Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Electronic Notes in Theoretical Computer Science 239 (2009) 167–178
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Reachability in Timed Counter Systems [1]

Florent Bouchy[†], Alain Finkel[†] and Arnaud Sangnier[†,‡]

[†]*LSV, ENS Cachan, CNRS* [‡]*EDF R&D*
61 avenue du Président Wilson
94230 Cachan, France
{bouchy,finkel,sangnier}@lsv.ens-cachan.fr

Abstract

We introduce Timed Counter Systems, a new class of systems mixing clocks and counters. Such systems have an infinite state space, and their reachability problems are generally undecidable. By abstracting clock values with a Region Graph, we show the Counter Reachability Problem to be decidable for three subclasses : Timed VASS, Bounded Timed Counter Systems, and Reversal-Bounded Timed Counter Systems.

Keywords: Counter Systems, Timed Automata, Verification, Reachability, Petri Nets, VASS, Reversal-Bounded

1 Introduction

Context. Formal verification of systems featuring temporal constraints or counting abilities has been largely studied. Indeed, clocks seem to be the most natural way to model time, and counters appear as the most used datatype in case studies. Usually, such systems are finite automata endowed with clocks or counters, whose values are determined by operations associated to automata transitions. In this paper, we follow this widespread approach and define Timed Counter Systems, based on two well-known models. Indeed, we express the same temporal requirements as timed automata [2], and we use counter systems extending Minsky machines [24] (more precisely, a combination of relational counter automata [11] and functional Presburger counter systems [16]). Timed Counter Systems have thus two different datatypes at the same time : continuous (i.e. real-valued, or dense) clocks, and discrete (i.e. integer-valued) counters.

Related work. A few classes of systems mixing clocks and counters have already been studied. Hybrid automata [1], a well-known extention of timed automata,

[1] Work supported by the Agence Nationale de la Recherche, grant ANR-06-SETIN-001.

doi:10.1016/j.entcs.2009.05.038

is a class able to encode Timed Counter Systems by simulating counters with the clocks' differential trajectories ; however, it is so general that the reachability problem remains undecidable even for very restricted subclasses. Several timed versions of Petri Nets are well-known (e.g. time intervals [23] or aging tokens [10]), but neither of them is able to easily simulate our clocks ; even when they do, our counters are more expressive. Dense Counter Machines [25] are Minsky machines augmented with non-deterministic fractional value changes ; they are not comparable to Timed Counter Systems, because their dense datatype has a different behaviour than our clocks. The same authors also defined Real-counter Automata [14], but it is not clear that they may encode our clocks and counters. They also investigated variations of Pushdown Timed Automata [13], in which the stack could be viewed as a counter ; but their clocks are integer-valued, and thus can be simulated by our clocks. Finally, Parametric Timed Counter Systems are used in the TReX tool [3,4], but their expressivity is not comparable to the one of Timed Counter Systems (see [15] for a study thereof) ; indeed, their clocks are more expressive than ours, but their counters are less expressive than ours (excepted some arithmetical terms using multiplication, which is not possible with our counters). Nevertheless, the systems of TReX are highly undecidable for reachability matters, and have no decidable subclass which seems natural, to the best of our knowledge.

Our contributions. A major interest in verification is reachability ; in this paper, we address the Counter Reachability Problem for Timed Counter Systems, in which the exact clock values are left apart (as usually done with timed automata). We prove it to be decidable for subclasses of such systems, in which the Region Graph belongs to a class of counter systems for which this problem is decidable. We also identify three of these subclasses.

2 Timed Counter Systems

2.1 Preliminary definitions

In order to use a homogeneous model for systems mixing clocks and counters, let us first define the basis we will be using. The next two paragraphs explain our way to handle clocks and counters, so that they can be handled at the same level.

Clocks

Let X be a set of m real-valued variables, called *clocks*. A *clock valuation* over X is a vector $\mathbf{x} \in \mathbb{R}_+^m$. Given a clock valuation $\mathbf{x}$ and a duration $\tau \in \mathbb{R}_+$, $\mathbf{x} + \tau$ is the clock valuation defined by $(\mathbf{x} + \tau)_i = \mathbf{x}_i + \tau$ for every $i \in [1, m]$.
Let $R_X = G_X \times \{0, 1\}^m$ be the set of operations on clocks, where :

- G_X denotes clock constraints (or *guards*), defined by the following grammar : $g ::= x - y \bowtie b \mid x \bowtie b \mid g \wedge g \mid \neg g$, with $\bowtie \in \{<, \leq, =, \geq, >\}$, $x, y \in X$, $b \in \mathbb{N}$.
- $\{0, 1\}^m$ intuitively denotes the clocks to be reset.

For a guard $g \in G_X$ and a clock valuation $\mathbf{x} \in \mathbb{R}_+^m$, we denote by $\mathbf{x} \models g$ the fact that the clock valuation $\mathbf{x}$ satisfies the guard g. By convention, when $X = \emptyset$, then $R_X = \{\emptyset\}$. Let $\mathbf{x}, \mathbf{x}' \in \mathbb{R}_+^m$ and $(g, \lambda) \in R_X$. Then $(\mathbf{x}, \mathbf{x}') \models (g, \lambda)$ is defined by : $\mathbf{x} \models g$ and $\forall i \in [1, m], \lambda_i = 0 \implies \mathbf{x}'_i = 0$ and $\lambda_i = 1 \implies \mathbf{x}'_i = \mathbf{x}_i$ (or, more simply, $\mathbf{x}'_i = \lambda_i \mathbf{x}_i$).

From now on, for the sake of readability, we suppose that clock guards do not use atomic diagonal guards (i.e. guards of the form $x - y \bowtie b$), and this, w.l.o.g. : indeed, [9] introduces a translation of timed automata into diagonal-free timed automata.

Counters

Let C be a set of n integer-valued variables, called *counters*. A *counter valuation* over C is a vector $\mathbf{c} \in \mathbb{Z}^n$. Let $R_C \subseteq \mathbb{Z}^n \times \mathbb{Z}^n$ be the set of relations which can be defined with a Presburger formula. Intuitively, such binary relations describe the effect of a transition on the counters ; that is, for some $r \in R_C$, $(\mathbf{c}, \mathbf{c}') \in r$ means that the valuation on counters is $\mathbf{c}$ before a transition labelled by r, and is $\mathbf{c}'$ after this transition. In fact, we encode the guards and operations on counters in a single formula r, whose solutions are $(\mathbf{c}, \mathbf{c}')$. By convention, when $C = \emptyset$, then $R_C = \{\emptyset\}$.

2.2 Syntax

Definition 2.1 A *Timed Counter System* (*TCS* for short) is a tuple $\langle Q, X, C, E \rangle$ where :

- Q is a finite set of control states (also called *locations*)
- $E \subseteq Q \times R_X \times R_C \times Q$ is a finite set of transitions (edges)

Notice that a TCS is in fact a combination of two well-known models : Timed Automata and Counter Systems. Let us define both of them with our notations :

Definition 2.2 A *Timed Automaton* (*TA* for short) is a TCS $\mathcal{S}$ where $C = \emptyset$. Similarly, a *Counter System* (*CS* for short) is a TCS $\mathcal{S}$ where $X = \emptyset$.

2.3 Semantics

In order to study the behaviour of a TCS, one can look in 3 directions, according to which kind of variables are interpreted. Indeed, a TCS can be unfolded along its clocks only, or along its counters only, or along both at the same time. We say that a TCS whose interpretation considers only clocks (resp. counters) is called a Timed (resp. Counting) Transition System ; if both clocks and counters are interpreted, then the full semantics of a TCS is given by a Transition System.

2.3.1 Timed Semantics

The timed behaviour of a TCS is described by a Timed Transition System (TTS) :

Definition 2.3 The timed semantics of a TCS $\mathcal{S} = \langle Q, X, C, E \rangle$ is given by a tuple $TTS(\mathcal{S}) = \langle S_T, \rightarrow_T \rangle$, where :

- $S_T = Q \times \mathbb{R}_+^m$ is the set of configurations
- $\rightarrow_T \subseteq S_T \times (E \cup \mathbb{R}_+) \times S_T$ is the transition relation composed of delays and steps :

$$(q, \mathbf{x}) \rightarrow_T (q', \mathbf{x}') \iff \begin{cases} \text{(delay, noted } \xrightarrow{\tau}_T \text{)} \\ q = q' \text{ and } \exists \tau \in \mathbb{R}_+ \text{ such that } \mathbf{x}' = \mathbf{x} + \tau \\ \\ \text{(step, noted } \xrightarrow{e}_T \text{)} \\ \exists e = (q, (g, \lambda), r, q') \in E \text{ such that } (\mathbf{x}, \mathbf{x}') \models (g, \lambda) \end{cases}$$

Notice that if $\mathcal{S}$ is a TA, then $TTS(\mathcal{S})$ gives the usual timed semantics of TA.

2.3.2 Counting Semantics

The counting behaviour of a TCS is described by a Counting Transition System (CTS) :

Definition 2.4 The counting semantics of a TCS $\mathcal{S} = \langle Q, X, C, E \rangle$ is given by a tuple $CTS(\mathcal{S}) = \langle S_C, \rightarrow_C \rangle$, where :

- $S_C = Q \times \mathbb{Z}^n$ is the set of configurations
- $\rightarrow_C \subseteq S_C \times E \times S_C$ is the transition relation defined by $(q, \mathbf{c}) \xrightarrow{e}_C (q', \mathbf{c}') \iff \exists (q, (g, \lambda), r, q') \in E$ such that $(\mathbf{c}, \mathbf{c}') \in r$

Notice that if $\mathcal{S}$ is a CS, then $CTS(\mathcal{S})$ gives the usual semantics of CS.

2.3.3 Full Semantics

We give the complete (i.e. timed and counting) behaviour of a TCS by combining a TTS and a CTS as follows :

Definition 2.5 The full semantics of a TCS $\mathcal{S} = \langle Q, X, C, E \rangle$ is given by a tuple $TS(\mathcal{S}) = \langle S, \rightarrow \rangle$, where :

- $S = Q \times \mathbb{R}_+^m \times \mathbb{Z}^n$ is the set of configurations
- $\rightarrow \subseteq S \times (E \cup \mathbb{R}_+) \times S$ is the transition relation composed of delays and steps :

$$(q, \mathbf{x}, \mathbf{c}) \rightarrow (q', \mathbf{x}', \mathbf{c}') \iff \begin{cases} \text{(delay, noted } \xrightarrow{\tau} \text{)} \\ q = q' \text{ and } \mathbf{c} = \mathbf{c}' \text{ and } \exists \tau \in \mathbb{R}_+ \text{ such that :} \\ \mathbf{x}' = \mathbf{x} + \tau \\ \\ \text{(step, noted } \xrightarrow{e} \text{)} \\ \exists e = (q, (g, \lambda), r, q') \in E \text{ such that :} \\ (\mathbf{c}, \mathbf{c}') \in r \text{ and } (\mathbf{x}, \mathbf{x}') \models (g, \lambda) \end{cases}$$

Using the previous definitions, the next proposition gives the relation between the different semantics :

Proposition 2.6 *Let $\mathcal{S} = \langle Q, X, C, E\rangle$ be a TCS. Then, we have :*

(i) $\forall e \in E$, $(q, \mathbf{x}, \mathbf{c}) \xrightarrow{e} (q', \mathbf{x}', \mathbf{c}')$ *iff* $(q, \mathbf{x}) \xrightarrow{e}_T (q', \mathbf{x}')$ *and* $(q, \mathbf{c}) \xrightarrow{e}_C (q', \mathbf{c}')$.

(ii) $\forall \tau \in \mathbb{R}_+$, $(q, \mathbf{x}, \mathbf{c}) \xrightarrow{\tau} (q, \mathbf{x}', \mathbf{c})$ *iff* $(q, \mathbf{x}) \xrightarrow{\tau}_T (q, \mathbf{x}')$.

An example of TCS

Figure 1 depicts an example of Timed Counter System, with two control locations q_1, q_2, two counters $\mathbf{c}_1, \mathbf{c}_2$, and two clocks $\mathbf{x}_1, \mathbf{x}_2$. We consider the initial configuration in q_1 with $\mathbf{c}_1 = \mathbf{c}_2 = \mathbf{x}_1 = \mathbf{x}_2 = 0$. Notice that $\mathbf{x}_2$ does not appear on transitions, but only stands as a universal clock.

This TCS represents a service offered on most digital televisions : the feature modelled here deals with the movies that the client can rent directly at home. This model mainly gives the following information : the total number of movies the client has rented so far ($\mathbf{c}_2$), the number of movies having been rented during the current day ($\mathbf{c}_1$), how long the client has been using this service ($\mathbf{x}_2$), and how much time has elapsed since the first daily movie ($\mathbf{x}_1$). The typical property this model aims at representing is "A client can rent a maximum of 5 movies in a 24-hour period". One could also model fares, and by using $\mathbf{c}_2$ and $\mathbf{x}_2$, offer a free movie every 30 rentals after a one-month membership. Other statistics can easily be derived from this model, such as the average number of movies a client uses to rent per hour.

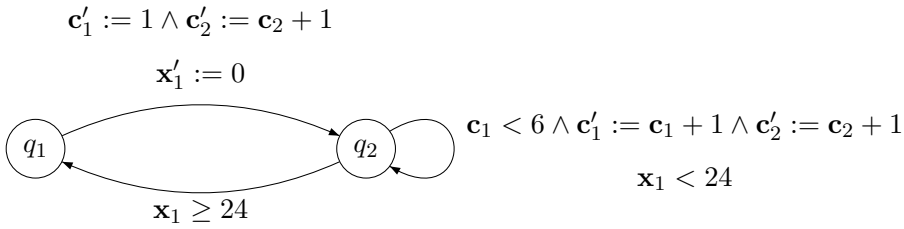

Fig. 1. An example of TCS

3 Reachability

A typical interesting problem in the field of verification is the reachability problem, which can roughly be defined as follows : "Given two configurations s, s' of a system, is there an execution of the system going from s to s' ?". In our case, we refine this problem : instead of checking if a full configuration is reachable, we will check if a pair $(q, \mathbf{c})$, where q is a control state and $\mathbf{c}$ a counter valuation, is reachable from an initial given configuration. We now formalize this notion.

Let $\mathcal{S}$ be a TCS and $TS(\mathcal{S}) = \langle S, \rightarrow\rangle$ its associated full semantics. We denote by $\xrightarrow{*}$ the reflexive and transitive closure of $\rightarrow$. Similarly we define $\xrightarrow{*}_C$ for the counting semantics. We then define the *reachability sets* of $\mathcal{S}$ as follows :

- $\mathrm{Reach}(\mathcal{S}, s_0) = \{s \in S \mid s_0 \xrightarrow{*} s\}$, for any $s_0 \in S$
- $\mathrm{Reach}_C(\mathcal{S}, s_0) = \{s \in S_C \mid s_0 \xrightarrow{*}_C s\}$, for any $s_0 \in S_C$

In this paper, we are interested in the Counter Reachability Problem, which we define as follows :

Counter Reachability Problem :

Inputs : A TCS $\mathcal{S}$, an initial configuration s_0 of $TS(S)$, and a configuration $(q, \mathbf{c})$ of $CTS(S)$.

Question : Is there a clock valuation $\mathbf{x}$ such that $(q, \mathbf{x}, \mathbf{c}) \in \text{Reach}(\mathcal{S}, s_0)$?

This problem considers only counter valuations, and not the clock valuations. We chose to look at this problem, instead of the reachability problem with a whole configuration, because we believe that the clocks are only used to introduce temporal requirements in the behavior of the system, and consequently, that there is no need to keep track of their exact values for verification matters.

Notice that this Counter Reachability Problem is an extension of the classical reachability problem in CS : the only difference is that we existentially quantify on a clock valuation so that the configuration matches a full TS configuration, and not just a CTS configuration. Therefore, we will equivalently speak of the Counter Reachability Problem for TCS and for CS (as usually defined, ie. without quantifying on clock valuations).

The Counter Reachability Problem is obviously undecidable for TCS, because it is already undecidable in CS. In order to be able to analyze CS, some restrictions leading to decidability (e.g. flat [12], reversal-bounded [19], VASS [20], ...) have been proposed. As we will show in section 5, some of these restrictions can be lifted up to the level of TCS. The main idea we will develop in this paper uses the fact that the undecidability of TCS is caused by the presence of counters. Therefore, we try to benefit from known decidability results on TA (detailed in section 4) and on some subclasses of CS (detailed in section 5).

4 Analysis of TCS via clock abstraction

A typical analysis of a TCS would be to compute the set of its reachable configurations, in order to address e.g. verification problems. Unfortunately, since a TCS handles variables whose domains are unbounded, its set of configurations might be infinite. A classical method to analyse such infinite-state systems consists in finding a finite abstraction, using for instance equivalence classes over configurations, and then ensuring that the reachability problem can be solved by reasoning on the abstracted system. The approach chosen in this paper uses this idea ; however, instead of reasoning on equivalence classes for the whole set of configurations, we only abstract clock valuations. In order to do so, we use a Region Graph, as usually done with TA.

There might be a possible dual approach : to abstract counters first, instead of clocks. The two main reasons why we chose not to use counter abstraction are (1) because counters evolve discretely through formulas on transitions, and not constantly in a dense space when staying in a control location, and (2) because the

region graph has been studied for a long time and proved efficient in several tools.

4.1 Region Graph Construction

Let $\mathcal{S}$ be a TCS defined over a set of m clocks. Let $\mathcal{M}_i$ be the largest constant to which each clock $\mathbf{x}_i$ is ever compared in guards, for all $i \in [1, m]$. As defined in [2], we consider an equivalence relation on clock valuations. Two clock valuations $\mathbf{x}$ and $\mathbf{x}'$ in $\mathbb{R}_+^m$ are said *region-equivalent* (written $\mathbf{x} \approx \mathbf{x}'$) whenever all of the three following conditions hold (where $\lfloor y \rfloor$ (resp. $\llcorner y \lrcorner$) denotes the integer (resp. fractional) part of any $y \in \mathbb{R}$) :

(i) $\lfloor \mathbf{x}_i \rfloor = \lfloor \mathbf{x}'_i \rfloor$ or $\mathbf{x}_i, \mathbf{x}'_i > \mathcal{M}_i$ for all $i \in [1, m]$.

(ii) $\llcorner \mathbf{x}_i \lrcorner = 0$ iff $\llcorner \mathbf{x}'_i \lrcorner = 0$ for all $i \in [1, m]$ such that $\mathbf{x}_i \leq \mathcal{M}_i$.

(iii) $\llcorner \mathbf{x}_i \lrcorner \leq \llcorner \mathbf{x}_j \lrcorner$ iff $\llcorner \mathbf{x}'_i \lrcorner \leq \llcorner \mathbf{x}'_j \lrcorner$, for all $i, j \in [1, m]$ such that $\mathbf{x}_i, \mathbf{x}_j \leq \mathcal{M}_i$.

This equivalence relation can be extended to states of $TTS(\mathcal{S})$, saying that $(q, \mathbf{x}) \approx (q', \mathbf{x}')$ iff $q = q'$ and $\mathbf{x} \approx \mathbf{x}'$. We use $[\mathbf{x}]$ to denote the equivalence class to which $\mathbf{x}$ belongs. A *region* ρ is an equivalence class of clock valuations ; the set of all regions is denoted by $\mathcal{R}$, and is finite. We equivalently write $x \in \rho$ and $[x] = \rho$. A nice known property of the equivalence relation $\approx$ is that it is compatible with clock constraints (denoted by (cc)) and time elapsing (denoted by (te)) :

$$\mathbf{x} \approx \mathbf{x}' \implies \begin{cases} (cc) & \forall g \in G_X,\ \mathbf{x} \models g \iff \mathbf{x}' \models g \\ (te) & \forall \tau \in \mathbb{R}_+, \exists \tau' \in \mathbb{R}_+ \text{ s.t. } \mathbf{x} + \tau \approx \mathbf{x}' + \tau' \end{cases}$$

This second point (te) enables us to define a successor function on $\mathcal{R}$. For a region $\rho \in \mathcal{R}$, we denote by $\mathrm{Succ}(\rho)$ the set of its *time-successors*, defined as follows : $\rho' \in \mathrm{Succ}(\rho) \iff \exists \mathbf{x} \in \rho\ \exists \tau \in \mathbb{R}_+$ s.t. $\mathbf{x} + \tau \in \rho'$. Then, we are able to define the region graph of $\mathcal{S}$:

Definition 4.1 Let $\mathcal{S} = \langle Q, X, C, E \rangle$ be a TCS ; its *region graph* is the tuple $RG(\mathcal{S}) = \mathcal{S}_{/\approx} = \langle \Gamma, \rightarrow_{RG} \rangle$ such that :

- $\Gamma = Q \times \mathcal{R}$ is the set of states ; we sometimes write $q_{\mathbf{x}}$ to denote the state $(q, [\mathbf{x}])$
- $\rightarrow_{RG} \subseteq \Gamma \times E \times \Gamma$ is the transition relation such that $\forall e = (q, (g, \lambda), r, q') \in E$, $(q, \rho) \xrightarrow{e}_{RG} (q', \rho')$ iff $\exists \rho'' \in \mathrm{Succ}(\rho)$ s.t. $\forall \mathbf{x}'' \in \rho''$, $\mathbf{x}'' \models g$ and $\mathbf{x}' \in \rho'$ and $\forall i \in [1, m]$, $\mathbf{x}'_i = \lambda_i \mathbf{x}_i$.

Such a region graph is the same as the classical region graph defined for TA ; its particularity is that its transitions are labelled by relations on counters, which have not been taken into account so far. The next step is, of course, to use them in order to get closer to the full semantics of a TCS.

4.2 The Region Graph as a Counter System

In this section, we first show that the region graph of a TCS can be analyzed as a CS. Then, we prove that the reachability problem can be lifted up to the level of the region graph. The Region Graph enjoys the following property [2] :

Proposition 4.2 *Let $\mathcal{S} = \langle Q, X, C, E\rangle$ be a TCS, $TTS(\mathcal{S}) = \langle S_T, \rightarrow_T\rangle$ its Timed Transition System, and $RG(\mathcal{S}) = \langle \Gamma, \rightarrow_{RG}\rangle$ its Region Graph. Then for any $(q, \mathbf{x}) \in S_T$, we have for all $e \in E$:*

(i) *If $\exists \tau \in \mathbb{R}_+$ and $\exists (q', \mathbf{x}')$ s.t. $(q, \mathbf{x}) \xrightarrow{\tau}_T (q, \mathbf{x}+\tau) \xrightarrow{e}_T (q', \mathbf{x}')$ then $q_{\mathbf{x}} \xrightarrow{e}_{RG} q'_{\mathbf{x}'}$*

(ii) *If $\exists q'_{\mathbf{x}'}$ s.t. $q_{\mathbf{x}} \xrightarrow{e}_{RG} q'_{\mathbf{x}'}$ then $\exists \tau \in \mathbb{R}_+$ and $\exists \mathbf{x}'' \in [\mathbf{x}']$ s.t. $(q, \mathbf{x}) \xrightarrow{\tau}_T (q, \mathbf{x}+\tau) \xrightarrow{e}_T (q', \mathbf{x}'')$*

Note that this property is about transitions, and can be naturally extended to sequences of such transitions : then, we obtain the well-known *time-abstract bisimulation* between $TTS(\mathcal{S})$ and $RG(\mathcal{S})$, denoted by $\simeq$. Informally, $TTS(\mathcal{S}) \simeq RG(\mathcal{S})$ means that both $TTS(\mathcal{S})$ and $RG(\mathcal{S})$ can follow the exact same sequences of transitions ; the only difference with a regular bisimulation is that $RG(\mathcal{S})$ does not keep track of clock valuations, but only their equivalence class.

Now, notice that since the Region Graph has a finite number of states and its transitions are labeled by relations on counters, we can view it like a classical counter system. Indeed, we can see $RG(\mathcal{S})$ as a TCS $\mathcal{S}' = \langle Q', X', C', E'\rangle$, where $Q' = Q \times \mathcal{R}$, $X' = \emptyset$, $C' = C$ and $E' = E$ (with $R_{X'} = \{\emptyset\}$, since $X' = \emptyset$). Thus, we will alternatively say, w.l.o.g., that $RG(\mathcal{S})$ is a RG, a TCS, or a CS.

We are now ready to prove that we can analyze the TCS through the counting semantics of its region graph, yielding a system which is an exact (w.r.t. Counter Reachability) abstraction of its full semantics. Indeed, from Propositions 2.6 and 4.2, we deduce the following property :

Proposition 4.3 *Let $\mathcal{S}$ be a TCS. Then, we have :*

(i) *If $(q', \mathbf{x}', \mathbf{c}') \in Reach\Big(\mathcal{S}, (q, \mathbf{x}, \mathbf{c})\Big)$, then $(q'_{\mathbf{x}'}, \mathbf{c}') \in Reach_C\Big(RG(\mathcal{S}), (q_{\mathbf{x}}, \mathbf{c})\Big)$*

(ii) *If $(q'_{\mathbf{x}'}, \mathbf{c}') \in Reach_C\Big(RG(\mathcal{S}), (q_{\mathbf{x}}, \mathbf{c})\Big)$, then there exists $\mathbf{x}'' \in \mathbb{R}_+^m$ such that $(q', \mathbf{x}'', \mathbf{c}') \in Reach\Big(\mathcal{S}, (q, \mathbf{x}, \mathbf{c})\Big)$ and $\mathbf{x}'' \in [\mathbf{x}']$.*

The picture on Figure 2 exhibits the different ways to interpret a TCS, and the relations existing between them. It also illustrates Proposition 4.3.

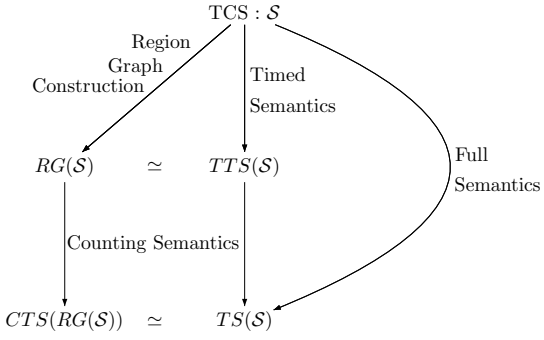

Fig. 2. The links between different semantics of TCS

Let $\mathfrak{C}$ be a class of TCS such that there is an algorithm solving the Counter Reachability Problem for $RG(\mathcal{S})$, for any $\mathcal{S} \in \mathfrak{C}$. From Proposition 4.3 and the fact that there is a finite number of regions, we deduce our main theorem :

Theorem 4.4 *The Counter Reachability Problem is decidable for* $\mathfrak{C}$.

Proof. Let $(q, \mathbf{x}, \mathbf{c})$ be an initial configuration of $TS(\mathcal{S})$ and $(q, \mathbf{c})$ a configuration of $CTS(\mathcal{S})$. Then, from Proposition 4.3, we deduce that there exists a clock valuation $\mathbf{x}'$ such that $(q, \mathbf{x}', \mathbf{c}') \in Reach(S, (q, \mathbf{x}, \mathbf{c}))$ if and only if there exists a region ρ such that $((q, \rho), \mathbf{c}') \in Reach_C(RG(\mathcal{S}), (q_{\mathbf{x}}, \mathbf{c}))$ and $\mathbf{x}' \in \rho$. Since a given TCS yields a finite number of regions, if we suppose that the Counter Reachability Problem is decidable for the counter system $RG(\mathcal{S})$, then the Counter Reachability Problem is decidable for $\mathcal{S}$. □

In the next part, we will use this theorem to show that many restrictions which lead to decidability when studying Counter Systems can be lifted up to the level of TCS in order to obtain the decidability of the counter reachability problem.

5 Subclasses of TCS

We can now address the Counting Reachability Problem for TCS, by making hypotheses on the class of CS to which the TCS's region graph belongs. Therefore, we introduce four subclasses of TCS.

5.1 Timed Counter Machines and Timed VASS

We introduce here the class of Timed Counter Machines, in which we restrict operations on counters. First, we give the definition of the relations over the counters valuations we allow in the Timed Counter Machines, extending the Counter Machines of [19] (which are a slight extension of Minsky machines [24]). We call a *guarded translation* (shortly, a translation) any function $t : \mathbb{N}^n \to \mathbb{N}^n$ such that there exist $\# \in \{=, \leq\}^n$, $\mu \in \mathbb{N}^n$, and $\delta \in \mathbb{Z}^n$ with $0 \leq \mu + \delta$ and $dom(t) = \{\mathbf{c} \in \mathbb{N}^n \mid \mu \# \mathbf{c}\}$ and for all $\mathbf{c} \in \mathbb{N}^n$, $t(\mathbf{c}) = \mathbf{c} + \delta$. Intuitively, μ is the guard and δ is the translation length. We will sometimes use the encoding $(\#, \mu, \delta)$ to represent a translation.
Note that a translation can be seen as a relation over $\mathbb{Z}^n \times \mathbb{Z}^n$. Indeed, for a translation $t : \mathbb{N}^n \to \mathbb{N}^n$ and two counter valuations $\mathbf{c}$ and $\mathbf{c}'$, we have $(\mathbf{c}, \mathbf{c}') \in t$ iff $\mathbf{c} \in dom(t)$ and $\mathbf{c}' = t(\mathbf{c})$. Thus, using the original formalism of TCS, a translation is a relation of the form $\bigwedge_{i \in [1..n]} \mu_i \#_i \mathbf{c}_i \wedge \mathbf{c}'_i = \mathbf{c}_i + \delta_i$.

Definition 5.1 A *Timed Counter Machine* (*TCM* for short) is a TCS $\mathcal{S} = \langle Q, X, C, E \rangle$ such that for all $(q, (g, \lambda), r, q') \in E$, r is a translation.

Note that even when considering TCM, the Counter Reachability Problem remains undecidable. Hence, if we want to obtain decidability, a solution is to restrict the translations, and in particular to forbid equality tests. This restriction comes down to using a timed version of Vector Addition Systems with States (VASS) [20],

or equivalently, Petri Nets. We hence recall the definition of Timed VASS, which is a model introduced in [18] [2] :

Definition 5.2 A *Timed VASS* (*TVASS* for short) is a TCM $\mathcal{S} = \langle Q, X, C, E\rangle$ such that for all $(q, (g, \lambda), r, q') \in E$, r is a translation $(\#, \mu, \delta)$ such that $\# = (\leq, \ldots, \leq)$.

5.2 Properties of a TCS and its Region Graph

Different restrictions can be done on Counter Machines to obtain decidability for the Counter Reachability Problem. First, remark that the restrictions we just introduced are obviously still true when considering the related region graph :

Proposition 5.3 *Let $\mathcal{S}$ be a TCS. If $\mathcal{S}$ is a TCM (resp. a TVASS), then the counter system $RG(\mathcal{S})$ is a counter machine (resp. a VASS).*

Since the counter reachability problem is decidable when considering VASS [21,22], from Theorem 4.4, we deduce that :

Theorem 5.4 *The Counter Reachability Problem is decidable for TVASS.*

The two definitions 5.1 and 5.2 are syntactical restrictions ; nonetheless, it is possible to restrict the behaviour of a TCS. We say that a pair $(\mathcal{S}, s_0)$ is an *intialized TCS* (resp. *intialized CS*), in which $\mathcal{S}$ is a TCS (resp. CS) and s_0 is an initial configuration of $TS(\mathcal{S})$ (resp. $CTS(\mathcal{S})$). Among the possible restrictions on its behaviour, we can consider bounded initialized TCS (resp. CS), for which there is a bound under which all the counter values stay, in all the possible executions. Then, from Theorem 4.4, we deduce that :

Proposition 5.5 *If an initialized TCS $(\mathcal{S}, s_0)$ is bounded, then the intialized counter system $(RG(\mathcal{S}), s'_0)$ is bounded, with $s_0 = (q, \mathbf{x}, \mathbf{c})$ and $s'_0 = (q_{\mathbf{x}}, \mathbf{c})$.*

The Counter Reachability Problem is obviously decidable for bounded initialized CS, since there is a finite number of reachable configurations ; thus, we deduce that :

Theorem 5.6 *The Counter Reachability Problem is decidable for bounded intialized TCS.*

Finally, we consider another restriction on the behaviour, but this time, only for TCM. In [19], the class of *reversal-bounded* counter machines has been introduced, and has been extended in [17]. This extension mentions that an intialized Counter Machine $(\mathcal{S}, s_0)$ is *k-reversal-b-bounded* for $k, b \in \mathbb{N}$, if in all the executions of $\mathcal{S}$ starting from s_0, each counter valuation alternates at most k times between non-increasing and non-decreasing modes over a bound b. We naturally extend this notion to TCM ; remark that an initialized TCM is reversal-bounded if it is k-reversal-b-bounded for some $k, b \in \mathbb{N}$. Then, thanks to Proposition 4.3, we deduce that :

[2] Actually, the emptiness problem of the language of a TVASS has been proved decidable in [18] and [8]

Proposition 5.7 *If an initialized TCM* $(\mathcal{S}, s_0)$ *is reversal-bounded, then the initialized counter machine* $(RG(\mathcal{S}), s'_0)$ *is reversal-bounded, with* $s_0 = (q, \mathbf{x}, \mathbf{c})$ *and* $s'_0 = (q_{\mathbf{x}}, \mathbf{c})$.

Since the Counter Reachability Problem is decidable for reversal-bounded counter machines [17], we have :

Theorem 5.8 *The Counter Reachability Problem is decidable for reversal-bounded initialized TCM.*

The following table summarizes the decidability results we obtained here :

Model	Region Graph	Counter Reachability
TCS	CS	Undecidable
TVASS	VASS	Decidable
Reversal-bounded TCM	Reversal-bounded CM	Decidable
Bounded TCS	Bounded CS	Decidable

Notice that TVASS is a recursive class, which is very interesting for implementation perspectives : hence, we propose an algorithm solving the Counter Reachability Problem for this class. However, it is impossible to decide if a system is reversal-bounded or bounded, in the general case.

Algorithm 1 : Solves the Counter Reachability Problem for TVASS.

Input : a TVASS $\mathcal{S}$, a configuration $(q, \mathbf{c})$, and an initial state s_0
Output : the answer to "Is there a $\mathbf{x}$ such that $(q, \mathbf{x}, \mathbf{c}) \in \text{Reach}(\mathcal{S}, s_0)$?"

```
build RG(S) = ⟨Γ, →_RG⟩
for all q'_x ∈ Γ do
  if q' = q then
    if (q_x, c) ∈ Reach_C(RG(S), s_0) then
      return True
    end if
  end if
end for
return False
```

6 Conclusion and Future work

We introduced a new model for systems mixing clocks and counters, and proved the Counter Reachability Problem to be decidable for three of its subclasses. Other subclasses might be interesting to study in order to broaden these results, such as flat TCS, following the approaches of [12] or [7]. Our ultimate goal is to extend the tool for counter systems FAST [5,6] so that it also handles clocks. Moreover, our main result, as stated in Theorem 4.4, can be extended to other dataypes than counters (e.g. pushdown stacks, lossy channels, etc...).

References

[1] Alur, R., C. Courcoubetis, T. A. Henzinger and P.-H. Ho, *Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems*, LNCS **736**, 1992, pp. 209–229.

[2] Alur, R. and D. L. Dill, *A theory of timed automata*, Theor. Comput. Sci. **126** (1994), pp. 183–235.

[3] Annichini, A., E. Asarin and A. Bouajjani, *Symbolic techniques for parametric reasoning about counter and clock systems*, in: *CAV*, LNCS **1855** (2000), pp. 419–434.

[4] Annichini, A., A. Bouajjani and M. Sighireanu, *TReX: A tool for reachability analysis of complex systems*, in: *CAV*, LNCS **2102** (2001), pp. 368–372.

[5] Bardin, S., A. Finkel, J. Leroux and L. Petrucci, *FAST: Fast Acceleration of Symbolic Transition systems.*, in: *CAV*, LNCS **2725** (2003), pp. 118–121.

[6] Bardin, S., A. Finkel, J. Leroux and L. Petrucci, *FAST: Acceleration from theory to practice*, International Journal on Software Tools for Technology Transfer (STTT) (2008), to appear.

[7] Bardin, S., A. Finkel, J. Leroux and P. Schnoebelen, *Flat acceleration in symbolic model checking*, in: *ATVA*, LNCS **3707**, 2005, pp. 474–488.

[8] Bérard, B., *Untiming timed languages*, Inf. Process. Lett. **55** (1995), pp. 129–135.

[9] Bérard, B., V. Diekert, P. Gastin and A. Petit, *Characterization of the expressive power of silent transitions in timed automata*, Fundamenta Informaticae **36** (1998), pp. 145–182.

[10] Bolognesi, T., F. Lucidi and S. Trigila, *From timed petri nets to timed lotos*, in: *PSTV* (1990), pp. 395–408.

[11] Comon, H. and Y. Jurski, *Multiple counters automata, safety analysis and presburger arithmetic*, in: *CAV*, LNCS **1427**, 1998, pp. 268–279.

[12] Comon, H. and Y. Jurski, *Timed automata and the theory of real numbers*, in: *CONCUR*, LNCS **1664**, 1999, pp. 242–257.

[13] Dang, Z., O. H. Ibarra, T. Bultan, R. A. Kemmerer and J. Su, *Binary reachability analysis of discrete pushdown timed automata*, in: *CAV*, LNCS **1855**, 2000, pp. 69–84.

[14] Dang, Z., O. H. Ibarra, P. S. Pietro and G. Xie, *Real-counter automata and their decision problems*, in: *FSTTCS*, LNCS **3328**, 2004, pp. 198–210.

[15] Darlot, C., A. Finkel and L. van Begin, *About FAST and TReX accelerations.*, Electronic Notes in Theoretical Computer Science **128** (2005), pp. 87–103.

[16] Finkel, A. and J. Leroux, *How to compose presburger-accelerations: Applications to broadcast protocols*, in: *FSTTCS*, LNCS **2556** (2002), pp. 145–156.

[17] Finkel, A. and A. Sangnier, *Reversal-bounded counter machines revisited*, in: *MFCS*, LNCS (2008), to appear.

[18] Gorrieri, R. and G. Siliprandi, *Real-time system verification using P/T nets*, in: *CAV*, LNCS **818** (1994), pp. 14–26.

[19] Ibarra, O. H., *Reversal-bounded multicounter machines and their decision problems*, J. ACM **25** (1978), pp. 116–133.

[20] Karp, R. M. and R. E. Miller, *Parallel program schemata*, J. Comput. Syst. Sci. **3** (1969), pp. 147–195.

[21] Kosaraju, S. R., *Decidability of reachability in vector addition systems (preliminary version)*, in: *STOC* (1982), pp. 267–281.

[22] Mayr, E. W., *An algorithm for the general Petri net reachability problem*, SIAM Journal on Computing **13** (1984), pp. 441–460.

[23] Merlin, P., "A study of the recoverability of computing systems," Ph.D. thesis, University of California, Irvine (1974).

[24] Minsky, M., "Computation: finite and infinite machines," Prentice-Hall, Inc., NJ, USA, 1967.

[25] Xie, G., Z. Dang, O. H. Ibarra and P. S. Pietro, *Dense counter machines and verification problems*, in: *CAV*, LNCS **2725**, 2003, pp. 93–105.