabet. It is in general undecidable (by Theorem 4.4) and becomes decidable if the synchronization alphabet is empty or if it is equal to the event alphabet of each system.

References

[1] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

[2] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[3] Rajeev Alur and David Dill. Automata for modeling real-time systems. In *ICALP'90: Automata, Languages, and Programming*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer, 1990.

[4] Rajeev Alur and David Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[5] Rajeev Alur, Limor Fix, and Thomas A. Henzinger. Event-clock automata: a determinizable class of timed automata. *Theoretical Computer Science*, 211:253–273, 1999.

[6] Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. In *HSCC'01: Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 49–62. Springer, 2001.

[7] Rajeev Alur and Parthasarathy Madhusudan. Decision problems for timed automata: A survey. In *SFM'04: School on Formal Methods*, volume 3185 of *Lecture Notes in Computer Science*, pages 1–24. Springer, 2004.

[8] Eugene Asarin, Paul Caspi, and Oded Maler. A kleene theorem for timed automata. In *LICS*, pages 160–171, 1997.

[9] Danièle Beauquier. Pumping lemmas for timed automata. In *Foundations of software science and computation structures (Lisbon, 1998)*, volume 1378 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 1998.

[10] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim G. Larsen, Paul Pettersson, Judi Romijn, and Frits Vaandrager. Minimum-cost reachability for priced timed automata. In *HSCC'01: Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2001.

[11] Béatrice Bérard, Antoine Petit, Volker Diekert, and Paul Gastin. Characterization of the expressive power of silent transitions in timed automata. *Fund. Inform.*, 36(2-3):145–182, 1998.

[12] Patricia Bouyer, Thomas Brihaye, and Fabrice Chevalier. Control in o-minimal hybrid systems. In *LICS'06: Logic in Computer Science*, pages 367–378. IEEE Computer Society Press, 2006.

[13] Patricia Bouyer, Thomas Brihaye, and Fabrice Chevalier. Weighted o-minimal hybrid systems are more decidable than weighted timed automata! In *LFCS'07: Logical Foundations of Computer Science*, volume 4514 of *LNCS*, pages 69–83. Springer, 2007.

[14] Patricia Bouyer, Thomas Brihaye, Marcin Jurdziński, Ranko Lazić, and Michał Rutkowski. Average-price and reachability-price games on hybrid automata with strong resets. In *FORMATS'08: Formal Modelling and Analysis of Timed Systems*, volume 5215 of *Lecture Notes in Computer Science*. Springer, 2008. To appear.

[15] Patricia Bouyer, Serge Haddad, and Pierre-Alain Reynier. Undecidability results for timed automata with silent transitions. Research Report LSV-07-12, Laboratoire Spécification et Vérification, ENS Cachan, France, February 2007. 22 pages.

[16] Patricia Bouyer and Antoine Petit. A Kleene/büchi-like theorem for clock languages. *Journal of Automata, Languages and Combinatorics*, 7(2):167–186, 2002.

[17] Thomas Brihaye. A note on the undecidability of the reachability problem for o-minimal dynamical systems. *MLQ. Mathematical Logic Quarterly*, 52(2):165–170, 2006.
[18] Thomas Brihaye. *Verification and Control of O-Minimal Hybrid Systems and Weighted Timed Automata*. Thèse de doctorat, Université Mons-Hainaut, Belgium, 2006.
[19] Thomas Brihaye and Christian Michaux. On the expressiveness and decidability of o-minimal hybrid systems. *Journal of Complexity*, 21(4):447–478, 2005.
[20] Thomas Brihaye, Christian Michaux, Cédric Rivière, and Christophe Troestler. On o-minimal hybrid systems. In *HSCC'04: Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 219–233. Springer, 2004.
[21] Véronique Bruyère, Georges Hansel, Christian Michaux, and Roger Villemaire. Logic and p-recognizable sets of integers. *Bull. Belg. Math. Soc. Simon Stevin*, 1(2):191–238, 1994. Journées Montoises (Mons, 1992).
[22] Alberto Casagrande, Pietro Corvaja, Carla Piazza, and Bud Mishra. Composing semi-algebraic o-minimal automata. In *HSCC'07: Hybrid Systems: Computation and Control*, volume 4416 of *LNCS*, pages 668–671. Springer, 2007.
[23] Chen-chung Chang and H. Jerome Keisler. *Model theory*. North-Holland Publishing Co., Amsterdam, 1973. Studies in Logic and the Foundations of Mathematics, Vol. 73.
[24] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronous skeletons using branching-time temporal logic. In *Proc. 3rd Workshop Logics of Programs (LOP'81)*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.
[25] Catalin Dima. Kleene theorems for event-clock automata. In *FCT'99: Fundamentals of Computation Theory*, volume 1684 of *LNCS*, pages 215–225. Springer, 1999.
[26] Catalin Dima. Real-time automata. *Journal of Automata, Languages and Combinatorics*, 6(1):3–24, 2001.
[27] Lou van den Dries. *Tame Topology and O-Minimal Structures*, volume 248 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 1998.
[28] Olivier Finkel. Undecidable problems about timed automata. In *FORMATS'06: Formal Modeling and Analysis of Timed Systems*, volume 4202 of *LNCS*, pages 187–199. Springer, 2006.
[29] Thomas A. Henzinger. The theory of hybrid automata. In *LICS'96: Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996.
[30] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. A user guide to HyTech. In *TACAS'95: Tools and Algorithms for the Construction and Analysis of Systems*, volume 1019 of *Lecture Notes in Computer Science*, pages 41–71. Springer-Verlag, 1995.
[31] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata. *Journal of Computer and System Sciences*, 57(1):94–124, 1998.
[32] P. Herrmann. Timed automata and recognizability. *Information Processing Letters*, 65:313–318, 1998.
[33] Wilfrid Hodges. *Model theory*, volume 42 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1993.
[34] Wilfrid Hodges. *A shorter model theory*. Cambridge University Press, Cambridge, 1997.
[35] Julia F. Knight, Anand Pillay, and Charles Steinhorn. Definable sets in ordered structures. II. *Transactions of the American Mathematical Society*, 295(2):593–605, 1986.
[36] Gerardo Lafferriere, George J. Pappas, and Shankar Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13(1):1–21, 2000.
[37] Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. A new class of decidable hybrid systems. In *HSCC'99: Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 137–151. Springer, 1999.
[38] Kim G. Larsen, Paul Pettersson, and Wang Yi. Uppaal: Status & developments. In *CAV'97: Computer Aided Verification*, volume 1254 of *Lecture Notes in Computer Science*, pages 456–459. Springer, 1997.
[39] David Marker. *Model theory*, volume 217 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2002. An introduction.

[40] Joseph S. Miller. Decidability and complexity results for timed automata and semi-linear hybrid automata. In *HSCC'00: Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 296–309. Springer, 2000.

[41] Anand Pillay and Charles Steinhorn. Definable sets in ordered structures. I. *Transactions of the American Mathematical Society*, 295(2):565–592, 1986.

[42] Amir Pnueli. Th e temporal logic of programs. In *Proc. 18th Ann. Symp. Foundations of Computer Science (FOCS'77)*, pages 46–57. IEEECSP, 1977.

[43] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In *Proc. 5th Intl Symp. on Programming (SOP'82)*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 1982.

[44] Gerald E. Sacks. *Saturated model theory*. W. A. Benjamin, Inc., Reading, Mass., 1972. Mathematics Lecture Note Series.

[45] Stavros Tripakis. Folk theorems on the determinization and minimization of timed automata. *Inf. Process. Lett.*, 99(6):222–226, 2006.

[46] Volker Weispfenning. Mixed real-integer linear quantifier elimination. In *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation (Vancouver, BC)*, pages 129–136 (electronic), New York, 1999. ACM.

[47] Alex J. Wilkie. Model completeness results for expansions of the ordered field of real numbers by restricted Pfaffian functions and the exponential function. *Journal of the American Mathematical Society*, 9(4):1051–1094, 1996.

[48] Li Xuandong, Zheng Tao, Hou Jianmin, Zhao Jianhua, and Zheng Guoliang. Hybrid regular expressions. In *Proceedings of the First International Workshop on Hybrid Systems: Computation and Control*, volume 1386 of *Lecture Notes in Computer Science*, pages 384–399. Springer, 1998.

Communicated by (The editor will be set by the publisher).
(The dates will be set by the publisher).