On the Automatic Verification of Systems with Continuous Variables and Unbounded Discrete Data Structures[*]

Ahmed Bouajjani[1**] Rachid Echahed[2] Riadh Robbana[1**]

[1] VERIMAG, Miniparc-Zirst, Rue Lavoisier, 38330 Montbonnot St-Martin, France.
email: Ahmed.Bouajjani@imag.fr, Riadh.Robbana@imag.fr
[2] LGI-IMAG, CNRS, BP53, 38041 Grenoble cedex, France.
email: Rachid.Echahed@imag.fr

Abstract. We address the verification problem of invariance properties for hybrid systems. We consider as general models of hybrid systems finite automata supplied with (unbounded) discrete data structures and continuous variables. We focus on the case of systems manipulating discrete counters and one pushdown stack, and on the other hand, constant slope continuous variables. The use of unbounded discrete data sturcture allows to consider systems with a powerful control, and to reason about important notions as the number of occurrences of events in some computation. The use of constant slope continuous variables allows to reason for instance about the time separating events and the durations of phases within some computation interval. We present decidability results for several subclasses of such models of hybrid systems; this provides automatic verification procedures for these systems.

1 Introduction

Hybrid systems consist of interacting continuous and discrete components. Typically, a hybrid system is obtained as a composition of some *dynamic system*, whose parameters (variables) change continuously with time progress, and some *controller*, which is a digital device (computer) that manipulates discrete data structures. Many examples of hybrid systems appear in various domains as in control of chemical processes, robotics, multimedia communication protocols, etc. The applications where hybrid systems are involved are in general *safety critical*, which means that their correctness is crucial, even vital. Thus, a challenging problem is to provide conception methods and analysis techniques for such systems.

The first step is to chose suitable models for hybrid systems. Roughly speaking, the nature of these systems suggests that their models should be combinations of models of dynamic systems (continuous variables + differential equation systems) with models of discrete machines (automata + discrete data structures).

[*] This paper is based on results previously presented in [BER94a] and [BER94b].

[**] Partially supported by the ESPRIT-BRA project REACT.

Hence, adequate models for hybrid system are obtained by considering finite control location graphs supplied with discrete data structures (counters, stacks, etc), and real valued variables that change continuously at each control location. The transitions between control locations are conditioned by constraints on the values (or configurations) of the (discrete and/or continuous) variables and data structures of the system; the execution of these transitions updates or resets the dicrete data structures and the continuous variables of the system.

Following this scheme, automata-based computational models of hybrid systems have been recently proposed as in [MMP92, NSY92, MP93, ACHH93, NOSY93]. Globally, these works consider systems with only continuous variables, and particularly, *linear hybrid systems* where the variables are constant slope (with constant derivatives) at each control location. Interesting special cases of such systems are *timed systems* (or timed graphs) [ACD90] (resp. *integrator systems* [Cer92, ACHH93, KPSY93]) corresponding to systems such that all their variables are *clocks* (resp. *integrators*), i.e., their rates are always 1 (resp. 0 or 1). The use of clocks allows to reason about the time elapsed in some computation interval, whereas integrators are useful for the reasoning about *durations* of phases, i.e., the accumulated time the computation is in some particular phase.

In this paper we consider more general models by introducing discrete data stuctures. There are several reasons that motivate the investigation of such extended models. First of all, these models allow to strengthen the control part of hybrid systems by considering systems manipulating powerful data structures. Moreover, the consideration of discrete variables (counters) allows to reason about interesting notions as the *number of occurrences* of events in some computation interval. Finally, it is interesting to explore the (simulation) links between systems with discrete variables and systems with continuous variables. When such links can be established, the analysis of hybrid systems can be reduced to the analysis of discrete systems, and reasoning on delays and durations becomes reasoning about number of occurrences of events.

The systems we investigate are linear hybrid systems extended with counters and a pushdown stack. We call these systems *pushdown counter linear hybrid systems*. Then, our aim is to identify subclasses of such systems that can be analysed automatically. The verification problem we tackle consists in deciding whether some system satisfies some given *invariance property* describing the set of correct (safe) behaviours. In fact, as we have mentioned earlier, safety requirements are the most important part of the specification of a hybrid system. Invariance properties are expressed as constraints on the values of the variables of the system. We distinguish two kinds of variables: *control variables* which are tested by the system to determine its transitions between control locations, and *observation variables* that are permanently updated without any influence on the dynamic of the system; observation variables are used to record relevant informations about the computations of the system, and to express invariance properties on its behaviours.

It is well known that invariance properties are duals of the *reachability properties* asserting that some (dangerous) configuration is reachable. Hence, the verification problem of invariance properties reduces to reachability problem in the considered models of hybrid systems.

Clearly, the reachability problem is in general undecidable for the systems we consider since it subsumes the halting problem of counter machines. Actually, the reachability problem is undecidable even for linear hybrid systems (without counters) since discrete counters can be simulated by continuous variables with rates -1, 0 or 1. Moreover, it has been shown that the reachability problem is undecidable even for integrator graphs [Cer92, HKPV95]. On the other hand, the reachability problem is decidable for timed graphs [ACD90], as well as for some restricted forms of integrator graphs [ACH93, KPSY93].

In this paper, we present gradually decidability results concerning several special cases of pushdown counter linear hybrid systems. We show that the verification problem is decidable for pushdown timed systems, and that this problem remains decidable when these systems are extended with monotonic control counters and observation counters. Moreover, we show that under some conditions, the verification problem is also decidable when these systems are extended with either one observation constant slope continuous variable, or by one control integrator. The general approach we adopt to establish these results is in two steps: the first one consists of reducing the reachability problem in the considered dense-time model to the reachability problem in some discrete structure. This step uses either a digitization or a partition (region graph) technique. Then, the second step consists of solving the reachability problem in the obtained discrete structure; this stucture being infinite in general due to the fact that we consider unbounded data structures.

The remainder of the paper is organized as follows. In Section 2 we introduce the pushdown counter linear hybrid systems. In Section 3 we introduce the invariance and reachability formulas we consider. In Section 4 we present the verification problem and its reduction to a reachability problem, give the existing results in the literature concerning this problem, describe our approach to solve it for the systems we consider, and then, in several subsections, we expose our different decidability results. Concluding remarks are given in Section 5.

2 Linear Hybrid Systems with a Stack and Counters

We introduce in this section models of hybrid systems, we call *pushdown counter linear hybrid systems* (PCLHS's for short), that are extensions of *linear hybrid systems* introduced in [NSY92, MMP92, ACHH93, NOSY93]. Roughly speaking, they consist of automata with a finitely many control locations, supplied with one pushdown stack, a finite set of discrete (integer valued) counters, and a finite set of (real valued) constant slope continuous variables, i.e., variables that change continuously at each control location with some integer rate.

2.1 Simple linear constraints

Let us start by introducing *simple linear constraints* that are used as enabling guards for the systems presented in the next subsection.

Let $\mathcal{V}$ be a set of real valued variables. A *simple linear constraint* over $\mathcal{V}$ is a boolean combination of constraints of the form $x \sim c$ where $x \in \mathcal{V}$, c is an integer constant ($c \in \mathbb{Z}$), and $\sim \in \{<, \leq\}$, or of the form $x \equiv_n c$ where $x \in \mathcal{V}$, $n \in \mathbb{N} - \{0\}$, and c is a natural constant ($c \in \mathbb{N}$). The symbols $<$ and $\leq$ represent the usual (strict and nonstrict) ordering relations over reals, and for every $n \in \mathbb{N} - \{0\}$, $\equiv_n$ stands for the relation between reals and positive reals defined by $\forall r_1 \in \mathbb{R}$, $\forall r_2 \in \mathbb{R}_{\geq 0}$, $r_1 \equiv_n r_2$ iff $0 \leq r_2 < n$, and $\exists q \in \mathbb{Z}$, $r_1 = q \cdot n + r_2$. For every variable x, we write simply $int(x)$ instead of $x \equiv_1 0$. Let $\mathcal{C}_{\mathcal{V}}$ be the set of simple linear constraints over $\mathcal{V}$.

A *valuation* over $\mathcal{V}$ is a function in $[\mathcal{V} \rightarrow \mathbb{R}]$. We say that a valuation ν is *integer* if $\forall x \in \mathcal{V}.\ \nu(x) \in \mathbb{Z}$. A satisfaction relation is defined as usual between valuations and constraints. Given valuation ν and $f \in \mathcal{C}_{\mathcal{V}}$, we denote by $\nu \models f$ the fact that ν satisfies f.

Given a valuation ν and a set of variables $X \subseteq \mathcal{V}$, we denote by $\nu[X \mapsto 0]$ the valuation which associates with each variable in X the value 0, and coincides with ν on all the other variables.

2.2 Definition

Let $\mathcal{P}$ be a finite set of atomic propositions and $\Sigma = 2^{\mathcal{P}}$. A pushdown counter linear hybrid system $\mathcal{H}$ over Σ consists of the following components:

- $\mathcal{L}$, a finite set of control locations,
- Π, a function in $[\mathcal{L} \rightarrow \Sigma]$, associating a set of atomic propositions with each location,
- $\mathcal{X}$, a finite set of continuous variables,
- ∂, a function in $[\mathcal{L} \times \mathcal{X} \rightarrow \mathbb{Z}]$, associating with each location ℓ and continuous variable x, a rate at which x changes continuously while the computation is at ℓ,
- $\mathcal{Y}$, a set of counters,
- Γ, a finite stack vocabulary,
- δ, a set of edges. Each edge is a tuple $e = (\ell, g, A, \gamma, \kappa, R, \ell')$ where $\ell, \ell' \in \mathcal{L}$ are the source and target locations, $g \in \mathcal{C}_{\mathcal{X} \cup \mathcal{Y}}$ is an enabling guard, $A \in \Gamma$ is the top of the stack, $\gamma \in \Gamma^*$ is the sequence of symbols replacing A, κ is a function in $[\mathcal{Y} \rightarrow \mathbb{Z}]$ associating with each counter an integer value that should be added to it, and $R \subseteq \mathcal{X} \cup \mathcal{Y}$ is the set of reseted variables.

We define the semantics of PCLHS's as sets of computation sequences. This is done by means of the notion of timed configuration.

A *configuration* of $\mathcal{H}$ is a tuple $\langle \ell, \lambda, \nu, \mu \rangle$ where ℓ is a control location, $\lambda \in \Gamma^*$ represents the stack (the first element from the left of λ being the top of the stack), ν is a valuation over $\mathcal{X}$, and μ is an integer valuation over $\mathcal{Y}$. A *timed configuration* of $\mathcal{H}$ is a tuple $\langle \ell, \lambda, \nu, \mu, t \rangle$ where $\langle \ell, \lambda, \nu, \mu \rangle$ is a configuration and $t \in \mathbb{R}_{\geq 0}$ is a time stamp.

To define the sequencing between timed configurations, we need to introduce some notations. Given a valuation ν over $\mathcal{X}$, a control location ℓ, and $t \in \mathbb{R}_{\geq 0}$, we denote by $[\nu + t]_\ell$ the valuation ν' such that $\forall x \in \mathcal{X}.\ \nu'(x) = \nu(x) + \partial(\ell, x) \cdot t$, i.e., the valuation of the variables obtained from ν by staying at location ℓ for an amount of time equal to t. Given a valuation μ over $\mathcal{Y}$, and a function $\kappa \in [\mathcal{Y} \to \mathbb{Z}]$, we denote by $\mu + \kappa$ the valuations $\mu' \in [\mathcal{Y} \to \mathbb{Z}]$ such that $\forall y \in \mathcal{Y}.\ \mu'(y) = \mu(y) + \kappa(y)$.

A *computation sequence* of $\mathcal{H}$ starting from a given configuration $\langle \ell, \lambda, \nu, \mu \rangle$ is an infinite sequence of timed configurations $\{\langle \ell_i, \lambda_i, \nu_i, \mu_i, t_i \rangle\}_{i \in \omega}$ such that

- $\langle \ell, \lambda, \nu, \mu, 0 \rangle = \langle \ell_0, \lambda_0, \nu_0, \mu_0, t_0 \rangle$,
- $\forall i \in \omega.\ t_{i+1} \geq t_i$,
- $\lim_{i \to \infty} t_i = \infty$, and
- $\forall i \in \omega$.
  - either $\ell_i = \ell_{i+1}$, $\lambda_{i+1} = \lambda_i$, $\mu_{i+1} = \mu_i$, and $\nu_{i+1} = [\nu_i + (t_{i+1} - t_i)]_{\ell_i}$,
  - or $t_i = t_{i+1}$ and $\exists (\ell_i, g, A, \gamma, \kappa, R, \ell_{i+1}) \in \delta.\ (\nu_i, \mu_i) \models g$, $\lambda_i = A \cdot \lambda'$, $\lambda_{i+1} = \gamma \cdot \lambda'$, $\nu_{i+1} = \nu_i[(R \cap \mathcal{X}) \mapsto 0]$, and $\mu_{i+1} = (\mu_i + \kappa)[(R \cap \mathcal{Y}) \mapsto 0]$.

Given a configuration α, we denote by $\mathcal{C}(\alpha, \mathcal{H})$ the set of computation sequences starting from α.

Now, we define a *trail* as an infinite sequence $\{\langle \ell_i, t_i \rangle\}_{i \in \omega}$, such that $t_0 = 0$, $\forall i \in \omega.\ t_{i+1} \geq t_i$, and $\lim_{i \to \infty} t_i = \infty$. Every computation sequence generates a trail obtained by abstracting from the stack and the valuations of the variables, i.e., the trail generated by the computation sequence $\sigma = \{\langle \ell_i, \lambda_i, \nu_i, \mu_i, t_i \rangle\}_{i \in \omega}$ is the sequence $\mathit{Trail}(\sigma) = \{\langle \ell_i, t_i \rangle\}_{i \in \omega}$.

Finally, we call *integer computation sequence* (resp. *integer trail*) any computation sequence $\{\langle \ell_i, \lambda_i, \nu_i, \mu_i, t_i \rangle\}_{i \in \omega}$ (resp. $\{\langle \ell_i, t_i \rangle\}_{i \in \omega}$) where all the t_i's are natural numbers. An *integer configuration* is any configuration $\langle \ell, \lambda, \nu, \mu \rangle$ where ν is an integer valuation over $\mathcal{X}$. Then, given an integer configuration α, we denote by $\mathcal{C}_{\mathbb{Z}}(\alpha, \mathcal{H})$ the set of integer computation sequences of $\mathcal{H}$ starting from α.

2.3 Particular cases

Different subclasses of PCLHS's can be considered depending on the nature of their variables, their guards, and the moments at which the variables are reset. We introduce in the sequel subclasses of PCLHS's we consider for the verification problem issue in the next sections.

First of all, we distinguish between variables that are involved in the dynamic of the systems, i.e., that are tested in the guards of the system, from those that are never tested by the system. We call the formers *control variables* and the latters *observation variables*. The aim of observation variables is to record informations about the evolution of the system; this is useful in the expression of invariance properties on the behaviours of the system.

We call *pushdown linear hybrid system*, PLHS for short, (resp. *counter linear hybrid system*, CLHS for short) a PCLHS without counters (resp. without stack). A *linear hybrid system* (LHS) is PCLHS without stack and counters. In each of these cases, we omit in the description of the system and its semantics the nonrelevant components.

It can be verified that every (P)(C)LHS can be transformed into an equivalent one such that each of its guards is of the form

$$\bigwedge_{x \in \mathcal{X} \cup \mathcal{Y}} ((a_x \sim_1^x x \sim_2^x b_x) \wedge \bigwedge_{i=1}^{m_x} x \approx_i^x c_i^x) \qquad (1)$$

where the $\sim_j^x$'s are in $\{<, \leq\}$, the $\approx_i^x$'s in $\{\equiv_n, \not\equiv_n\}$, the a_x's in $\mathbb{Z} \cup \{-\infty\}$, the b_x's in $\mathbb{Z} \cup \{+\infty\}$, and the c_i^x's in $\mathbb{N}$. We say that a (P)(C)LHS is *weak* if in every guard appearing in one of its edges (given by some expression as (1)), all the $\sim_j^x$'s are $\leq$'s (i.e., nonstrict inequalities), and all the $\approx_i^x$'s are $\equiv_n$'s.

A discrete (resp. continuous) variable x is *monotonic* if for every edge $e = (\ell, g, A, \gamma, \kappa, R, \ell')$ (resp. location ℓ), we have $\kappa(x) \geq 0$ (resp. $\partial(\ell, x) \geq 0$). Now, we say that a variable x is a *clock* (resp. *integrator*) if for every control location ℓ, we have $\partial(\ell, x) = 1$ (resp. $\partial(\ell, x) \in \{0, 1\}$). Then, a (pushdown) (counter) timed system, (P)(C)TS for short, is a (P)(C)LHS such that all its continuous variables are clocks, whereas a (pushdown) (counter) integrator system, (P)(C)IS for short, is a (P)(C)LHS such that all its variables are integrators.

A n-(P)(C)IS is a (P)(C)IS with any number of clocks and exactly n continuous variables that are not clocks but integrators. Moreover, a *single-clock* n-(P)(C)IS is a n-(P)(C)IS which has exactly one clock. We say also that a n-(P)(C)IS is *simple* if for every edge $(\ell, g, A, \gamma, \kappa, R, \ell')$, if R contains some clock, then the guard g contains a constraint of the form $x \equiv_m c$ or $x = c$, for some clock x (not necessarily in R). Intuitively, this condition on the moments of the resets of the clocks implies that all the clocks have always the same fractional parts (we suppose that their initial values are natural numbers) and hence, reach integer values simultaneously. Notice that this condition does not concern the n variables that are not clocks.

3 Invariance/Reachability Properties

Let $\mathcal{H}$ be a PCLHS over $\Sigma = 2^{\mathcal{P}}$. We use letters $P, Q, \ldots$ to range over the set of atomic propositions $\mathcal{P}$, and letters $f, g, \ldots$ to range over simple linear constraints over $\mathcal{X} \cup \mathcal{Y}$. Consider the set of formulas defined by the following grammar:

$$\varphi ::= P \mid f \mid \neg\varphi \mid \varphi \vee \varphi \mid \forall\Box\varphi$$

The semantics of these formulas is defined using a *satisfaction relation* $\models$ by the configurations of $\mathcal{H}$. For every configuration $\alpha = \langle \ell, \lambda, \mu, \nu \rangle$, the relation $\models$ is inductively defined by:

$\alpha \models P$ iff $P \in \Pi(\ell)$
$\alpha \models f$ iff $(\nu, \mu) \models f$
$\alpha \models \neg\varphi$ iff $\alpha \not\models \varphi$
$\alpha \models \varphi_1 \vee \varphi_2$ iff $\alpha \models \varphi_1$ or $\alpha \models \varphi_2$
$\alpha \models \forall\Box\varphi$ iff $\forall\sigma = \{\langle \ell_i, \lambda_i, \nu_i, \mu_i, t_i \rangle\}_{i \in \omega} \in \mathcal{C}(\alpha, \mathcal{H})$. $\forall i \in \omega$. $\langle \ell_i, \lambda_i, \nu_i, \mu_i \rangle \models \varphi$

An *integer satisfaction relation* $\models_{\mathbb{Z}}$ can be defined similarly between the integer configurations of $\mathcal{H}$ and the set of formulas above by considering the set of integer computations $\mathcal{C}_{\mathbb{Z}}(\alpha, \mathcal{H})$ instead of the whole set of computations $\mathcal{C}(\alpha, \mathcal{H})$.

Let us introduce the abbreviation $\exists\Diamond\varphi = \neg\forall\Box\neg\varphi$. An *invariance property* (resp. *reachability property*) is described by a formula of the particular form $\forall\Box\phi$ (resp. $\exists\Diamond\phi$) where ϕ is a boolean combination of atomic propositions and simple linear constraints.

Notice that, for every integer configuration α and every invariance formula φ, $\alpha \models \varphi$ implies that $\alpha \models_{\mathbb{Z}} \varphi$. On the other hand, for every reachability formula φ, $\alpha \models_{\mathbb{Z}} \varphi$ implies that $\alpha \models \varphi$. However, in both cases, the converse does not hold in general.

Using standard laws of the boolean connectives, together with the fact that any formula $\exists\Diamond(\phi_1 \vee \phi_2)$ is equivalent to $(\exists\Diamond\phi_1) \vee (\exists\Diamond\phi_2)$, it can be easily seen that every invariance formula is equivalent to the negation of a disjunction of reachability formulas of the form

$$\exists\Diamond(\pi \wedge \bigwedge_{x \in \mathcal{X} \cup \mathcal{Y}} ((a_x \sim_1^x x \sim_2^x b_x) \wedge \bigwedge_{i=1}^{m_x} x \approx_i^x c_i^x)) \tag{2}$$

where π is a boolean combination of atomic propositions, the $\sim_j^x$'s are in $\{<, \leq\}$, the $\approx_i^x$'s in $\{\equiv_n, \not\equiv_n\}$, the a_x's in $\mathbb{Z} \cup \{-\infty\}$, the b_x's in $\mathbb{Z} \cup \{+\infty\}$, and the c_i^x's in $\mathbb{N}$. We say that a reachability formula (2) is *weak* if all the inequalities it contains are nonstrict (i.e., all the $\sim_j^x$'s are $\leq$'s), and all the $\approx_i^x$'s are $\equiv_n$'s.

Finally, a *propositional* invariance (resp. reachability) formula is a formula of the form $\forall\Box\pi$ (resp. $\exists\Diamond\pi$) where π is a boolean combination of atomic propositions.

4 Verification Problem

The verification problem we consider is whether some given configuration α of a system $\mathcal{H}$ satisfies some given invariance formula φ, i.e., whether $\alpha \models \varphi$. This verification problem reduces to the verification problem of propositional reachability formulas which is equivalent to the control location reachability problem in PCLHS's. Indeed, as we have seen in the previous section, each invariance formula φ is the negation of a union of reachability formulas $\psi_1, \cdots, \psi_n$. Then, the verification problem $\alpha \models \varphi$ reduces the checking that for every $i \in \{1, \cdots, n\}$, $\alpha \models \psi_i$ does not hold. Deciding whether $\alpha \models \psi_i$, with $\psi_i = \exists\Diamond(\pi \wedge \bigwedge_{x \in \mathcal{X} \cup \mathcal{Y}}((a_x \sim_1^x x \sim_2^x b_x) \wedge \bigwedge_{j=1}^{m_x} x \approx_j^x c_j^x))$ reduces to deciding whether $\alpha \models \exists\Diamond$ at-*target*, where at-*target* is true only at a special location *target* in the system obtained by extending $\mathcal{H}$ with *target* and edges that lead to it from all the locations ℓ such that $\Pi(\ell) = \pi_i$, and that are guarded by $\bigwedge_{x \in \mathcal{X} \cup \mathcal{Y}}((a_x \sim_1^x x \sim_2^x b_x) \wedge \bigwedge_{j=1}^{m_x} x \approx_j^x c_j^x)$.

It is well known that the (control location) reachability problem is decidable for timed systems [ACD90]. On the other hand, this problem can be shown to be undecidable in general for LHS's. Indeed, the halting problem of n-counter machines can be reduced to the reachability problem of single-clock n-LHS's

[KPSY93]: each counter can be simulated by a continuous variable with rates in $\{-1, 0, 1\}$, the additional clock is used to let a variable increase (resp. decrease) for exactly one time unit at each incrementation (resp. decrementation) of the corresponding counter. It can be seen that this reduction uses weak and simple single-clock n-LHS's. Thus, the reachability problem is undecidable for simple weak single-clock 2-LHS's. Concerning integrator systems, their reachability problem has been shown to be undecidable in [Cer92]. Furthermore, it has been shown that even for weak 1-IG's, the reachability problem is undecidable [HKPV95]. Decidable particular cases of linear hybrid systems has been identified as in [ACH93, KPSY93, PV94]. In particular, it is shown in [ACH93] (resp. [KPSY93]) that the verification problem of reachability formulas for timed systems (resp. weak timed systems) with one additional observation integrator (resp. observation variable) is decidable.

In this section, we present gradually decidability results concerning several special cases of linear hybrid systems with a stack and counters. We extend the results in [ACD90] to the case of a system with a stack, and any number of observation counters and monotonic control counters. Then, we show that the reachability problem is also decidable for weak such systems augmented by one continuous observation variable (extending the result given in [KPSY93]), as well as for for weak simple systems augmented by one additional control integrator.

The general principle we adopt for proving these results is in two steps. The first step consists of reducing the reachability problem in the original dense system to a reachability problem in some discrete structure. Then, the second step consists of solving the reachability problem in the discrete structure, which is infinite in general due to the consideration of nonbounded discrete data stuctures (stack and counters).

The first step is carried out using either a *digitization* or a *partition* technique. Digitization consists of proving that the reachability problem in a dense-time system can be solved by considering only the integer computations of the system. This approach has been used for instance in [HMP92] to decide the verification problem for a class of timed linear-time properties and weak timed systems. On the other hand, the partition technique consists of finding a countable partition of the dense space of valuations of the continuous variables, such that configurations with the same control location and valuations in the same class (region) satisfy the same reachability properties. This approach has been introduced in [ACD90] to decide the verification problem for the logic TCTL and timed systems.

In all the cases we consider, the reachability problem can be reduced, using either digitization or partition, to a reachability problem in a pushdown automaton, and in some case to a pushdown automaton with observation counters. This reachability problem can be solved as a nonemptiness problem of context-free languages, or as an integer linear programming problem when observation counters are considered.

4.1 Pushdown Timed Systems

Let $\mathcal{H}$ be a pushdown timed system. We show in the sequel that the verification problem of reachability formulas for these systems is decidable. As we have said in the beginning of Section 4, the verification problem of reachability formulas reduces to the verification problem of propositional reachability formulas, which is equivalent to solve a control location reachability problem. More precisely, deciding whether some reachability formula is satisfied starting from some configuration α of $\mathcal{H}$ is equivalent to decide whether some special location *target* is reachable from α in an appropriately extended system. To solve this reachability problem, we extend the partition (region graph) method introduced in [ACD90]. Indeed, we consider a finite index equivalence relation on the set of valuations of clocks which preserves the propositional reachability properties. This allows to reduce the control location reachability problem on a dense system to a control location reachability problem on a discrete structure. This discrete structure is however infinite since we consider a system with a stack. Then, we show that our reachability problem can be reduced to a control location reachability problem in a pushdown automaton, which is equivalent to the nonemptiness problem of context-free languages.

Let us start by considering a partition of the set of valuations of $\mathcal{X}$; let $\mathcal{E}$ denotes this set, i.e., $\mathcal{E} = [\mathcal{X} \rightarrow \mathbb{R}]$. In [ACD90], an equivalence relation $\simeq$ on $\mathcal{E}$ is introduced so that from any pair of configurations of a (finite) timed system having the same control location and $\simeq$-equivalent valuations, the set of reachable control locations are the same. The basic idea behind the definition of this equivalence is that the exact value of every clock x is only relevant while it is less than the greater constant which is compared with it in the system, say c_x. Indeed, beyond c_x, all the values the clock x can take satisfy the same guards appearing in the system.

So, let us recall the definition of the relation $\simeq$. For every ν, $\nu' \in \mathcal{E}$, we have $\nu \simeq \nu'$ iff the following conditions hold:

- $\forall x \in \mathcal{X}$, if $\nu(x) \leq c_x$, then $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ and $fract(\nu(x)) = 0$ iff $fract(\nu'(x)) = 0$,
- $\forall x, y \in \mathcal{X}$, if $\nu(x) \leq c_x$ and $\nu(y) \leq c_y$, then $fract(\nu(x)) \leq fract(\nu(y))$ iff $fract(\nu'(x)) \leq fract(\nu'(y))$.

Actually, to take into account the fact that $\mathcal{H}$ may contain contraints of the form $x \equiv_n c$, we consider the equivalence $\cong$ which is defined as the refinement of the equivalence $\simeq$ w.r.t. all the (finite-index) $\equiv_n$'s appearing in $\mathcal{H}$. Then, it is easy to verify that the result of [ACD90] mentioned above holds also for pushdown timed systems.

Lemma 4.1 *Let ν, $\nu' \in \mathcal{E}$. Then, $\nu \cong \nu'$ implies that, for every propositional reachability formula φ, every $\ell \in \mathcal{L}$, and every $\lambda \in \Gamma^*$, $\langle \ell, \lambda, \nu \rangle \models \varphi$ iff $\langle \ell, \lambda, \nu' \rangle \models \varphi$.*

Lemma 4.1 allows to reduce the (control location) reachability problem in a dense configuration graph to a reachability problem in a discrete (countable) structure. This structure corresponds to what is called *region graph* in [ACD90] in the case of timed systems. This region graph is obtained by considering the quotient of the configuration graph of $\mathcal{H}$ w.r.t. the equivalence $\cong$. Let us see how the region graph is defined. Let $[\mathcal{E}]$ be the quotient set of $\mathcal{E}$ under the relation $\cong$. The set $[\mathcal{E}]$ can be supplied by a successor function *succ* between equivalence classes which captures time progress. The function *succ* is defined in the following manner: For every $\nu_1, \nu_2 \in \mathcal{E}$, $succ([\nu_1]) = [\nu_2]$ iff $\nu_1 \not\cong \nu_2$, and $\exists t \in I\!\!R^+$ such that $\nu' \cong \nu + t$, and $\forall t' \in I\!\!R^+$, $0 \leq t' < t$, either $\nu + t' \cong \nu$ or $\nu + t' \cong \nu'$. Now, we define a *region* as a triplet $\langle \ell, \lambda, [\nu] \rangle$ where $\ell \in \mathcal{L}$, $\lambda \in \Gamma^*$, and $[\nu] \in [\mathcal{E}]$. Let Reg be the set of such regions. Then, the region graph of $\mathcal{H}$, denoted $\mathcal{R}(\mathcal{H})$, is defined as the graph $\langle Reg, Edg \rangle$ where the vertex set is Reg defined above, and the set of edges Edg is the smallest set such that:

- Each vertex (region) $\langle \ell, \lambda, [\nu_1] \rangle$ has an edge to $\langle \ell, \lambda, succ([\nu_2]) \rangle$,
- Each vertex $\langle \ell, A \cdot \lambda, [\nu_1] \rangle$ has an edge to $\langle \ell', \gamma \cdot \lambda, [\nu_2[R \mapsto 0]] \rangle$ for each edge $(\ell, g, A, \lambda, R, \ell') \in \delta$ such that $\nu_1 \models g$.

Clearly, the region graph $\mathcal{R}(\mathcal{H})$ is infinite due to the use of the stack. However, recall that the reachability problem we consider on this graph is whether some configuration with a particular control location *target* is reachable starting from some configuration $\alpha = \langle \ell, \lambda, \nu \rangle$. In other words, the value of the stack is not relevant in the target configurations; only the fact that they correspond to the control location *target* is relevant. Then, we can reduce this reachability problem to a control location reachability problem in the input-free pushdown automaton $\mathcal{A}$ having $\mathcal{R}(\mathcal{H})$ as configuration graph. The automaton $\mathcal{A}$ is such that:

- the stack vocabulary is Γ,
- the set of control locations is $\mathcal{L}' = \mathcal{L} \times \mathcal{E}$,
- the set of final control locations is the set of locations $\langle target, \varepsilon \rangle \in \mathcal{L}'$, for any $\varepsilon \in [\mathcal{E}]$,
- the transition relation is given by the set of edges $(\langle \ell_1, [\nu_1] \rangle, A, \gamma, \langle \ell_2, [\nu_2] \rangle)$ such that
  - either $\ell_1 = \ell_2$, $\gamma = A$, and $succ([\nu_1]) = [\nu_2]$,
  - or $\exists (\ell_1, g, A, \gamma, R, \ell_2) \in \delta$ such that $\nu_1 \models g$, and $[\nu_2] = [\nu_1[R \mapsto 0]]$.

Then, it can be easily seen that the location *target* in $\mathcal{H}$ is reachable from α iff the automaton $\mathcal{A}$ accepts some word starting from the location $\langle \ell, [\nu] \rangle$ with a stack equal to λ. This problem is equivalent to the nonemptiness problem of context-free languages. Hence, we obtain the following decidability result.

Theorem 4.1 *The verification problem of reachability formulas for PTS's is decidable.*

4.2 Pushdown Timed Systems + Counters

In this subsection, we investigate the verification problem of reachability properties for extensions of pushdown timed systems with some kind of counters. We start by the simple case of PTS's with monotonic control counters, and then, we consider the case of PTS's extended with observation counters. We show that for these systems, the verification of reachability formulas is decidable.

PTS's + Monotonic Control Counters Let $\mathcal{H}$ be a PTS's augmented by monotonic control counters. Actually, in this case, we can handle the counters in the same manner as clocks. Indeed, since a monotonic counter never decreases, its exact value is only relevant while it is less than the greater constant which is compared with it in the system. Then, in the same manner as in the subsection 4.1, we can define a finite-index equivalence relation $\cong$ on valuations of counters and clocks so that configurations with the same location, the same stack, and $\cong$-equivalent valuations, satisfy the same propositional reachability formulas.

Let us give the definition of the relation $\cong$. For each variable y of the system $\mathcal{H}$, let c_y denotes the greater constant which is compared with y in the guards of $\mathcal{H}$. Then, consider the equivalence $\simeq$ such that $(\nu, \mu) \simeq (\nu', \mu')$ iff the following conditions hold:

- $\forall x \in \mathcal{X}$, if $\nu(x) \leq c_x$, then $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ and $fract(\nu(x)) = 0$ iff $fract(\nu'(x)) = 0$,
- $\forall x, y \in \mathcal{X}$, if $\nu(x) \leq c_x$ and $\nu(y) \leq c_y$, then $fract(\nu(x)) \leq fract(\nu(y))$ iff $fract(\nu'(x)) \leq fract(\nu'(y))$,
- $\forall z \in \mathcal{Y}$, if $\mu(z) \leq c_z$, then $\mu(z) = \mu'(z)$.

As in the subsection 4.1, we define the relation $\cong$ as the refinement of the equivalence $\simeq$ defined above w.r.t. all the $\equiv_n$'s that appear in the guard of $\mathcal{H}$. Therefore, we can reduce the reachability problem in $\mathcal{H}$ to a reachability problem in a pushdown automaton, and decide it as a nonemptiness problem of a context-free language. Then, we obtain the following result.

Theorem 4.2 *The verification problem of reachability formulas for PTS's with monotonic counters is decidable.*

PTS's + Observation Counters We consider now the verification problem of reachability formulas for PTS's with observation counters. Let $\mathcal{H}$ be such a system, $\alpha = \langle \ell, \lambda, \nu, \mu \rangle$ a configuration of $\mathcal{H}$, and φ a reachability formula given by:

$$\exists\Diamond(\pi \wedge f \wedge \bigwedge_{y \in \mathcal{Y}} ((a_y \sim_1^y y \sim_2^y b_y) \wedge \bigwedge_{i=1}^{m_y} y \approx_i^y c_i^y)) \tag{3}$$

where f involves only clocks, and suppose that we want to decide whether $\alpha \models \varphi$.

We have shown in the subsection 4.1 that if we consider only the reachability formula $\exists\Diamond(\pi \wedge f)$, then the verification problem reduces to check the nonemptiness of the language accepted by an input-free pushdown automaton $\mathcal{A}$ constructed from the system $\mathcal{H}$. In the present case, the verification problem reduces to a *constrained nonemptiness problem* which consists of checking whether the automaton $\mathcal{A}$ accepts some sequence which satisfies the constraints on the observation counters involved in φ. In other words, we have to decide whether there exists some sequence accepted by $\mathcal{A}$ such that, for each observation counter y, if we accumulate its initial value (i.e., $\mu(y)$) with all the effects of the taken transitions in the sequence, then the obtained value, say v_y, should satisfy $(a_y \sim_1^y v_y \sim_2^y b_y) \wedge \bigwedge_{i=1}^{m_y} v_y \approx_i^y c_i^y$. The effect of each transition of the pushdown automaton is either null if the transition corresponds to time progress, or given by the function κ of the system $\mathcal{H}$ if it corresponds to the execution of a guarded transition of $\mathcal{H}$.

To solve this constrained nonemptiness problem, we actually transform the automaton $\mathcal{A}$ to an equivalent context-free grammar $\mathcal{G} = (\Sigma, N, Prod, S)$ where $\Sigma = \emptyset$ is the vocabulary, N is the set of nonterminals, $Prod$ is the set of productions, and S is the starting symbol. Notice that all the productions are of the form $A \rightarrow \gamma$ where $\gamma \in N^*$. Hence, if $L(\mathcal{G}) \neq \emptyset$, then $L(\mathcal{G}) = \{\epsilon\}$ where ϵ is the empty sequence. The transformation we use is the standard one given in [HU79]. Each production (may be several) in $\mathcal{G}$ is obtained in some way from a transition in the automaton $\mathcal{A}$. Then, we can define a function κ' which describes the effect of the application of the productions in $\mathcal{G}$ on the observation variables, i.e., κ' associates with each production p of $\mathcal{G}$ and each observation counter y an integer which should be added to y at each application of p.

In the sequel we denote by $(\Longrightarrow)$ the derivation relation in the grammar $\mathcal{G}$, and by $(\overset{*}{\Longrightarrow})$ its reflexive-transitive closure. We use subscripts to precise the set of productions or the sequences of productions used in the derivation.

Then, our constrained nonemptiness problem consists of deciding whether there exists some derivation σ in $\mathcal{G}$ given by

$$S \Longrightarrow_{p_1} \lambda_1 \cdots \Longrightarrow_{p_n} \lambda_n \Longrightarrow_{p_{n+1}} \epsilon \tag{4}$$

where the pi_i's are in $Prod$ and the λ_i's are nonempty sequences of nonterminals (in N^+), and such that $\bigwedge_{y \in \mathcal{Y}}((a_y \sim_1^y v_y \sim_2^y b_y) \wedge \bigwedge_{i=1}^{m_y} v_y \approx_i^y c_i^y)$ where $v_y = \mu(y) + \sum_{j=1}^{n+1} \kappa'(p_j, y)$.

We show in the sequel that this decision problem can be solved as an integer linear programming problem, i.e., we construct a linear formula Ω which is satisfiable if and only if a derivation like σ above exists.

Let us define the sets of variables that are involved in the formula Ω. We associate with every observation counter y a variable v_y which stands for its value at the end of σ. Moreover, with every production $p \in Prod$ we associate a variable w_p which stands for the number of applications of p in σ.

First of all, we have to express that obviousely, the v_y's and the w_p's are positive by the formula POS:

$$(\bigwedge_{y \in \mathcal{Y}} v_y \geq 0) \wedge (\bigwedge_{p \in Prod} w_p \geq 0)$$

Then, for every observation counter y, the value of v_y is linked with the values of the w_p's by the formula ACC_y:

$$v_y = \mu(y) + \sum_{p \in Prod} \kappa'(p, y) \cdot w_p$$

Then, we have to consider the constraints on the observation counters involved in the formula φ (3). Hence, we define for every counter y the linear constraint COND_y:

$$(a_y \sim_1^y v_y \sim_2^y b_y) \wedge \bigwedge_{i=1}^{m_y} v_y \approx_i^y c_i^y$$

Now, let us define the constraints on the w_p's. First of all, These constraints must express the fact that any occurrence of a nonterminal appearing along σ must be reduced so that we get the empty sequence ϵ at the end of σ. Thus, in the derivation σ, for any nonterminal A, the number of the reductions of A, i.e., applications of productions p having A as left-hand side (A-productions), must be equal to the number of the A-introductions, i.e., the number of the occurrences of A in the right-hand sides of the applied productions, plus one if A is equal to S.

For every production $p : A \rightarrow \gamma$, let us denote by $lhs(p)$ and $rhs(p)$ respectively the left-hand side and the right-hand side of p, i.e., $lhs(p) = A$ and $rhs(p) = \gamma$. Given $\gamma \in N^*$, and $A \in N$, we denote by $|\gamma|_A$ the number of occurrences of A in γ.

Then, for every nonterminal A we define the formula REDUCT_A:

$$\sum_{p \in Prod} |lhs(p)|_A \cdot w_p = |S|_A + \sum_{p \in Prod} |rhs(p)|_A \cdot w_p$$

Another fact we have to express is that a nonterminal is reduced if and only if it is reachable by applying productions participating to the derivation σ. Indeed, consider some production $p = B \rightarrow B$ and suppose that B never appears in σ. Then, we can asssign any positive value to w_p and still satisfy the constraints REDUCT_A for every A. This is not acceptable since the values v_y's calculated using the constraints ACC_y's will not necessarily correspond to values that can be obtained from existing derivation in $\mathcal{G}$. Thus, we must express the fact that for any nonterminal A, there exists some A-production p with $w_p > 0$ if and only if A is S or it appears in the λ_i's along σ.

Let us introduce first some notation. We say that a sequence of productions $\theta \in Prod^*$ is *elementary* if all its productions apply to different nonterminals. Given a nonterminal A we define Θ_A to be the set of elementary sequences θ on $Prod$ such that $\exists \gamma, \gamma' \in N^*$, $S \overset{*}{\Longrightarrow}_\theta \gamma \cdot A \cdot \gamma'$. Notice that the set Θ_A is finite.

Then, we define for every nonterminal A the formula REACH_A:

$$\sum_{p \in Prod} |lhs(p)|_A \cdot w_p > 0 \Leftrightarrow \bigvee_{\theta \in \Theta_A} \bigwedge_{p \in \theta} w_p > 0.$$

Finally, the formula Ω is defined by:

$$\mathsf{POS} \ \wedge \ (\bigwedge_{A \in N} \mathsf{REDUCT}_A \wedge \mathsf{REACH}_A) \ \wedge \ (\bigwedge_{y \in \mathcal{Y}} \mathsf{ACC}_y \wedge \mathsf{COND}_y)$$

Since the satisfiability problem of linear formulas on integers is deciable, we obtain the following result.

Theorem 4.3 *The verification problem of reachability formulas for PTS's with observation counters is decidable.*

Actually, it can be shown that the verification problem of reachability formulas is also decidable for PTS's with monotonic control counters and observation counters. For that, it suffices to contruct the grammar $\mathcal{G}$ from the pushdown automaton used for the justification of Theorem 4.2 (see Section 4.2).

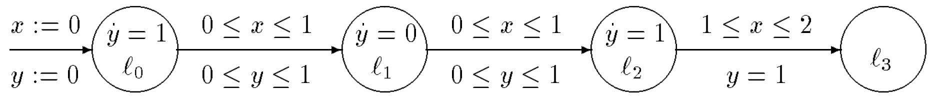
4.3 Pushdown Timed Systems + Continuous Variables

This subsection concerns extensions of pushdown timed systems with continuous variables. We consider either weak PTS's with one observation control variable or weak simple PTS's with one control integrator. We show that for these systems, the verification of weak reachability formulas is decidable. For that, we use digitization techniques to reduce their verification problems to the reachability problem in pushdown automata with either one observation counter or one monotonic control counter, respectively.

Digitization We introduce a notion of digitization which extends the notion defined in [HMP92]. We call *digitization quantum* any value $\epsilon \in [0,1)$. We call *digitization sequence* any infinite sequence of digitization quanta $\vec{\epsilon} = \{\epsilon_i\}_{i\in\omega}$. In particular, we say that the digitization sequence is *uniform* if all the ϵ_i's are equal. Given some real time amount $t \in \mathbb{R}_{\geq 0}$, we define, for every digitization quantum $\epsilon \in [0,1)$, the integer time amount $[t]_\epsilon =$ if $t \leq \lfloor t \rfloor + \epsilon$ then $\lfloor t \rfloor$ else $\lceil t \rceil$.

Now, consider a trail $\rho = \{\langle \ell_i, t_i\rangle\}_{i\in\omega}$, and a digitization sequence $\vec{\epsilon} = \{\epsilon_i\}_{i\in\omega}$. Then, the *digitization* of ρ w.r.t. $\vec{\epsilon}$ is the integer trail $[\rho]_{\vec{\epsilon}} = \{\langle \ell_i, [t_i]_{\vec{\epsilon}(t_i)}\rangle\}_{i\in\omega}$ where $\forall t \in \mathbb{R}_{\geq 0}.\ \vec{\epsilon}(t) = \epsilon_{\lfloor t \rfloor}$. A digitization of some trail is *uniform* if it uses a uniform digitization sequence.

The notion of digitization defined in [HMP92] corresponds actually to the notion of uniform digitization. This notion is useful for the reasoning about weak timed systems, since for each such a system, the set of trails generated by its computation sequences is closed under uniform digitization. This notion of uniform digitization is also useful for the case of weak timed systems with one observation continuous variable as it has been shown in [KPSY93]; we extend this fact to pushdown timed systems. However, this notion does not allow to reason about integrator systems. Consider for instance the weak simple 1-IG $\mathcal{H}$, with a clock x and an integrator y, which is represented by the following picture:



and let σ be the computation sequence $\langle \ell_0, (x=0, y=0), 0\rangle\langle \ell_0, (x=0.5, y=0.5), 0.5\rangle\langle \ell_1, (x=0.5, y=0.5), 0.5\rangle\langle \ell_1, (x=1, y=0.5), 1\rangle\langle \ell_2, (x=1, y=0.5), 1\rangle\langle \ell_2, (x=1.5, y=1), 1.5\rangle\langle \ell_3, (x=1.5, y=1), 1.5\rangle$.

Then, it can be seen that the system $\mathcal{H}$ above has no integer computation that generates some uniform digitization of $Trail(\sigma)$. Indeed, if we take any uniform digitization sequence such that all its elements are equal to some $\epsilon \in [0, 0.5)$, then we obtain the integer trail

$$\rho_1 = \langle \ell_0, 0\rangle\langle \ell_0, 1\rangle\langle \ell_1, 1\rangle\langle \ell_1, 1\rangle\langle \ell_2, 1\rangle\langle \ell_2, 2\rangle\langle \ell_3, 2\rangle$$

whereas if we consider that $\epsilon \in [0.5, 1)$, we obtain the integer trail

$$\rho_2 = \langle \ell_0, 0\rangle\langle \ell_0, 0\rangle\langle \ell_1, 0\rangle\langle \ell_1, 1\rangle\langle \ell_2, 1\rangle\langle \ell_2, 1\rangle\langle \ell_3, 1\rangle$$

Then, it can be verified that these trails cannot be generated by computation sequences, since from the time stamps of ρ_1 (resp. ρ_2) we can deduce that the transition from ℓ_2 to ℓ_3 is executed when the value of y is 2 (resp. 0), which falsifies the guard $y = 1$.

However, the system $\mathcal{H}$ has two integer computations generating trails that are *nonuniform* digitizations of $Trail(\sigma)$. These digitizations are

$$\rho_3 = \langle \ell_0, 0\rangle\langle \ell_0, 0\rangle\langle \ell_1, 0\rangle\langle \ell_1, 1\rangle\langle \ell_2, 1\rangle\langle \ell_2, 2\rangle\langle \ell_3, 2\rangle$$

and

$$\rho_4 = \langle \ell_0, 0\rangle\langle \ell_0, 1\rangle\langle \ell_1, 1\rangle\langle \ell_1, 1\rangle\langle \ell_2, 1\rangle\langle \ell_2, 1\rangle\langle \ell_3, 1\rangle$$

the trail ρ_3 (resp. ρ_4) is obtained when the time values in the interval $[0, 1]$ are digitized w.r.t. some $\epsilon_1 \in [0.5, 1)$ (resp. $\epsilon_1 \in [0, 0.5)$), and the time values in the interval $[1, 2]$ w.r.t. some $\epsilon_2 \in [0, 0.5)$ (resp. $\epsilon_2 \in [0.5, 1)$).

PTS's + One Continuous Observation Variable Let $\mathcal{H}$ be a weak PTS extended with one continuous observation variable; let y be this variable. We show in the sequel that the verification of weak reachability formulas on such a system is decidable (when the starting configuration is integer). For that, we use the digitization result proved in [KPSY93] which says that if a constraint $y \leq c$ is satisfied along some trail ρ at some position k, and assuming that its initial value is 0, then $y \leq c$ is also satisfied along some uniform digitization of ρ at the same position k.

Lemma 4.2 *Let $\rho = \{\langle \ell_i, t_i\rangle\}_{i\in\omega}$ be a trail. Then, if there is some $k \in \omega$ such that $\sum_{i=0}^{k-1} \partial(\ell_i, y) \cdot (t_{i+1} - t_i) \leq c$, then there exists some digitization quantum $\epsilon \in [0, 1)$ such that $\sum_{i=0}^{k-1} \partial(\ell_i, y) \cdot ([t_{i+1}]_\epsilon - [t_i]_\epsilon) \leq c$.*

It has been shown in [HMP92] that for weak timed systems, for every computation sequence σ, and every uniform digitization sequence $\vec{\epsilon}$, there exists a computation sequence σ' such that $[Trail(\sigma)]_{\vec{\epsilon}} = Trail(\sigma')$. Hence, by Lemma 4.2, it can be deduced that for timed systems with one observation variable y, for every computation sequence σ which starts from some integer configuration, and where the constraint $y \leq c$ is satisfied at some position k, there exist an integer computation σ' and a digitization sequence $\vec{\epsilon}$ such that $[Trail(\sigma)]_{\vec{\epsilon}} = Trail(\sigma')$, and $y \leq c$ is satisfied in σ' at position k. This fact can be extended straightforwardly to systems with a stack and counters, since digitization leads to computations that perform the same sequence of operations on these discrete data stuctures than the original one. Then, we obtain the following fact.

Proposition 4.1 *Let $\mathcal{H}$ be a weak PTS with one additional continuous observation variable, α an integer configuration of $\mathcal{H}$, and $\sigma \in \mathcal{C}(\alpha, \mathcal{H})$. Then, there exists a digitization sequence $\vec{\epsilon}$, and $\sigma' \in \mathcal{C}_{\mathbb{Z}}(\alpha, \mathcal{H})$, such that $[Trail(\sigma)]_{\vec{\epsilon}} = Trail(\sigma')$.*

By Proposition 4.1, we can deduce that the verification of reachability formulas for weak PTS's with one continuous observation variable can be solved by considering only integer valuations of the continuous variables (clocks + the observation variable y). Then, all the clocks become monotonic control counters whereas the variable y becomes an observation counter. Hence, using a similar reasoning as in Section 4.2, the verification of a reachability formula on $\mathcal{H}$ reduces to a reachability problem in a pushdown automaton with constraints on observation counters, which is solvable as an integer linear programming problem.

Theorem 4.4 *Let $\mathcal{H}$ be a weak PTS with one additional continuous observation variable, α an integer configuration of $\mathcal{H}$, and φ a weak reachability formula. Then, the verification problem $\alpha \models \varphi$ is decidable.*

PTS's + One Control Integrator Let $\mathcal{H}$ be a weak simple 1-PIG. We prove in the sequel that given any integer configuration α of $\mathcal{H}$, for every computation sequence $\sigma \in \mathcal{C}(\alpha, \mathcal{H})$, there exists an integer computation sequence $\sigma' \in \mathcal{C}_{\mathbb{Z}}(\alpha, \mathcal{H})$ and a digitization sequence $\vec{\epsilon}$ such that $[Trail(\sigma)]_{\vec{\epsilon}} = Trail(\sigma')$. Let us give an informal description of the proof.

Assume without loss of generality that the computation $\sigma = \{\ell_i, \lambda_i, \nu_i, \mu_i, t_i\rangle$ is such that for every $k \in \omega$, there exists some index i_k, such that $t_{i_k} = k$. Indeed, each computation sequence can be *completed* into a computation satisfying this condition, and which visits the same locations. Let $\rho = Trail(\sigma)$.

Then, remember that since $\mathcal{H}$ is a simple system, all its clocks have the same fractional parts along the sequence σ, and notice that this fractional part is equal at each timed configuration of σ to the fractional part of the time stamp t_i. Hence, since any digitization associates with each noninteger time stamp t either $\lfloor t \rfloor$ or $\lceil t \rceil$, and keeps unchanged integer time stamps, and since $\mathcal{H}$ is a weak system, then it suffices to consider the effect of the digitization on the integrator y. The basic idea is to prove that there exists a digitization sequence $\vec{\epsilon}$ (i.e., a digitization quantum ϵ_u for each time interval $[u, u+1]$) such that, at every position k where a guarded transition is taken by σ, the distance between the (original) value of y, and its integer value according to $[\rho]_{\vec{\epsilon}}$, is strictly less than 1. Let us call this distance the *error* on y at position k. Indeed, if such a digitization sequence exists, then, since $\mathcal{H}$ is a weak system, the integer trail $[\rho]_{\vec{\epsilon}}$ respects all the guards of the transitions fired by σ, and hence it corresponds to a computation sequence of $\mathcal{H}$. We prove the existence of $\vec{\epsilon}$ by showing that for every time interval $\mathcal{I}_u = [u, u+1]$, if it is possible to digitize ρ in all the time intervals before $\mathcal{I}_u$ in such a manner that the error on y at the beginning of $\mathcal{I}_u$ is < 1, then it is always possible to choose a digitization quantum ϵ_u such that the error on y remains < 1 at each position in $\mathcal{I}_u$. Then, we obtain the following digitization result (see the detailed proof in the appendix).

Proposition 4.2 *Let $\mathcal{H}$ be a weak simple* 1-*PIS, α an integer configuration of $\mathcal{H}$, and $\sigma \in \mathcal{C}(\alpha, \mathcal{H})$. Then, there exists a digitization sequence $\vec{\epsilon}$, and an integer computation $\sigma' \in \mathcal{C}_{\mathbb{Z}}(\alpha, \mathcal{H})$, such that $[Trail(\sigma)]_{\vec{\epsilon}} = Trail(\sigma')$.*

By Proposition 4.2, the verification problem of weak reachability formulas for weak simple 1-PIS's can be solved by considering only integer valuations of the continuous variables. Hence, all these continuous variables can be seen as monotonic control counters, and then, the verification problem of reachability formulas reduces to the reachability problem in pushdown automata with monotonic control counters, which is decidable (see Section 4.2).

Theorem 4.5 *Let $\mathcal{H}$ be a weak simple* 1-*PIS, α an integer configuration of $\mathcal{H}$, and φ a weak reachability formula. Then, the verification problem $\alpha \models \varphi$ is decidable.*

5 Conclusion

We have investigated the verification of invariance/reachability properties for hybrid systems modeled by finite control location graphs supplied with unbounded discrete data structures and constant slope continuous variables. We have shown that this problem is decidable for various special cases of such systems. The decision results we present are based on two facts. The first one is that, in all the cases we consider, we can show that the verification problem is reducible to a verification problem on a discrete structure. This is done either using some adequate partitioning of the dense configuration space preserving the satisfaction of the reachability properties, or by using digitization principles in order to show that for every computation sequence of the system, there exists some *integer* computation that visits the same control locations and fires the same transitions. Then, the second fact, is that the verification problem on the discrete structure we obtain reduces to a control location (constrained) reachability problem in a pushdown automaton, which can be solved as a nonemptiness problem of a context-free language, or as an integer linear programming problem in the constrained case.

References

[ACD90] R. Alur, C. Courcoubetis, and D. Dill. Model-Checking for Real-Time Systems. In *5th Symp. on Logic in Computer Science (LICS'90)*. IEEE, 1990.

[ACH93] R. Alur, C. Courcoubetis, and T. A. Henzinger. Computing Accumulated Delays in Real-time Systems. In *Proc. Intern. Conf. on Computer Aided Verification (CAV'93)*. LNCS 697, 1993.

[ACHH93] R. Alur, C. Courcoubetis, T. Henzinger, and P-H. Ho. Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. In *Hybrid Systems*. LNCS 736, 1993.

[BER94a] A. Bouajjani, R. Echahed, and R. Robbana. Verification of Context-Free Timed Systems using Linear Hybrid Observers. In *Proc. Intern. Conf. on Computer Aided Verification (CAV'94)*. LNCS 818, 1994.

[BER94b] A. Bouajjani, R. Echahed, and R. Robbana. Verifying Invariance Properties of Timed Systems with Duration Variables. In *Intern. Symp. on Formal Techniques in Real Time and Fault Tolerant Systems (FTRTFT'94)*. LNCS 863, 1994.

[Cer92] K. Cerans. Decidability of Bisimulation Equivalence for Parallel Timer Processes. In *Proc. Intern. Conf. on Computer Aided Verification (CAV'92)*. LNCS 663, 1992.

[HKPV95] T. Henzinger, P. Kopke, A. Puri, and P. Varaiya. What's Decidable about Hybrid Automata. In *STOC'95*, 1995.

[HMP92] T. Henzinger, Z. Manna, and A. Pnueli. What Good are Digital Clocks? In *19th. Intern. Coll. on Automata, Languages and Programming (ICALP'92)*. LNCS 623, 1992.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley Pub. Comp., 1979.

[KPSY93] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration Graphs: A Class of Decidable Hybrid Systems. In *Hybrid Systems*. LNCS 736, 1993.

[MMP92] O. Maler, Z. Manna, and A. Pnueli. From Timed to Hybrid Systems. In *REX workshop on Real-Time: Theory and Practice*. LNCS 600, 1992.

[MP93] Z. Manna and A. Pnueli. Verifying Hybrid Systems. In *Hybrid Systems*. Springer-Verlag, 1993. LNCS 736.

[NOSY93] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An Approach to the Description and Analysis of Hybrid Systems. In *Hybrid Systems*. LNCS 736, 1993.

[NSY92] X. Nicollin, J. Sifakis, and S. Yovine. From ATP to Timed Graphs and Hybrid Systems. In *REX workshop on Real-Time: Theory and Practice*. LNCS 600, 1992.

[PV94] A. Puri and P. Varaiya. Decidability of Hybrid Systems with Rectangular Differential Inclusions. In *Proc. Intern. Conf. on Computer Aided Verification (CAV'94)*. LNCS 818, 1994.

Appendix

Proof of Proposition 4.2

Let α be an integer configuration of $\mathcal{H}$, $\sigma = \{\langle \ell_i, \lambda_i, \nu_i, \tau_i \rangle\}_{i \in \omega}$ a computation sequence of $\mathcal{H}$ starting from α, and $\rho = \{\langle \ell_i, \tau_i \rangle\}_{i \in \omega}$ the trail generated by σ. We assume without loss of generality that for every $k \in I\!N$, there exists some index i_k, such that $t_{i_k} = k$. To simplify the exposition of the proof, we assume that the variable y is never reset along σ, and that its initial value is 0. Taking into account resets of y, and the consideration of other integer initial values for y, do not present any difficulty.

Let $\vec{\epsilon} = \{\epsilon_i\}_{i \in \omega}$ be a digitization sequence. For every $k \in \omega$, we define

$$\Delta^\rho(k) = \tau_{k+1} - \tau_k$$

and

$$\Delta^\rho_{\vec{\epsilon}}(k) = [\tau_{k+1}]_{\epsilon_{\lfloor \tau_{k+1} \rfloor}} - [\tau_k]_{\epsilon_{\lfloor \tau_k \rfloor}}.$$

Intuitively, $\Delta^\rho(k)$ is the time taken by the k^{th} transition in the real trail ρ (i.e., the time spent at location ℓ_k) whereas $\Delta^\rho_{\vec{\epsilon}}(k)$ is the time taken by the same transition in the integer trail $[\rho]_{\vec{\epsilon}}$.

Given $k \in \omega$, we define $E^k_{\vec{\epsilon}}$ to be the difference between the values of the integrator y in $[\rho]_{\vec{\epsilon}}$ and ρ at position k, i.e., just after reaching the k^{th} configuration (if it exists). Let us give the formal definition. For every $k \in \omega$, it is clear that

$$\nu_k(y) = \sum_{i=0}^{k-1} \partial(\ell_i, y) \cdot \Delta^\rho(i).$$

Then, let us define

$$\nu^k_{\vec{\epsilon}}(y) = \sum_{i=0}^{k-1} \partial(\ell_i, y) \cdot \Delta^\rho_{\vec{\epsilon}}(i)$$

Intuitively, $\nu^k_{\vec{\epsilon}}(y)$ is the value of of y at position k in the integer trail $[\rho]_{\vec{\epsilon}}$. So, we have

$$E^k_{\vec{\epsilon}} = \nu^k_{\vec{\epsilon}}(y) - \nu_k(y)$$

First, it is easy to check that the following fact holds.

Lemma 5.1 *Let $k \in \omega$. Then, $|E^k_{\vec{\epsilon}}| < 1$ implies that for every $n \in I\!N$,*

- *$\nu_k(y) = n$ implies that $\nu^k_{\vec{\epsilon}}(y) = n$,*
- *$n < \nu_k(y) < n+1$ implies that $\nu^k_{\vec{\epsilon}}(y) \in \{n, n+1\}$.*

Since $\mathcal{H}$ is a weak system, we can deduce that when $|E^k_{\vec{\epsilon}}| < 1$, then $\nu_k(y)$ and $\nu^k_{\vec{\epsilon}}(y)$ are equivalent w.r.t. the guards of $\mathcal{H}$. Hence, if $|E^k_{\vec{\epsilon}}| < 1$ for every position k where some guarded transition is taken in σ, then the trail $[\rho]_{\vec{\epsilon}}$ corresponds to a computation sequence of $\mathcal{H}$. We prove hereafter that it is always possible to find $\vec{\epsilon}$ such that $|E^k_{\vec{\epsilon}}| < 1$ for every $k \in \omega$.

Let $u \in I\!N$, $\mathcal{I}_u = [u, u+1]$, and $J = \{k, k+1, \ldots, k+m\}$ be the maximal set of indices such that $\forall j \in J$, $\tau_j \in \mathcal{I}_u$. Notice that, by the assumptions on the computation σ mentioned above, we have necessarily $\tau_k = u$ and $\tau_{k+m} = u+1$.

Then, we show that if there exists some digitization sequence $\vec{\epsilon}$ such that $\epsilon_0, \cdots, \epsilon_{u-1}$ allows to digitize ρ until position k in such a way that the distance between the real value of y and its digitization (i.e., the error on y) is always less than 1, then it is possible to find a digitization quantum ϵ allowing to prolong this digitization to the interval $\mathcal{I}_u$ and keep the error on y less than 1 along the interval $\mathcal{I}_u$ too.

Lemma 5.2 *For every $\vec{\epsilon}$, if $|E^k_{\vec{\epsilon}}| < 1$, then there exists $\vec{\epsilon}'$ such that*

- *$\vec{\epsilon}'_i = \vec{\epsilon}_i$ for every $i < u$, and*
- *$\forall j \in J$, $|E^j_{\vec{\epsilon}'}| < 1$.*

Proof. Let us denote by J^+ the subset of $J - \{k+m\}$ such that for every $j \in J^+$, we have $\Delta^\rho(j) = \tau_{j+1} - \tau_j \neq 0$. Intuitively, J^+ is the set of indices of transitions corresponding to time progress in the interval $\mathcal{I}_u$, in other words, to transitions that (may) modify the values of the variables. Let $\imath_1$ and $\imath_2$ denote respectively the minimum and the maximum of J^+. Notice that $\tau_{\imath_1} = \tau_k = u$ and $\tau_{\imath_2+1} = \tau_{k+m} = u+1$. Notice also that $E^k_{\vec{\epsilon}} = E^k_{\vec{\epsilon}'}$.

Now, assume that $|E^k_{\vec{\epsilon}}| < 1$. We show that we can chose a value in $\epsilon \in [0, 1[$ such that, for any sequence $\vec{\epsilon}' = \vec{\epsilon}(0) \cdots \vec{\epsilon}(u-1) \cdot \epsilon \cdots$, we have for every $j \in J$, $|E^j_{\vec{\epsilon}'}| < 1$. Actually, it suffices to consider the values $E^{j+1}_{\vec{\epsilon}'}$'s with $j \in J^+$, i.e., just after some transition corresponding to time progress, since the variables are not modified by the other transitions. We recall that, by definition, we have

$$E^{j+1}_{\vec{\epsilon}'} = E^k_{\vec{\epsilon}'} + \sum_{i \in J^+, i \leq j} \partial(\ell_i, y) \cdot (\Delta^\rho_{\vec{\epsilon}'}(i) - \Delta^\rho(i)) \tag{5}$$

The proof is carried out by considering two cases according to the fact that $E^k_{\vec{\epsilon}}$ is positive or negative.

Let us start with the case $0 \leq E^k_{\vec{\epsilon}} < 1$. Again, there are tree subcases to consider, according to the fact that all of the rates of y in the locations ℓ_j with $j \in J^+$ are 1, or all of them are 0, or there are some of them that are positive and others that are null.

1.1. Suppose that for every $j \in J^+$, we have $\partial(\ell_j, y) = 0$. Then, we can take any $\vec{\epsilon}'$ such that as $\vec{\epsilon}'(u)$ any value in $[0, 1[$. Indeed, in this case, from (5), we have for every $j \in J^+$,

$$E^{j+1}_{\vec{\epsilon}'} = E^k_{\vec{\epsilon}'}$$

1.2. Suppose that for every $j \in J^+$, we have $\partial(\ell_j, y) = 1$. Then, we show that we can take any $\vec{\epsilon}'$ such that $\vec{\epsilon}'(u) = \epsilon \in [\mathit{fract}(\tau_{\imath_2}), 1[$.
Now, since the rates of y are 1 in all the locations ℓ_j's, by definition of the digitization, we have for every $i \in J^+$, $[\tau_i]_\epsilon = u$. Thus, for every $i \in J^+$ with $i \neq \imath_2$, we have $\Delta^\rho_{\vec{\epsilon}'}(i) = 0$, and $\Delta^\rho_{\vec{\epsilon}'}(\imath_2) = 1$.
Then, from (5), we have for every $j \in J^+$ such that $j \neq \imath_2$,

$$E^{j+1}_{\vec{\epsilon}'} = E^k_{\vec{\epsilon}'} - \sum_{i \in J^+, i \leq j} \Delta^\rho(i)$$

and since

$$0 < \sum_{i \in J^+, i \leq j} \Delta^\rho(i) = \tau_{j+1} - \tau_{\imath_1} < 1$$

and since $0 \leq E^k_{\vec{\epsilon}} < 1$ by hypothesis, we have necessarily

$$-1 < E^{j+1}_{\vec{\epsilon}'} < E^k_{\vec{\epsilon}} < 1 \tag{6}$$

On the other hand, we obtain also

$$E^{\imath_2+1}_{\vec{\epsilon}'} = E^k_{\vec{\epsilon}'} + 1 - \sum_{i \in J^+} \Delta^\rho(i)$$

then, since

$$\sum_{i \in J^+} \Delta^\rho(i) = \tau_{\imath_2+1} - \tau_{\imath_1} = 1$$

we have

$$E^{\imath_2+1}_{\vec{\epsilon}'} = E^k_{\vec{\epsilon}'} \tag{7}$$

Thus, from (6) and (7), we obtain that for every $j \in J^+$,

$$-1 < E^{j+1}_{\vec{\epsilon}'} \leq E^k_{\vec{\epsilon}} < 1$$

1.3. Suppose that y has some rates equal to 1 and others equal to 0. Then, let J^+_1 (resp. J^+_0) be the set of indices $j \in J^+$ such that $\partial(\ell_j, y) = 1$ (resp. $\partial(\ell_j, y) = 0$).
Let p be any index in J^+_0. We show that we can take $\vec{\epsilon}'$ such that $\vec{\epsilon}'(u)$ in $[\mathit{fract}(\tau_p), \mathit{fract}(\tau_{p+1})[$ if $\tau_{p+1} \neq u + 1$, otherwise in $[\mathit{fract}(\tau_p), 1[$.
Indeed, from (5), we have for every $j \in J^+$,

$$E^{j+1}_{\vec{\epsilon}'} = E^k_{\vec{\epsilon}'} - \sum_{i \in J^+_1, i \leq j} \Delta^\rho(i)$$

Then, it is clear that we have

$$0 \leq \sum_{i \in J^+_1, i \leq j} \Delta^\rho(i) < 1$$

and thus, since $0 \leq E^k_{\vec{\epsilon}'} < 1$, we deduce that for every $j \in J^+$,

$$-1 < E^{j+1}_{\vec{\epsilon}'} \leq E^k_{\vec{\epsilon}'} < 1$$

This terminates the proof in the case $0 \leq E^k_{\vec{\epsilon}} < 1$. So, let us consider now the case $-1 < E^k_{\vec{\epsilon}'} \leq 0$.

Then, we have to consider the same tree subcases as above according to the rates of y.

The first case is exactly the same as previously (see case 1.1). Then, let us consider the other cases.

2.1. Suppose that for every $j \in J^+$, we have $\partial(\ell_j, y) = 1$. We show that we can take $\vec{\epsilon}'(u) = \epsilon \in [0, \mathit{fract}(\tau_{\imath_1+1})[$ if $\tau_{\imath_1+1} \neq u+1$, otherwise we take $\vec{\epsilon}'(u) = \epsilon \in [0,1[$.
Indeed, we have in this case $[\tau_{\imath_1}]_\epsilon = u$, and for every $i \in J^+$ such that $i \neq \imath_1$, we have $[\tau_i]_\epsilon = u+1$. Then, from (5), for every $j \in J^+$, we have

$$E^{j+1}_{\vec{\epsilon}'} = E^k_{\vec{\epsilon}'} + 1 - \sum_{i \in J^+, i \leq j} \Delta^\rho(i)$$

thus, since

$$0 < \sum_{i \in J^+, i \leq j} \Delta^\rho(i) \leq 1$$

and since $-1 < E^k_{\vec{\epsilon}'} \leq 0$ by hypothesis, we have necessarily for every $j \in J^+$,

$$-1 < E^k_{\vec{\epsilon}'} \leq E^{j+1}_{\vec{\epsilon}'} < 1$$

2.2. Suppose now that y has some rates equal to 1 and others equal to 0. Then, let J^+_1 and J^+_0 be the partition of J^+ defined exactly as in the case (1.3) above. Let p be the minimum of the set J^+_1. Then, we have to take $\vec{\epsilon}'$ such that $\vec{\epsilon}'(u)$ is in the interval $[\mathit{fract}(\tau_p), \mathit{fract}(\tau_{p+1})[$ if $\tau_{p+1} \neq u+1$, otherwise in $[\mathit{fract}(\tau_p), 1[$.
Indeed, in this case, for every $j \in J^+$ such that $j < p$ (hence, j is necessarily in J^+_0 if it exists), we have

$$E^{j+1}_{\vec{\epsilon}'} = E^k_{\vec{\epsilon}'} \tag{8}$$

On the other hand, for every $j \in J^+$ such that $j \geq p$, we have

$$E^{j+1}_{\vec{\epsilon}'} = E^k_{\vec{\epsilon}'} + 1 - \sum_{i \in J^+_1, i \leq j} \Delta^\rho(i)$$

thus, since it can be seen that

$$0 < \sum_{i \in J^+_1, i \leq j} \Delta^\rho(i) < 1$$

and since $-1 < E^k_{\vec{\epsilon}'} \leq 0$ by hypothesis, we have necessarily for every $j \in J^+$ with $j \geq p$,

$$-1 < E^k_{\vec{\epsilon}'} < E^{j+1}_{\vec{\epsilon}'} < 1 \tag{9}$$

Hence, from (8) and (9), we obtain that for every $j \in J^+$,

$$-1 < E^k_{\vec{\epsilon}'} \leq E^{j+1}_{\vec{\epsilon}'} < 1$$

This terminates the proof in the case $-1 < E^k_{\vec{\epsilon}} \leq 0$ too.

This article was processed using the LaTeX macro package with LLNCS style