On Reachability for Hybrid Automata over Bounded Time*

Thomas Brihaye[1], Laurent Doyen[2], Gilles Geeraerts[3],
Joël Ouaknine[4], Jean-François Raskin[3], and James Worrell[4]

[1] Université de Mons, Belgium
[2] LSV, ENS Cachan & CNRS, France
[3] Université Libre de Bruxelles, Belgium
[4] Oxford University Computing Laboratory, UK

Abstract. This paper investigates the time-bounded version of the reachability problem for hybrid automata. This problem asks whether a given hybrid automaton can reach a given target location within $\mathbf{T}$ time units, where $\mathbf{T}$ is a constant rational value. We show that, in contrast to the classical (unbounded) reachability problem, the timed-bounded version is *decidable* for rectangular hybrid automata provided only non-negative rates are allowed. This class of systems is of practical interest and subsumes, among others, the class of stopwatch automata. We also show that the problem becomes undecidable if either diagonal constraints or both negative and positive rates are allowed.

1 Introduction

The formalism of hybrid automata [1] is a well-established model for hybrid systems whereby a digital controller is embedded within a physical environment. The state of a hybrid system changes both through discrete transitions of the controller, and continuous evolutions of the environment. The discrete state of the system is encoded by the *location* ℓ of the automaton, and the continuous state is encoded by *real-valued variables* X evolving according to dynamical laws constraining the first derivative $\dot{X}$ of the variables. Hybrid automata have proved useful in many applications, and their analysis is supported by several tools [6, 5].

A central problem in hybrid-system verification is the *reachability problem* which is to decide if there exists an execution from a given initial location ℓ to a given goal location ℓ'. While the reachability problem is undecidable for simple classes of hybrid

* Work supported by the projects: (i) QUASIMODO (FP7- ICT-STREP-214755), Quasimodo: "Quantitative System Properties in Model-Driven-Design of Embedded", `http://www.quasimodo.aau.dk/`, (ii) GASICS (ESF-EUROCORES LogiCCC), Gasics: "Games for Analysis and Synthesis of Interactive Computational Systems", `http://www.ulb.ac.be/di/gasics/`, (iii) Moves: "Fundamental Issues in Modelling, Verification and Evolution of Software", `http://moves.ulb.ac.be`, a PAI program funded by the Federal Belgian Government, (iv) the ARC project AUWB-2010–10/15-UMONS-3, (v) the FRFC project 2.4515.11 and (vi) a grant from the National Bank of Belgium.

automata (such as linear hybrid automata [1]), the decidability frontier of this problem is sharply understood [7, 8]. For example, the reachability problem is decidable for the class of initialized rectangular automata where (i) the flow constraints, guards, invariants and discrete updates are defined by rectangular constraints of the form $a \leq \dot{x} \leq b$ or $c \leq x \leq d$ (where a, b, c, d are rational constants), and (ii) whenever the flow constraint of a variable x changes between two locations ℓ and ℓ', then x is reset along the transition from ℓ to ℓ'. Of particular interest is the class of timed automata which is a special class of initialized rectangular automata [2].

In recent years, it has been observed that new decidability results can be obtained in the setting of time-bounded verification of real-time systems [10, 11]. Given a time bound $\mathbf{T} \in \mathbb{N}$, the time-bounded verification problems consider only traces with duration at most $\mathbf{T}$. Note that due to the density of time, the number of discrete transitions may still be unbounded. Several verification problems for timed automata and real-time temporal logics turn out to be decidable in the time-bounded framework (such as the language-inclusion problem for timed automata [10]), or to be of lower complexity (such as the model-checking problem for MTL [11]). The theory of time-bounded verification is therefore expected to be more robust and better-behaved in the case of hybrid automata as well.

Following this line of research, we revisit the reachability problem for hybrid automata with time-bounded traces. The *time-bounded reachability problem* for hybrid automata is to decide, given a time bound $\mathbf{T} \in \mathbb{N}$, if there exists an execution of duration less than $\mathbf{T}$ from a given initial location ℓ to a given goal location ℓ'. We study the frontier between decidability and undecidability for this problem and show how bounding time alters matters with respect to the classical reachability problem. In this paper, we establish the following results. First, we show that the time-bounded reachability problem is *decidable* for non-initialized rectangular automata when only positive rates are allowed[5]. The proof of this fact is technical and, contrary to most decidability results in the field, does not rely on showing the existence of an underlying finite (bi)simulation quotient. We study the properties of time-bounded runs and show that if a location is reachable within $\mathbf{T}$ time units, then it is reachable by a timed run in which the number of discrete transitions can be bounded. This in turn allows us to reduce the time-bounded reachability problem to the satisfiability of a formula in the first-order theory of real addition, decidable in EXPSPACE [4].

Second, we show that the time-bounded reachability problem is *undecidable* for non-initialized rectangular hybrid automata if both positive and negative rates are allowed. Third, we show that the time-bounded reachability problem is *undecidable* for initialized rectangular hybrid automata with positive singular flows if diagonal constraints in guards are allowed. These two undecidability results allow to precisely characterise the boundary between decidability and undecidability.

The undecidability results are obtained by reductions from the halting problem for two-counter machines. We present novel encodings of the execution of two-counter machines that fit into time-bounded executions of hybrid automata with either negative rates, or diagonal constraints.

[5] This class is interesting from a practical point of view as it includes, among others, the class of stopwatch automata [3], for which unbounded reachability is undecidable.

2 Definitions

Let $\mathcal{I}$ be the set of intervals of real numbers with endpoints in $\mathbb{Z} \cup \{-\infty, +\infty\}$. Let X be a set of continuous variables, and let $X' = \{x' \mid x \in X\}$ and $\dot{X} = \{\dot{x} \mid x \in X\}$ be the set of primed and dotted variables, corresponding respectively to variable updates and first derivatives. A *rectangular constraint* over X is an expression of the form $x \in I$ where x belongs to X and I to $\mathcal{I}$. A *diagonal constraint* over X is a constraint of the form $x - y \sim c$ where x, y belong to X, c to $\mathbb{Z}$, and $\sim$ is in $\{<, \leq, =, \geq, >\}$. Finite conjunctions of diagonal and rectangular constraints over X are called *guards*, over $\dot{X}$ they are called *rate constraints*, and over $X \cup X'$ they are called *update constraints*. A guard or rate constraint is *rectangular* if all its constraints are rectangular. An update constraint is *rectangular* if all its constraints are either rectangular or of the form $x = x'$. We denote by $\mathcal{G}(X), \mathcal{R}(X), \mathcal{U}(X)$ respectively the sets of guards, rate constraints, and update constraints over X.

Linear hybrid automata. A *linear hybrid automaton* (LHA) is a tuple $\mathcal{H} = (X, \mathrm{Loc}, \mathrm{Edges}, \mathrm{Rates}, \mathrm{Inv}, \mathrm{Init})$ where $X = \{x_1, \ldots, x_{|X|}\}$ is a finite set of continuous *variables*; Loc is a finite set of *locations*; $\mathrm{Edges} \subseteq \mathrm{Loc} \times \mathcal{G}(X) \times \mathcal{U}(X) \times \mathrm{Loc}$ is a finite set of *edges*; $\mathrm{Rates} : \mathrm{Loc} \mapsto \mathcal{R}(X)$ assigns to each location a constraint on the *possible variable rates*; $\mathrm{Inv} : \mathrm{Loc} \mapsto \mathcal{G}(X)$ assigns an *invariant* to each location; and $\mathrm{Init} \in \mathrm{Loc}$ is an *initial location*. For an edge $e = (\ell, g, r, \ell')$, we denote by $\mathsf{src}(e)$ and $\mathsf{trg}(e)$ the location ℓ and ℓ' respectively, g is called the *guard* of e and r is the *update* (or *reset*) of e. In the sequel, we denote by rmax the maximal constant occurring in the constraints of $\{\mathrm{Rates}(\ell) \mid \ell \in \mathrm{Loc}\}$

A LHA $\mathcal{H}$ is *singular* if for all locations ℓ and for all variables x of $\mathcal{H}$, the only constraint over $\dot{x}$ in $\mathrm{Rates}(\ell)$ is of the form $\dot{x} \in I$ where I is a singular interval; it is *fixed rate* if for all variables x of $\mathcal{H}$ there exists $I_x \in \mathcal{I}$ such that for all locations ℓ of $\mathcal{H}$, the only constraint on $\dot{x}$ in $\mathrm{Rates}(\ell)$ is the constraint $\dot{x} \in I_x$. It is *multirate* if it is not fixed rate. It is *non-negative rate* if for all variables x, for all locations ℓ, the constraint $\mathrm{Rates}(\ell)$ implies that $\dot{x}$ must be non-negative.

Rectangular hybrid automata. A *rectangular hybrid automaton* (RHA) is a linear hybrid automaton in which all guards, rates, and invariants are rectangular. In this case, we view each reset r as a function $X' \mapsto \mathcal{I} \cup \{\perp\}$ that associates to each variable $x \in X$ either an interval of possible reset values $r(x)$, or $\perp$ when the value of the variable x remains unchanged along the transition. When it is the case that $r(x)$ is either $\perp$ or a singular interval for each x, we say that r is *deterministic*. In the case of RHA, we can also view rate constraints as functions $\mathrm{Rates} : \mathrm{Loc} \times X \to \mathcal{I}$ that associate to each location ℓ and each variable x an interval of possible rates $\mathrm{Rates}(\ell)(x)$. A rectangular hybrid automaton $\mathcal{H}$ is *initialized* if for every edge (ℓ, g, r, ℓ') of $\mathcal{H}$, for every $x \in X$, if $\mathrm{Rates}(\ell)(x) \neq \mathrm{Rates}(\ell')(x)$ then $r(x) \neq \perp$, i.e., every variable whose rate constraint is changed must be reset.

LHA semantics. A *valuation* of a set of variables X is a function $\nu : X \mapsto \mathbb{R}$. We further denote by $\mathbf{0}$ the valuation that assigns 0 to each variable.

Given an LHA $\mathcal{H} = (X, \mathrm{Loc}, \mathrm{Edges}, \mathrm{Rates}, \mathrm{Inv}, \mathrm{Init}, X)$, a *state* of $\mathcal{H}$ is a pair (ℓ, ν), where $\ell \in \mathrm{Loc}$ and ν is a valuation of X. The semantics of $\mathcal{H}$ is defined as follows. Given a state $s = (\ell, \nu)$ of $\mathcal{H}$, an *edge step* $(\ell, \nu) \xrightarrow{e} (\ell', \nu')$ can occur and change the state to (ℓ', ν') if $e = (\ell, g, r, \ell') \in \mathrm{Edges}$, $\nu \models g$, $\nu'(x) = \nu(x)$ for all x s.t. $r(x) = \bot$, and $\nu'(x) \in r(x)$ for all x s.t. $r(x) \neq \bot$; given a time delay $t \in \mathbb{R}^+$, a *continuous time step* $(\ell, \nu) \xrightarrow{t} (\ell, \nu')$ can occur and change the state to (ℓ, ν') if there exists a vector $r = (r_1, \ldots r_{|X|})$ such that $r \models \mathrm{Rates}(\ell)$, $\nu' = \nu + (r \cdot t)$, and $\nu + (r \cdot t') \models \mathrm{Inv}(\ell)$ for all $0 \leq t' \leq t$.

A *path* in $\mathcal{H}$ is a finite sequence $e_1, e_2, \ldots, e_n$ of edges such that $\mathsf{trg}(e_i) = \mathsf{src}(e_{i+1})$ for all $1 \leq i \leq n-1$. A *cycle* is a path $e_1, e_2, \ldots, e_n$ such that $\mathsf{trg}(e_n) = \mathsf{src}(e_1)$. A cycle $e_1, e_2, \ldots, e_n$ is *simple* if $\mathsf{src}(e_i) \neq \mathsf{src}(e_j)$ for all $i \neq j$. A *timed path* of $\mathcal{H}$ is a finite sequence of the form $\pi = (t_1, e_1), (t_2, e_2), \ldots, (t_n, e_n)$, such that $e_1, \ldots, e_n$ is a path in $\mathcal{H}$ and $t_i \in \mathbb{R}^+$ for all $0 \leq i \leq n$. We lift the notions of cycle and simple cycle to the timed case accordingly. Given a timed path $\pi = (t_1, e_1), (t_2, e_2), \ldots, (t_n, e_n)$, we denote by $\pi[i : j]$ (with $1 \leq i \leq j \leq n$) the timed path $(t_i, e_i), \ldots, (t_j, e_j)$.

A *run* in $\mathcal{H}$ is a sequence $s_0, (t_0, e_0), s_1, (t_1, e_1), \ldots, (t_{n-1}, e_{n-1}), s_n$ such that:

- $(t_0, e_0), (t_1, e_1), \ldots, (t_{n-1}, e_{n-1})$ is a timed path in $\mathcal{H}$, and
- for all $1 \leq i < n$, there exists a state s_i' of $\mathcal{H}$ with $s_i \xrightarrow{t_i} s_i' \xrightarrow{e_i} s_{i+1}$.

Given a run $\rho = s_0, (t_0, e_0), \ldots, s_n$, let $\mathsf{first}(\rho) = s_0 = (\ell_0, \nu_0)$, $\mathsf{last}(\rho) = s_n$, $\mathsf{duration}(\rho) = \sum_{i=1}^{n-1} t_i$, and $|\rho| = n + 1$. We say that ρ is (i) *strict* if $t_i > 0$ for all $1 \leq i \leq n-1$; (ii) *k-variable-bounded* (for $k \in \mathbb{N}$) if $\nu_0(x) \leq k$ for all $x \in X$, and $s_i \xrightarrow{t_i} (\ell_i, \nu_i)$ implies that $\nu_i(x) \leq k$ for all $0 \leq i \leq n$; (iii) *$\mathbf{T}$-time-bounded* (for $\mathbf{T} \in \mathbb{N}$) if $\mathsf{duration}(\rho) \leq \mathbf{T}$.

Note that a unique timed path $\mathsf{TPath}(\rho) = (t_0, e_0), (t_1, e_1), \ldots, (t_{n-1}, e_{n-1})$, is associated to each run $\rho = s_0, (t_0, e_0), s_1, \ldots, (t_{n-1}, e_{n-1}), s_n$. Hence, we sometimes abuse notation and denote a run ρ with $\mathsf{first}(\rho) = s_0$, $\mathsf{last}(\rho) = s$ and $\mathsf{TPath}(\rho) = \pi$ by $s_0 \xrightarrow{\pi} s$. The converse however is not true: given a timed path π and an initial state s_0, it could be impossible to build a run starting from s_0 and following π because some guards or invariants along π might be violated. However, if such a run exists it is necessarily unique *when the automaton is singular and all resets are deterministic*. In that case, we denote by $\mathsf{Run}(s_0, \pi)$ the function that returns the unique run ρ such that $\mathsf{first}(\rho) = s_0$ and $\mathsf{TPath}(\rho) = \pi$ if it exists, and $\bot$ otherwise.

Time-bounded reachability problem for LHA. While the reachability problem asks to decide the existence of any timed run that reaches a given goal location, we are only interested in runs having bounded duration.

Problem 1 (Time-bounded reachability problem). Given an LHA $\mathcal{H} = (X, \mathrm{Loc}, \mathrm{Edges}, \mathrm{Rates}, \mathrm{Inv}, \mathrm{Init})$, a location $\mathrm{Goal} \in \mathrm{Loc}$ and a time bound $\mathbf{T} \in \mathbb{N}$, the *time-bounded reachability problem* is to decide whether there exists a finite run $\rho = (\mathrm{Init}, \mathbf{0}) \xrightarrow{\pi} (\mathrm{Goal}, \cdot)$ of $\mathcal{H}$ with $\mathsf{duration}(\rho) \leq \mathbf{T}$.

In the following table, we summarize the known facts regarding decidability of the reachability problem for LHA, along with the results on time-bounded reachability that we prove in the rest of this paper. Note that decidability for initialized rectangular hybrid automata (IHRA) follows directly from [7]. We show decidability for (non-initialized) RHA that only have non-negative rates in Section 3. The undecidability of the time-bounded reachability problem for RHA and LHA is not a consequence of the known results from the literature and require new proofs that are given in Section 4.

HA classes	Reachability	Time-Bounded Reachability
LHA	U [1]	**U** (see Section 4)
RHA	U [7]	**U** (see Section 4)
non-negative rates RHA	U [7]	**D** (see Section 3)
IRHA	D [7]	D [7]

3 Decidability for RHA with Non-Negative Rates

In this section, we prove that the time-bounded reachability problem is *decidable* for the class of (non-initialized) *rectangular* hybrid automata having *non-negative rates*, while it is *undecidable* for this class in the classical (unbounded) case [7]. Note that this class is interesting in practice since it contains, among others, the important class of *stopwatch automata*, a significant subset of LHA that has several useful applications [3]. We obtain decidability by showing that for RHA with non-negative rates, a goal location is reachable within $\mathbf{T}$ time units iff there exists a witness run of that automaton which reaches the goal (within $\mathbf{T}$ time units) by a run ρ of length $|\rho| \leq K_{\mathbf{T}}^{\mathcal{H}}$ where $K_{\mathbf{T}}^{\mathcal{H}}$ is a parameter that depends on $\mathbf{T}$ and on the size of the automaton $\mathcal{H}$. Time-bounded reachability can thus be reduced to the satisfiability of a formula in the first order theory of the reals encoding the existence of runs of length at most $K_{\mathbf{T}}^{\mathcal{H}}$ and reaching Goal.

For simplicity of the proofs, we consider RHA with the following restrictions: (i) the guards *do not contain strict inequalities*, and (ii) the rates are *singular*. We argue at the end of this section that these restrictions can be made without loss of generality. Then, in order to further simplify the presentation, we show how to syntactically simplify the automaton while preserving the time-bounded reachability properties. The details of the constructions can be found in the appendix.

Proposition 1. *Let $\mathcal{H}$ be a singular RHA with non-negative rates and without strict inequalities, and let* Goal *be a location of $\mathcal{H}$. We can build a hybrid automaton $\mathcal{H}'$ with the following the properties:*

- H_1 *$\mathcal{H}'$ is a singular RHA with non-negative rates*
- H_2 *$\mathcal{H}'$ contains only deterministic resets*
- H_3 *for every edge (ℓ, g, r, ℓ') of $\mathcal{H}'$, g is either $\mathbf{true}$ or of the form $x_1 = 1 \wedge x_2 = 1 \wedge \cdots \wedge x_k = 1$, and $r \equiv x_1' = 0 \wedge \cdots \wedge x_k' = 0$.*

and a set of locations S of $\mathcal{H}'$ such that $\mathcal{H}$ admits a $\mathbf{T}$-time bounded run reaching Goal *iff $\mathcal{H}'$ admits a strict 1-variable-bounded, and $\mathbf{T}$-time bounded run reaching S.*

Proof (Sketch). The proof exposes three transformations that we apply to $\mathcal{H}$ in order to obtain $\mathcal{H}'$. The first transformation turns $\mathcal{H}$ into $\mathsf{DetReset}\,(\mathcal{H})$, containing deterministic resets only. The idea is to replace (non-deterministic) resets in $\mathcal{H}$ with resets to 0 in

$\mathsf{DetReset}\,(\mathcal{H})$ and to compensate by suitably altering the guards of subsequent transitions in $\mathsf{DetReset}\,(\mathcal{H})$. To achieve this, locations in $\mathsf{DetReset}\,(\mathcal{H})$ are elements of the form (ℓ, ρ), where ℓ is a location of $\mathcal{H}$ and ρ associates an interval to each variable, where $\rho(j)$ represents the interval in which variable x_j was last reset.

With the second transformation, we can restrict our analysis to runs where the variables are bounded by 1. The idea is to encode the integer parts of the variables in the locations, and to adapt the guards and the resets. Let $\mathcal{H}'$ be an RHA obtained from the first step, with maximal constant cmax. We build $\mathsf{CBound}\,(\mathcal{H}')$ whose locations are of the form $(\ell, \mathbf{i})$, where ℓ is a location of $\mathcal{H}'$, and $\mathbf{i}$ is a function that associates a value from $\{0, \ldots, \mathrm{cmax}\}$ to each variable. Intuitively, $\mathbf{i}(j)$ represents the integer part of x_j in the original run of $\mathcal{H}'$, whereas the fractional part is tracked by x_j (hence all the variables stay in the interval $[0, 1]$). Guards and resets are adapted consequently.

The third and last construction allows to consider only runs where time is stricly increasing. We describe it briefly assuming all the invariants are $\mathbf{true}$ to avoid technicalities. Consider a sequence of edges and null time delays of the form $e_1, 0, e_2, 0, \ldots, 0, e_n$ (remark that this sequence ends and starts with an edge). Since all time delays are null, the only effect of firing such as sequence is to reset (to zero by the first construction) all the variables that are reset by one of the e_i's. Thus, this sequence can be replaced by a single edge $e = (\ell, g, r, \ell')$ with the same effect, that is, where ℓ is the origin of e_1, ℓ' is the destination of e_n, r resets to zero all the variables that are reset by one of the e_i's and g summarises the guards of all the e_i's (taking the resets into account). Moreover, we only need to consider sequences where $e_1, e_2, \ldots, e_n$ is a path where each simple loop appears at most once (traversing a second time a simple loop would only reset variables that are already equal to zero because the time delays are null). Thus, the construction amounts to enumerate all the possible paths π where each simple loop appears at most once, and to add to the automaton an edge e_π that summarises π as described above. Since there are finitely many such π the construction is effective.

As a consequence, to prove decidability of time-bounded reachability of RHA with non-negative rates, we only need to prove that we can decide whether an RHA respecting H_1 through H_3 admits a *strict* run ρ reaching the goal within $\mathbf{T}$ time units, and where all variables are bounded by 1 along ρ.

Bounding the number of equalities. As a first step to obtain a witness of time-bounded reachability, we bound the number of transitions guarded by equalities along a run of bounded duration:

Proposition 2. *Let $\mathcal{H}$ be an LHA, with set of variables X and respecting hypothesis H_1 through H_3. Let ρ be a $\mathbf{T}$-time bounded run of $\mathcal{H}$. Then, ρ contains at most $|X|\cdot\mathrm{rmax}\cdot\mathbf{T}$ transitions guarded by an equality.*

Bounding runs without equalities. Unfortunately, it is not possible to bound the number of transitions that do not contain equalities, even along a time-bounded run. However, we will show that, given a time-bounded run ρ without equality guards, we can build a run ρ' that is equivalent to ρ (in a sense that its initial and target states are the same), and whose length is *bounded* by a parameter depending on the size of the automaton. More precisely:

Proposition 3. *Let $\mathcal{H}$ be an RHA with non-negative rates. For any 1-variable bounded and $\frac{1}{\mathrm{rmax}+1}$-time bounded run $\rho = s_0 \xrightarrow{\pi} s$ of $\mathcal{H}$ that contains no equalities in the guards, $\mathcal{H}$ admits a 1-variable bounded and $\frac{1}{\mathrm{rmax}+1}$-time bounded run $\rho' = s_0 \xrightarrow{\pi'} s$ such that $|\rho'| \leq 2|X| + (2|X|+1) \cdot |\mathrm{Loc}| \cdot (2^{(|\mathrm{Edges}|+1)} + 1)$.*

Note that Proposition 3 applies only to runs of duration at most $\frac{1}{\mathrm{rmax}+1}$. However, this is not restrictive, since any $\mathbf{T}$-time-bounded run can always be split into at most $\mathbf{T}\cdot(\mathrm{rmax}+1)$ subruns of duration at most $\frac{1}{\mathrm{rmax}+1}$, provided that we add a self-loop with guard $\mathbf{true}$ and no reset on every location (this can be done without loss of generality as far as reachability is concerned).

To prove Proposition 3, we rely on a *contraction operation* that receives a *timed path* and returns another one of smaller length. Let $\pi = (t_1, e_1), (t_2, e_2), \ldots, (t_n, e_n)$ be a timed path. We define $\mathsf{Cnt}\,(\pi)$ by considering two cases. Let j, k, j', k' be four positions such that $1 \leq j \leq k < j' \leq k' \leq n$ and $e_j \ldots e_k = e'_j \ldots e'_k$ is a *simple cycle*. **If** such j, k, j', k' exist, then let:

$$\mathsf{Cnt}\,(\pi) = \pi[1:j-1]\cdot(e_j, t_j + t_{j'})\cdots(e_k, t_k + t_{k'})\cdot\pi[k+1:j'-1]\cdot\pi[k'+1:n]$$

Otherwise, we let $\mathsf{Cnt}\,(\pi) = \pi$. Observe that π and $\mathsf{Cnt}\,(\pi)$ share the same source and target locations, even when $\pi[k'+1:n]$ is empty.

Then, given a timed path π, we let $\mathsf{Cnt}^0\,(\pi) = \pi$, $\mathsf{Cnt}^i\,(\pi) = \mathsf{Cnt}\left(\mathsf{Cnt}^{i-1}\,(\pi)\right)$ for any $i \geq 1$, and $\mathsf{Cnt}^*\,(\pi) = \mathsf{Cnt}^n\,(\pi)$ where n is the least value such that $\mathsf{Cnt}^n\,(\pi) = \mathsf{Cnt}^{n+1}\,(\pi)$. Clearly, since π is finite, and since $|\mathsf{Cnt}\,(\pi)| < |\pi|$ or $\mathsf{Cnt}\,(\pi) = \pi$ for any π, $\mathsf{Cnt}^*\,(\pi)$ always exists. Moreover, we can always bound the length of $\mathsf{Cnt}^*\,(\pi)$. This stems from the fact that $\mathsf{Cnt}^*\,(\pi)$ is a timed path that contains at most one occurrence of each simple cycle. The length of such paths can be bounded using classical combinatorial arguments.

Lemma 1. *For any timed path π of an LHA $\mathcal{H}$ with $|\mathrm{Loc}|$ locations and $|\mathrm{Edges}|$ edges: $|\mathsf{Cnt}^*\,(\pi)| \leq |\mathrm{Loc}| \cdot (2^{(|\mathrm{Edges}|+1)} + 1)$.*

Note that the contraction operation is purely syntactic and works on the timed path only. Hence, given a run $s_0 \xrightarrow{\pi} s$, we have no guarantee that $\mathsf{Run}\,(s_0, \mathsf{Cnt}^*\,(\pi)) \neq \perp$. Moreover, even in the alternative, the resulting run might be $s_0 \xrightarrow{\mathsf{Cnt}^*(\pi)} s'$ with $s \neq s'$. Nevertheless, we can show that $\mathsf{Cnt}^*\,(\pi)$ preserves some properties of π. For a timed path $\pi = (t_1, e_1), \ldots, (t_n, e_n)$ of an LHA $\mathcal{H}$ with rate function Rates, we let $\mathsf{Effect}\,(\pi, x) = \sum_{i=1}^{n} \mathrm{Rates}(\ell_i)(x) \cdot t_i$, where ℓ_i is the initial location of e_i for any $1 \leq i \leq n$. Note thus that, for any run $(\ell, \nu) \xrightarrow{\pi} (\ell', \nu')$, for any variable x *which is not reset along* π, $\nu'(x) = \nu(x) + \mathsf{Effect}\,(\pi, x)$. It is easy to see that $\mathsf{Cnt}^*\,(\pi)$ preserves the effect of π. Moreover, the duration of $\mathsf{Cnt}^*\,(\pi)$ and π are equal.

Lemma 2. *For any timed path π: (i)* $\mathsf{duration}\,(\pi) = \mathsf{duration}\,(\mathsf{Cnt}^*\,(\pi))$ *and (ii) for any variable x:* $\mathsf{Effect}\,(\pi, x) = \mathsf{Effect}\,(\mathsf{Cnt}^*\,(\pi), x)$.

We are now ready to show, given a timed path π (with $\mathsf{duration}\,(\pi) \leq \frac{1}{\mathrm{rmax}+1}$ and without equality tests in the guards), how to build a timed path $\mathsf{Contraction}\,(\pi)$ that fully preserves the values of the variable, as stated in Proposition 3. The key ingredient to obtain $\mathsf{Contraction}(\pi)$ is to apply Cnt^* to selected portions of π, in such a way that for each edge e that resets a variable for the *first* or the *last* time along π, the time distance between the occurrence of e and the beginning of the timed path is the same in both π and $\mathsf{Contraction}(\pi)$.

The precise construction goes as follows. Let $\pi = (t_1, e_1), \ldots, (t_n, e_n)$ be a timed path. For each variable x, we denote by S_x^π the set of positions i such that e_i is either the *first* or the *last* edge in π to reset x (hence $|S_x^\pi| \in \{0, 1, 2\}$ for any x). Then, we decompose π as: $\pi_1 \cdot (t_{i_1}, e_{i_1}) \cdot \pi_2 \cdot (t_{i_2}, e_{i_2}) \cdots (t_{i_k}, e_{i_k}) \cdot \pi_{k+1}$ with $\{i_1, \ldots, i_k\} = \cup_x S_x^\pi$. From this decomposition of π, we let $\mathsf{Contraction}(\pi) = \mathsf{Cnt}^*(\pi_1) \cdot (t_{i_1}, e_{i_1}) \cdot \mathsf{Cnt}^*(\pi_2) \cdot (t_{i_2}, e_{i_2}) \cdots (t_{i_k}, e_{i_k}) \cdot \mathsf{Cnt}^*(\pi_{k+1})$.

We first note that, thanks to Lemma 1, $|\mathsf{Contraction}(\pi)|$ is bounded.

Lemma 3. *Let $\mathcal{H}$ be an LHA with set of variable X, set of edges* Edges *and set of location* Loc*, and let π be a timed path of $\mathcal{H}$. Then* $|\mathsf{Contraction}(\pi)| \leq 2 \cdot |X| + (2 \cdot |X| + 1) \cdot |\mathrm{Loc}| \cdot (2^{(|\mathrm{Edges}|+1)} + 1)$.

We can now prove Proposition 3.

Proof (Sketch – Proposition 3). Let $\pi = \mathsf{TPath}(\rho)$ and let π' denote $\mathsf{Contraction}(\pi)$. We let $\rho' = s_0 \xrightarrow{\pi'} (\ell', \nu')$, and prove (i) that firing π' from s_0 will always keep all the variable values ≤ 1, which implies $\mathsf{Run}(s_0, \pi') \neq \bot$, and (ii) that $\rho = s_0 \xrightarrow{\pi} (\ell, \nu)$ implies $\ell' = \ell$ and $\nu = \nu'$. These two points hold because $\mathsf{duration}(\mathsf{Cnt}^*(\pi_j)) = \mathsf{duration}(\pi_j)$ for any j. Hence, the first and last resets of each variable happen at the same time (relatively to the beginning of the timed path) in both π and $\mathsf{Contraction}(\pi)$. Intuitively, preserving the time of occurrence of the first reset (of some variable x) guarantees that x will never exceed 1 along $\mathsf{Contraction}(\pi)$, as $\mathsf{duration}(\mathsf{Contraction}(\pi)) = \mathsf{duration}(\pi) \leq \frac{1}{\mathrm{rmax}+1}$. Symmetrically, preserving the last reset of some variable x guarantees that the final value of x will be the same in both π and $\mathsf{Contraction}(\pi)$. Moreover, the contraction preserves the value of the variables that are not reset, by Lemma 2.

Handling '$<$' and non-singular rates. Let us now briefly explain how we can adapt the construction of this section to cope with strict guards and non-singular rates. First, when the RHA $\mathcal{H}$ contains strict guards, the RHA $\mathcal{H}'$ of Proposition 1 will also contain guards with atoms of the form $x < 1$. Thus, when building a 'contracted path' ρ' starting from a path ρ (as in the proof of Proposition 3), we need to ensure that these strict guards will also be satisfied along ρ'. It is easy to use similar arguments to establish this: if some guard $x < 1$ is not satisfied in ρ', this is necessarily before the first reset of x, which means that the guard was not satisfied in ρ either. On the other hand, to take non-singular rates into account, we need to adapt the definition of timed path. A timed path is now of the form $(t_0, r_0, e_0) \cdots (t_n, r_n, e_n)$, where each r_i is a vector of reals of size $|X|$, indicating the actual rate that was chosen for each variable when the i-th continuous step has been taken. It is then straightforward to adapt the definitions of Cnt, Effect and $\mathsf{Contraction}$ to take those rates into account and still keep the properties stated in Lemma 1 and 3 and in Proposition 3 (note that we need to rely on the convexity of the invariants in RHA to ensure that proper rates can be found when building $\mathsf{Cnt}(\pi)$).

Theorem 1. *The time-bounded reachability problem is decidable for the class of rectangular hybrid automata with non-negative rates.*

Proof (Sketch). Given an RHA H, a bound K, and a goal Goal, we can build a formula φ of $\mathsf{FO}(\mathbb{R}, \leq, +)$ that is satisfiable iff $\mathcal{H}$ admits a run of length $\leq K$ reaching Goal. By Proposition 1 (and taking into account the above remarks to cope with strict guards and rectangular rates), this is sufficient to decide time-bounded reachability on RHA with non-negative rates. The required result now follows from the decidability of satisfiability for $\mathsf{FO}(\mathbb{R}, \leq, +)$. $\square$

4 Undecidability Results

In this section, we show that the time-bounded reachability problem for linear hybrid automata becomes undecidable if either both positive and negative rates are allowed, or diagonal constraints are allowed in the guards. Along with the decidability result of Section 3, these facts imply that the class of rectangular hybrid automata having positive rates only and no diagonal constraints forms a maximal decidable class. Our proofs rely on reductions from the halting problem for Minsky two-counters machines.

A *two-counter machine* M consists of a finite set of control states Q, an initial state $q_I \in Q$, a final state $q_F \in Q$, a set C of counters ($|C| = 2$) and a finite set δ_M of instructions manipulating two integer-valued counters. Instructions are of the form:

$q :\ c := c + 1$ **goto** q', or
$q :$ **if** $c = 0$ **then goto** q' **else** $c := c - 1$ **goto** q''.

Formally, instructions are tuples (q, α, c, q') where $q, q' \in Q$ are source and target states respectively, the action $\alpha \in \{inc, dec, 0?\}$ applies to the counter $c \in C$.

A *configuration* of M is a pair (q, v) where $q \in Q$ and $v : C \to \mathbb{N}$ is a valuation of the counters. An *accepting run* of M is a finite sequence $\pi = (q_0, v_0)\delta_0(q_1, v_1)\delta_1 \ldots \delta_{n-1}(q_n, v_n)$ where $\delta_i = (q_i, \alpha_i, c_i, q_{i+1}) \in \delta_M$ are instructions and (q_i, v_i) are configurations of M such that $q_0 = q_I$, $v_0(c) = 0$ for all $c \in C$, $q_n = q_F$, and for all $0 \leq i < n$, we have $v_{i+1}(c) = v_i(c)$ for $c \neq c_i$, and (i) if $\alpha = inc$, then $v_{i+1}(c_i) = v_i(c_i) + 1$, (ii) if $\alpha = dec$, then $v_i(c_i) \neq 0$ and $v_{i+1}(c_i) = v_i(c_i) - 1$, and (iii) if $\alpha = 0?$, then $v_{i+1}(c_i) = v_i(c_i) = 0$. The *halting problem* asks, given a two-counter machine M, whether M has an accepting run. This problem is undecidable [9].

Undecidability for RHA with negative rates. Given a two-counter machine M, we construct an RHA $\mathcal{H}_M$ (thus without diagonal constraints) such that M has an accepting run if and only if the answer to the time-bounded reachability problem for $(\mathcal{H}_M, \mathrm{Goal})$ with time bound 1 is YES. The construction of $\mathcal{H}_M$ crucially makes use of both positive and negative rates.

Theorem 2. *The time-bounded reachability problem is undecidable for rectangular hybrid automata even if restricted to singular rates.*

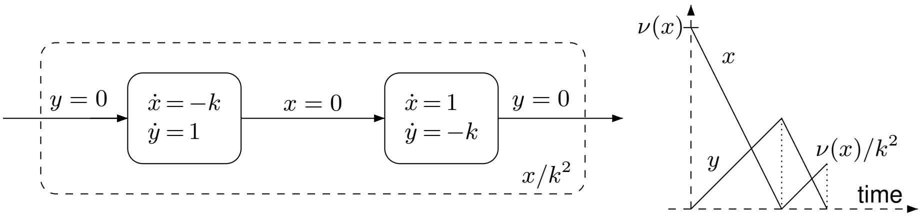

Fig. 1. Gadget for division of a variable x by k^2. The variable y is internal to the gadget. The duration of the division is $v \cdot (\frac{1}{k} + \frac{1}{k^2})$ where v is the value of x before division.

Proof (Sketch). First, remark that the main difficulty of the reduction is to encode *unbounded* computations of M within a *bounded* time slot. The execution steps of M are simulated in $\mathcal{H}_M$ by a (possibly infinite) sequence of *ticks* within one time unit. The ticks occur at time $t_0 = 0, t_1 = 1 - \frac{1}{4}, t_2 = 1 - \frac{1}{16}$, etc. The counters are encoded as follows. If the value of counter $c \in C$ after i execution steps of M is $v(c)$, then the variable x_c in $\mathcal{H}_M$ has value $\frac{1}{4^{i+v(c)}}$ at time t_i. Note that this encoding is time-dependent and that the value of x_c at time t_i is always smaller than $1 - t_i = \frac{1}{4^i}$, and equal to $\frac{1}{4^i}$ if the counter value is 0. To maintain this encoding (if a counter c is not modified in an execution step), we need to divide x_c by 4 before the next tick occurs. We use the divisor gadget in Fig. 1 to do this. Using the diagram in the figure, it is easy to check that the value of variable x_c is divided by k^2 where k is a constant used to define the variable rates. Note also that the division of $\nu(x_c)$ by k^2 takes $\nu(x_c) \cdot (\frac{1}{k} + \frac{1}{k^2})$ time units, which is less than $\frac{3 \cdot \nu(x_c)}{4}$ for $k \geq 2$. Since $\nu(x_c) \leq \frac{1}{4^i}$ at step t_i, the duration of the division is at most $\frac{3}{4^i} = t_{i+1} - t_i$, the duration of the next tick.

The divisor gadget can also be used to construct an automaton $\mathcal{A}_{\mathsf{tick}}$ that generates the ticks. Finally, we obtain $\mathcal{H}_M$ by taking the product of $\mathcal{A}_{\mathsf{tick}}$ with an automaton that encodes the instructions of the machine. For example, assuming the set of counters is $C = \{c, d\}$ the instruction (q, inc, c, q') is encoded by connecting a location ℓ_q to a location $\ell_{q'}$, synchronized with divisor gadgets that divide x_c by 16 and x_d by 4 (details omitted).

Undecidability with diagonal constraints. We now show that diagonal constraints also leads to undecidability. The result holds even if every variable has a positive, singular, fixed rate.

Theorem 3. *The time-bounded reachability problem is undecidable for LHA that use only singular, strictly positive, and fixed-rate variables.*

Proof. The proof is again by reduction from the halting problem for two-counter machines. We describe the encoding of the counters and the simulation of the instructions.

Given a counter c, we represent c via two auxiliary counters c_{bot} and c_{top} such that $v(c) = v(c_{\mathrm{top}}) - v(c_{\mathrm{bot}})$.

Incrementing and decrementing c are achieved by incrementing either c_{top} or c_{bot}. Zero-testing for c corresponds to checking whether the two auxiliary counters have the same value. Therefore, we do not need to simulate decrementation of a counter.

We encode the value of counter c_{bot} using two real-valued variables x and y, by postulating that $|x-y| = \frac{1}{2^{v(c_{\mathrm{bot}})}}$. Both x and y have rate $\dot{x} = \dot{y} = 1$ at all times and in all locations of the hybrid automaton. Incrementing c_{bot} now simply corresponds to halving the value of $|x-y|$. In order to achieve this, we use two real-valued variables z and w with rate $\dot{z} = 2$ and $\dot{w} = 3$.

All operations are simulated in 'rounds'. At the beginning of a round, we require that the variables x, y, z, w have respective value $\frac{1}{2^{v(c_{\mathrm{bot}})}}, 0, 0, 0$. We first explain how we merely *maintain* the value of c_{bot} throughout a round:

1. Starting from the beginning of the round, let all variables evolve until $x = z$, which we detect via a diagonal constraint. Recall that z evolves at twice the rate of x.
2. At that point, $x = \frac{2}{2^{v(c_{\mathrm{bot}})}}$ and $y = \frac{1}{2^{v(c_{\mathrm{bot}})}}$. Reset x and z to zero.
3. Now let all variables evolve until $y = z$, and reset y, z and w to zero. It is easy to see that all variables now have exactly the same values as they had at the beginning of the round. Moreover, the invariant $|x-y| = \frac{1}{2^{v(c_{\mathrm{bot}})}}$ is maintained throughout.

Note that the total duration of the above round is $\frac{2}{2^{v(c_{\mathrm{bot}})}}$. To *increment* c_{bot}, we proceed as follows:

1′. Starting from the beginning of the round, let all variables evolve until $x = w$. Recall that the rate of w is three times that of x.

2′. At that point, $x = \frac{1.5}{2^{v(c_{\mathrm{bot}})}}$ and $y = \frac{0.5}{2^{v(c_{\mathrm{bot}})}} = \frac{1}{2^{v(c_{\mathrm{bot}})+1}}$. Reset x, z, and w to zero.

3′. Now let all variables evolve until $y = z$, and reset y, z and w to zero. We now have $x = \frac{1}{2^{v(c_{\mathrm{bot}})+1}}$, and thus the value of $|x-y|$ has indeed been halved as required.

Note that the total duration of this incrementation round is $\frac{1}{2^{v(c_{\mathrm{bot}})}}$, where $v(c_{\mathrm{bot}})$ denotes the value of counter c_{bot} prior to incrementation.

Clearly, the same operations can be simulated for counter c_{top} (using further auxiliary real-valued variables). Note that the durations of the rounds for c_{bot} and c_{top} are in general different—in fact c_{bot}-rounds are never faster than c_{top}-rounds. But because they are powers of $\frac{1}{2}$, it is always possible to synchronize them, simply by repeating maintain-rounds for c_{bot} until the round for c_{top} has completed.

Finally, zero-testing the original counter c (which corresponds to checking whether $c_{\mathrm{bot}} = c_{\mathrm{top}}$) is achieved by checking whether the corresponding variables have the same value at the very beginning of a c_{bot}-round (since the c_{bot}- and c_{top}-rounds are then synchronized).

We simulate the second counter d of the machine using further auxiliary counters d_{bot} and d_{top}. It is clear that the time required to simulate one instruction of a two-counter machine is exactly the duration of the slowest round. Note however that since counters c_{bot}, c_{top}, d_{bot}, and d_{top} are never decremented, the duration of the slowest round is at most $\frac{2}{2^p}$, where p is the smallest of the initial values of c_{bot} and d_{bot}. If a two-counter machine has an accepting run of length m, then the total duration of the simulation is at most $\frac{2m}{2^p}$.

In order to bound this value, it is necessary before commencing the simulation to initialize the counters c_{bot}, c_{top}, d_{bot}, and d_{top} to a sufficiently large value, for example any number greater than $\log_2(m) + 1$. In this way, the duration of the simulation is at most 1.

Initializing the counters in this way is straightforward. Starting with zero counters (all relevant variables are zero) we repeatedly increment c_{bot}, c_{top}, d_{bot}, and d_{top} a nondeterministic number of times, via a self-loop. When each of these counters has value k, we can increment all four counters in a single round of duration $\frac{1}{2^k}$ as explained above. So over a time period of duration at most $\sum_{k=0}^{\infty} \frac{1}{2^k} = 2$ the counters can be initialized to $\lceil \log_2(m) + 1 \rceil$.

Let us now combine these ingredients. Given a two-counter machine M, we construct a hybrid automaton $\mathcal{H}_M$ such that M has an accepting run iff $\mathcal{H}_M$ has a run of duration at most 3 that reaches the final state Goal.

$\mathcal{H}_M$ uses the real-valued variables described above to encode the counters of M. In the initialization phase, $\mathcal{H}_M$ nondeterministically assigns values to the auxiliary counters, hence guessing the length of an accepting run of M, and then proceeds with the simulation of M. This ensures a correspondence between an accepting run of M and a time-bounded run of $\mathcal{H}_M$ that reaches Goal.

References

1. R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *TCS*, 138(1), 1995.
2. R. Alur and D. L. Dill. A theory of timed automata. *Th. Comp. Sci.*, 126(2):183–235, 1994.
3. F. Cassez and K. G. Larsen. The impressive power of stopwatches. In *Proc. of CONCUR*, LNCS 1877, pages 138–152. Springer, 1877.
4. J. Ferrante and C. Rackoff. A decision procedure for the first order theory of real addition with order. *SIAM J. Comput.*, 4(1):69–76, 1975.
5. G. Frehse. Phaver: algorithmic verification of hybrid systems past hytech. *Int. J. Softw. Tools Technol. Transf.*, 10:263–279, May 2008.
6. T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. In *Proc. of CAV*, LNCS 1254, pages 460–463. Springer, 1997.
7. T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *J. Comput. Syst. Sci.*, 57(1):94–124, 1998.
8. T. A. Henzinger and J.-F. Raskin. Robust undecidability of timed and hybrid systems. In *Proc. of HSCC*, LNCS 1790, pages 145–159. Springer, 2000.
9. M. L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall Inc., Englewood Cliffs, N.J., 1967. Prentice-Hall Series in Automatic Computation.
10. J. Ouaknine, A. Rabinovich, and J. Worrell. Time-bounded verification. In *Proc. of CONCUR*, LNCS 5710, pages 496–510. Springer, 2009.
11. J. Ouaknine and J. Worrell. Towards a theory of time-bounded verification. In *Proc. of ICALP (II)*, LNCS 6199, pages 22–37. Springer, 2010.