Model-Checking CTL* over Flat Presburger Counter Systems ⋆

Stéphane Demri[*] — Alain Finkel[*] — Valentin Goranko[**] — Govert van Drimmelen [***]

[*] *LSV, ENS Cachan, CNRS, INRIA*
61 av. Pdt. Wilson, 94235 Cachan Cedex, France
`{demri,finkel}@lsv.ens-cachan.fr`

[**] *Informatics and Mathematical Modelling, Technical University of Denmark*
Richard Petersens Plads, DK-2800 Kgs. Lyngby, Denmark
`vfgo@imm.dtu.dk`

[***] *Department of Mathematics, University of Johannesburg*
Private Bag X1, Auckland Park 2006, South Africa
`{govertvd}@uj.ac.za`

ABSTRACT. *This paper studies model-checking of fragments and extensions of CTL* on infinite-state counter systems, where the states are vectors of integers and the transitions are determined by means of relations definable within Presburger arithmetic. In general, reachability properties of counter systems are undecidable, but we have identified a natural class of admissible counter systems (ACS) for which we show that the quantification over paths in CTL* can be simulated by quantification over tuples of natural numbers, eventually allowing translation of the whole Presburger-CTL* into Presburger arithmetic, thereby enabling effective model checking. We provide evidence that our results are close to optimal with respect to the class of counter systems described above.*

KEYWORDS: *model-checking, infinite-state transition systems, Presburger arithmetic, CTL* , counter systems*

DOI:10.3166/JANCL.20.313–342 © 2010 Lavoisier, Paris

⋆. Supported by CNRS/NRF projects No 15469 and No 19812, RNTL project AVERILES, ANR projects AVERISS and VERIDYC.

1. Introduction

Background. Model-checking of infinite-state systems (for a survey see (Burkart *et al.*, 2001)) is a rapidly growing area of formal verification. It has been successfully applied to real-time and hybrid systems, concurrent systems, Petri nets, asynchronous communication devices (unbounded FIFO channels), infinite and unbounded data structures (counters, queues, lists), control systems, parameterized systems (networks of arbitrary number of processes), etc. The single most important property of practical interest in infinite-state transition systems is *state reachability* which is often undecidable in structures with otherwise decidable first-order theories, such as automatic structures (Khoussainov *et al.*, 1995). Therefore, intensive research has been devoted to identifying classes of finitely presentable infinite structures with decidable reachability and related properties.

Transition systems determined by relations definable in Presburger arithmetic provide a large natural class of infinite-state transition systems (Bardin *et al.*, 2005), suitable for modeling in various applications such as the TTP Protocol (embedded systems) (Bardin *et al.*, 2004), broadcast protocols (Esparza *et al.*, 1999), and programs with pointer variables (Bardin *et al.*, 2006a; Bouajjani *et al.*, 2006; Finkel *et al.*, 2009). Important cases of such transition systems with computable reachability have been established in (Ibarra, 1978; Fribourg *et al.*, 1997; Comon *et al.*, 1998; Finkel *et al.*, 2000; Finkel *et al.*, 2002). The method of acceleration for computing reachability sets has been developed in (Boigelot, 1998; Leroux, 2003) and is implemented in the verification tool FAST (Leroux, 2003; Bardin *et al.*, 2004; Bardin *et al.*, 2006b); see also the verification tools LASH (Boigelot, 1998) and TReX (Annichini *et al.*, 2001).

Motivation. For practical model-checking, an infinite-state system must be provided with an effective finitary presentation, and in particular, must admit a symbolic representation of sets of states and transitions. Such representations can be based on:

– automata (finite, pushdown, on infinite words or trees, etc.), as in pushdown graphs (Muller *et al.*, 1985), prefix-recognizable graphs (Caucal, 2003), and automatic structures (Blumensath *et al.*, 2004),
– interpretations into sufficiently rich infinite structures with respective decidable theories, e.g., again, automatic structures (Blumensath *et al.*, 2004) and tree-interpretable structures (Blumensath, 2002), (Caucal, 2003).
– algebraic equations or operations (Courcelle, 1990), etc.

Presburger arithmetic (PrA) is a logical formalism that is intrumental in many applications and it is a particularly appropriate platform for symbolic representation of a wide variety of infinite state systems, such as *counter systems* (see (Bardin *et al.*, 2003)) where vectors of integers are subjected to linear transformations according to a finite control graph. These strongly extend counter automata (Minsky, 1967) and even very simple examples of counter systems can have notoriously difficult and unpredictable behaviour, a witness being the Collatz problem (a.k.a. the Syracuse problem), see e.g. (Lagarias, 1985). An important and natural class of counter systems, in which various practical cases of infinite state systems can be modelled (e.g. broadcast protocols (Finkel *et al.*, 2002)), are those with a *flat* control graph, where no control location occurs in more than one simple cycle (see (Boigelot, 1998; Comon *et al.*, 1998; Comon *et al.*, 2000; Finkel *et al.*, 2002; Bardin *et al.*, 2003; Leroux, 2003; Leroux *et al.*, 2005; Bardin *et al.*, 2005; Bozga *et al.*, 2009)). Essential results on verifying safety and reachability properties on flat counter systems have been obtained in (Comon *et al.*, 1998; Finkel *et al.*, 2002). However, until recently such properties had not been considered in the framework of any formal specification language, and thus a natural question arises *to identify expressive logical languages in which formal specification and verification of properties of counter systems can be conducted*.

On the other hand, most of the studies on CTL*-model checking so far have been restricted to (unfoldings of) finite transition systems, and few decidability results for CTL*-model checking on essentially infinite-state systems are known (Finkel *et al.*, 1997; Bouajjani *et al.*, 1997). This is particularly surprising since CTL* is one of the most known applied non-classical logics among the temporal logics. Actually, most of these results are immediate consequences of stronger results about decidable modal mu-calculus, or even the whole monadic second-order logic (MSO) in such systems, see e.g. (Walukiewicz, 2001). Furthermore, these decidability results typically refer to the propositional CTL*, while model checking of first-order extensions of even much simpler temporal logics is typically undecidable. That is why, it is important *to search for larger classes of effectively generated infinite state systems (without necessarily decidable MSO), but in which natural first-order extensions of CTL* have decidable model-checking problems*.

Our contribution. In this paper, which is an improved and extended version of (Demri *et al.*, 2006), We jointly address both problems described above, and we obtain a nearly optimal solution of them. Our main contributions are the following:

1) We introduce an extension of CTL* (Emerson *et al.*, 1986) over Presburger arithmetic, i.e., where atomic propositions range over Presburger-definable sets of configuration states. We interpret that extension over Presburger counter systems (abbreviated by PCS), thus proposing a very powerful specification language for such systems. Presburger counter systems are infinite-state transition systems with states being vectors of integers (counter values) and transition relations definable in Presburger arithmetic. This class of models naturally includes the counter automata (or Minsky machines). Presburger counter systems are interesting in two complementary ways: they naturally arise in the reachability analysis of counter systems, and on the other hand they can be viewed as models for symbolic representation of infinite state transition systems.

2) We identify a class of Presburger counter systems for which the local model checking problem for the Presburger-CTL* is decidable. These are Presburger counter systems defined over flat control graphs with arcs labelled by transition functions defined by Presburger formulae, for which counting iteration over every cycle in the control graph is Presburger definable. A well-studied case when the latter condition is satisfied is when the composition monoids generated by the transition functions over

every cycle are finite (see (Finkel *et al.*, 2002)).

3) We show that the decidability results described above persist in some strong extensions of the Presburger-CTL*, i.e. with a class of temporal operators defined by means of constrained queue-content decision diagrams (aka CQDD) (see (Bouajjani *et al.*, 1999)) in a way analogous to Wolper's Extended temporal logic (Wolper, 1983).

4) We provide evidence that our results are close to optimal with respect to the class of Presburger counter systems described above, by showing that small relaxations of each of the conditions lead to undecidability. For example, by dropping either the counting iteration property or the flatness condition, undecidability is obtained.

Related work. Analyzing the reachability problem for counter systems is paramount for the verification of infinite-state systems, see e.g. (Ibarra *et al.*, 2000) (reversal-bounded systems), (Comon *et al.*, 1998) (flat systems), (Finkel *et al.*, 2002) (flat Presburger transition systems), (Dang *et al.*, 2003) (discrete timed automata), see also the decidability of reachability for classes of 2-counter systems (Finkel *et al.*, 2000). It is worth noting that, even though decidability can be obtained only at the cost of making drastic restrictions on counter systems, there is a natural class of counter systems that are sufficiently expressive for modelling different case studies and for which one may verify the safety properties by means of the effective computation of the reachability relation (Finkel *et al.*, 2002; Bardin *et al.*, 2003; Leroux, 2003; Leroux *et al.*, 2005; Bardin *et al.*, 2005). For instance, the flattable systems (Leroux *et al.*, 2005) admit a flat finite unfolding of the control graph with the same reachability set. On the logical side, temporal logics with Presburger constraints have been defined and investigated in (Čerāns, 1994; Bouajjani *et al.*, 1995; Bultan *et al.*, 1997; Comon *et al.*, 2000; Schuele *et al.*, 2004; Demri, 2006; Bruyère *et al.*, 2003), some of which have quite expressive decidable fragments. However, undecidability of the reachability problem can be proved for quite restricted counter systems, see e.g. (Cortier, 2002; Potapov, 2004) while at the same time very few classes of counter systems are decidable for CTL* (see e.g. (Finkel *et al.*, 1997) for one-counter systems). A logical formalism closer to the one developed in this paper is presented in (Bultan *et al.*, 1997) where an undecidable temporal logic with CTL-like operators and atomic formulae in Presburger arithmetic is introduced and the models are counter systems. The class of models is not restricted (hence decidability does not hold) but model-checking is performed by a symbolic analysis and an approximation algorithm. Interestingly, if we restrict ourselves to the same temporal operators, it is open whether our main decidability result can be established by giving up functionality. Model checking discrete timed automata with parametric timed CTL is also shown decidable by translation into Presburger arithmetic in (Bruyère *et al.*, 2003).

Structure and content of the paper. In Section 2 we present preliminary definitions about graphs and Presburger arithmetic. In Section 3 we introduce the class of Presburger counter systems (PCS) and we present the branching-time temporal logic FOPCTL$^\star(\mathrm{PrA})[n]$ whose models are transition systems generated from PCS. Admissible PCS are introduced in Section 4 and we recall (un)decidability results of the reachability problem for some classes of PCS. In Section 5, we show our main decidability result about model-checking admissible counter systems with FOPCTL$^\star(\mathrm{PrA})[n]$. Section 6 provides undecidability results indicating that our result in Theorem 20 is close to optimal.

In Section 7 we show the decidability of model-checking problems over admissible PCS even when CQDD-based temporal operators are added to the temporal logic. Section 8 contains concluding remarks and states open problems related to our results.

2. Preliminaries

Graphs, paths, cycles. A labelled graph $\mathcal{G} = \langle \Sigma, Q, E \rangle$ is a structure such that Q is a non-empty set, Σ is a non-empty finite alphabet and $E \subseteq Q \times \Sigma \times Q$. Graphs with a singleton alphabet are the standard graphs. As usual, $\langle q, a, q' \rangle \in E$ is also denoted by $q \xrightarrow{a} q'$. A *path* in $\mathcal{G}$ is a sequence $q_0 \xrightarrow{a_0} q_1 \ldots \xrightarrow{a_{n-1}} q_n$ such that for $i \in \{0, \ldots, n-1\}$, $q_i \xrightarrow{a_i} q_{i+1}$ is a transition. A *cycle* in a labelled graph is a closed path (where the initial and final vertices coincide) with no repeating edges. A *simple cycle* is a cycle in which the only repeated vertex is the initial (and final) vertex. Observe that herein we use notions about cycles a bit different from those in graph theory. Given a path $\lambda = q_0 \xrightarrow{a_0} q_1 \ldots \xrightarrow{a_{n-1}} q_n$, where each $q_i \in Q$, $a_i \in \Sigma$, we define the *length of* λ to be $|\lambda| = n$. A graph is *flat* if every cycle in it is a simple cycle; equivalently, if every vertex occurs in at most one cycle.

Presburger arithmetic. Presburger arithmetic is the first-order theory PrA of the structure $\langle \mathbb{N}, +, \leq \rangle$, well-known to be decidable (Presburger, 1929). However, all results in this paper will still hold in a more general setting, based on the structure $\langle \mathbb{Z}, +, \leq \rangle$ which is easily seen to be first-order interpretable into $\langle \mathbb{N}, +, \leq \rangle$, and therefore has a decidable first-order theory, too. For simplicity of notation, and with a benign abuse of terminology, hereafter we will refer to the first-order theory of $\langle \mathbb{Z}, +, \leq \rangle$ as Presburger arithmetic, too, and will use the same notation, PrA, for it. Given a Presburger formula $A(x_1, \ldots, x_n)$ with free variables in $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ and $\mathbf{a} = \langle a_1, \ldots, a_n \rangle \in \mathbb{Z}^n$, the truth of $A(x_1, \ldots, x_n)$ with respect to the assignment of values $\mathbf{a}$ to $\mathbf{x}$ is denoted by $\mathbf{a} \models_{\mathrm{PrA}} A(\mathbf{x})$. Elements of $\mathbb{Z}^n$ will be usually denoted by $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\ldots$ and vectors of variables will be denoted by $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, $\mathbf{t}$, $\ldots$, possibly decorated. A set $X \subseteq \mathbb{Z}^n$ is said to be *Presburger definable* iff there is a Presburger formula $A(\mathbf{x})$ with free variables $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ such that $X = \{\mathbf{a} \in \mathbb{Z}^n : \mathbf{a} \models_{\mathrm{PrA}} A(\mathbf{x})\}$. For $n > 0$, A *binary relation of dimension* n is a relation $R \subseteq \mathbb{Z}^n \times \mathbb{Z}^n$. Respectively, R is Presburger definable iff there is a Presburger formula $A(\mathbf{x}, \mathbf{x}')$ with free variables $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ and $\mathbf{x}' = \langle x'_1, \ldots, x'_n \rangle$ such that $R = \{\langle \mathbf{a}, \mathbf{a}' \rangle \in \mathbb{Z}^n \times \mathbb{Z}^n : \mathbf{a}, \mathbf{a}' \models A(\mathbf{x}, \mathbf{x}')\}$.

DEFINITION 1. — *Let f be a partial function from $\mathbb{Z}^n$ to $\mathbb{Z}^n$ with domain* $\mathrm{dom}(f)$.

– *f is a* translation *if there exists $\mathbf{b} \in \mathbb{Z}^n$ such that for every $\mathbf{a} \in \mathrm{dom}(f)$ we have* $f(\mathbf{a}) = \mathbf{a} + \mathbf{b}$.

– f *is* affine *if there exist a matrix* $\mathbf{A} \in \mathbb{Z}^{n \times n}$ *and* $\mathbf{b} \in \mathbb{Z}^n$ *such that for every* $\mathbf{a} \in \mathrm{dom}(f)$ *we have* $f(\mathbf{a}) = \mathbf{A}\mathbf{a} + \mathbf{b}$.
– f *is* Presburger definable *iff the graph of* f *is a Presburger definable relation.*

3. Temporal Logics on Presburger Counter Systems

In this section, we introduce Presburger counter systems and a first-order extension of the temporal logic CTL* interpreted over such systems.

3.1. *Presburger Counter Systems*

The Presburger transition systems defined below are infinite state transition systems that can be finitely described by formulae in Presburger arithmetic.

When infinite state transition systems arise in the modeling of computational processes, there is often a natural factoring of each system state into a control component and a memory component, where the set of control states (locations) is typically finite. We refer to the combined state of the system, containing the location, the memory state and the position of the head, as a *configuration* of the system.

We will be interested in systems where the memory states are n-dimensional vectors of integers. In particular, we define systems where the transition relation on such vectors may be described by relations definable in Presburger arithmetic.

DEFINITION 2. — *A* Presburger counter system (PCS) of dimension n *is a labelled graph* $\mathcal{C} = \langle \Sigma, Q, \delta, n \rangle$*, where*

– Σ *is a finite set of Presburger formulae of the form* $A(\mathbf{x}, \mathbf{x}')$ *where* $\mathbf{x}$ *and* $\mathbf{x}'$ *are tuples of* n *variables,*
– Q *is a finite set of locations,*
– $\delta \subseteq Q \times \Sigma \times Q$ *is the transition relation.*

By convention, prime variables in $\mathbf{x}'$ *are intended to be interpreted as the next-state values of the unprimed variables in* $\mathbf{x}$*.*

Given two locations q and q', we write $A_{q,q'}(\mathbf{x}, \mathbf{x}')$ to denote the disjunction of all the formulae $B(\mathbf{x}, \mathbf{x}')$ such that $\langle q, B(\mathbf{x}, \mathbf{x}'), q' \rangle \in \delta$. Thus, without any loss of generality, we can assume that there is a unique transition between every two control states. When there is no transition between a pair of states in the counter system, we introduce one labelled by falsum $\perp$.

Thus, a PCS can be regarded as a labelled graph with alphabet made of specific Presburger formulae.

Figure 1 contains a simple Presburger counter system, augmented with an initial location and final location.

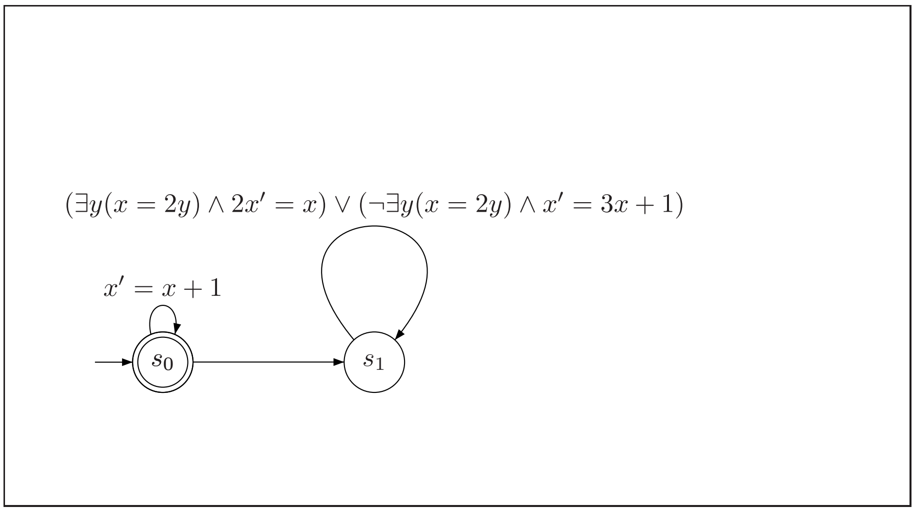

$(\exists y(x = 2y) \wedge 2x' = x) \vee (\neg \exists y(x = 2y) \wedge x' = 3x + 1)$
$x' = x + 1$
s_0
s_1

Figure 1. *A simple Presburger system*

Every PCS $\mathcal{C} = \langle \Sigma, Q, \delta \rangle$ of dimension n naturally induces a *Presburger transition system (of dimension n)*: $S_{\mathcal{C}} = \langle S, \rightarrow \rangle$ where $S = Q \times \mathbb{Z}^n$ is a set of *configurations* and $\langle q, \mathbf{a} \rangle \rightarrow \langle q', \mathbf{a}' \rangle$ iff $\mathbf{a}, \mathbf{a}' \models_{\mathrm{PrA}} A_{q,q'}(\mathbf{x}, \mathbf{x}')$. As usual, $\rightarrow^*$ denotes the reflexive and transitive closure of the relation $\rightarrow$. Whenever, $\langle q, \mathbf{a} \rangle \rightarrow^* \langle q', \mathbf{a}' \rangle$, we say that $\langle q', \mathbf{a}' \rangle$ is *reachable* from $\langle q, \mathbf{a} \rangle$. Without any loss of generality, we can assume that $Q \subseteq \mathbb{N}$, hence $S \subseteq \mathbb{Z}^{n+1}$. Depending on the context, the configurations of $S_{\mathcal{C}}$ will be written as $\mathbf{a} = \langle q, a_1, \ldots, a_n \rangle$ (location encoded in the first element of $\mathbf{a}$) or simply as $\langle q, \mathbf{a} \rangle \in Q \times \mathbb{Z}^n$. A *configuration path* in $\mathcal{C}$ is an infinite path in $S_{\mathcal{C}}$.

We say that:

– $\mathcal{C}$ is *functional*, if for all q, q', the formula $A_{q,q'}(\mathbf{x}, \mathbf{x}')$ defines a partial function.
– a functional PCS $\mathcal{C}$ is a *counter automaton,* if for all q, q', $A_{q,q'}(\mathbf{x}, \mathbf{x}')$ defines a translation.
– a functional PCS $\mathcal{C}$ is *affine* if for all q, q', $A_{q,q'}(\mathbf{x}, \mathbf{x}')$ defines an affine function.

PROPOSITION 3. — *Each of the following properties of Presburger counter system: being functional, translation (i.e., a counter automaton), or affine, is definable in PrA, and therefore decidable.*

PROOF 4. — Let $A(\mathbf{x}, \mathbf{x}')$ be a Presburger formula over the free variables $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ and $\mathbf{x}' = \langle x'_1, \ldots, x'_n \rangle$. It is immediate to check that:

– $A(\mathbf{x}, \mathbf{x}')$ is functional iff

$$\models_{\mathrm{PrA}} \forall \mathbf{x} \forall \mathbf{y} \forall \mathbf{y}' ((A(\mathbf{x}, \mathbf{y}) \wedge A(\mathbf{x}, \mathbf{y}')) \Rightarrow (\mathbf{y} = \mathbf{y}')).$$

– $A(\mathbf{x}, \mathbf{x}')$ is a translation iff

$$\models_{\mathrm{PrA}} \bigwedge_{i=1}^{n} \exists z \forall \mathbf{x} \forall \mathbf{y} (A(\mathbf{x}, \mathbf{y}) \Rightarrow y_i = x_i + z).$$

To check whether $A(\mathbf{x}, \mathbf{x}')$ is affine requires a bit more work. We want to check the existence of a matrix $\mathbf{A} \in \mathbb{Z}^{n \times n}$ and a vector $\mathbf{b} \in \mathbb{Z}^n$ such that for every $\mathbf{a} \in \mathrm{dom}(f)$ we have $f(\mathbf{a}) = \mathbf{A}\mathbf{a} + \mathbf{b}$, where, $f(\mathbf{a})$ is the unique $\mathbf{a}'$ such that $\mathbf{a}, \mathbf{a}' \models A(\mathbf{x}, \mathbf{y})$. The solution below is a bit more complex than the straightforward approach. Indeed, $f(\mathbf{0}) = \mathbf{b}$, which allows to define $\mathbf{b}$. A similar reasoning would allow to compute each column of $\mathbf{A}$ by applying f to unit elements. However, this is not sufficient since f is partial and for instance $f(\mathbf{0})$ may be undefined. A less straighforward approach is described below. Here is how this can be done.

– First, note that $\mathrm{dom}(f)$ is Presburger definable by the formula $\exists \mathbf{y} A(\mathbf{x}, \mathbf{y})$.
– Therefore, $\mathrm{dom}(f)$ can be defined as a finite union of sets of the form

$$S_i = \{\mathbf{b}_i + \lambda_1 \mathbf{p}_{i,1} + \cdots + \lambda_{n_i} \mathbf{p}_{i,n_i} : \lambda_1, \ldots, \lambda_{n_i} \in \mathbb{N}\}$$

where $\mathbf{b}_i, \mathbf{p}_{i,1}, \ldots, \mathbf{p}_{i,n_i} \in \mathbb{Z}^n$ (the basis and the periods). All these integers can be effectively computed (Ginsburg *et al.*, 1966).

Suppose the union has K sets. If f is affine, then $f(\mathbf{a}) = \hat{\mathbf{A}}\mathbf{a} + \hat{\mathbf{b}}$ for some $\hat{\mathbf{A}}$ and $\hat{\mathbf{b}}$. Then for every $1 \leq i \leq K$ and $1 \leq j \leq n_i$, there is a unique integer vector $\mathbf{c_{ij}}$ $(= \hat{\mathbf{A}}\mathbf{p}_{j,n_j})$ such that for every $\mathbf{z} \in S_i$ we have that

$$f(\mathbf{z} + \mathbf{p}_{j,n_j}) - f(\mathbf{z}) = \mathbf{c_{ij}}. \qquad (*)$$

(Indeed: $f(\mathbf{z} + \mathbf{p}_{j,n_j}) - f(\mathbf{z}) = \hat{\mathbf{A}}(\mathbf{z} + \mathbf{p}_{j,n_j}) + \hat{\mathbf{b}} - (\hat{\mathbf{A}}\mathbf{z} + \hat{\mathbf{b}}) = \hat{\mathbf{A}}\mathbf{z} + \hat{\mathbf{A}}\mathbf{p}_{j,n_j} + \hat{\mathbf{b}} - \hat{\mathbf{A}}\mathbf{z} - \hat{\mathbf{b}} = \hat{\mathbf{A}}\mathbf{p}_{j,n_j} = \mathbf{c_{ij}}$.)

The existence of such unique vector can be easily expressed by a Presburger formula and then verified. If false, then f is not affine. If true, then $\hat{\mathbf{A}}$ and $\hat{\mathbf{b}}$ must, in particular, satisfy the equations:

$$\hat{\mathbf{A}}\mathbf{p}_{j,n_j} \quad = f(\mathbf{b_i} + \mathbf{p}_{j,n_j}) - f(\mathbf{b_i}) \text{ for every } 1 \leq i \leq K \text{ and } 1 \leq j \leq n_i. \qquad (**)$$

$$\hat{\mathbf{A}}\mathbf{b_i} + \hat{\mathbf{b}} = f(\mathbf{b_i}) \text{ for every } 1 \leq i \leq K. \qquad (***)$$

This is a system of linear equations with integer coefficients for the $n^2 + n$ integer entries of the matrix $\hat{\mathbf{A}}$ and the vector $\hat{\mathbf{b}}$. To find an integer solution of that system, or to show that there is none, one can use e.g., the method from (Papadimitriou, 1981) or (Borosh *et al.*, 1976). If there is no integer solution, then such matrix and vector do not exist, so f is not affine. If there is an integer solution, take any one; it determines a matrix $\mathbf{A}$ and a vector $\mathbf{b}$.

We can now check that $f(\mathbf{a}) = \mathbf{A}\mathbf{a} + \mathbf{b}$ for any $\mathbf{a} \in \mathrm{dom}(f)$. Indeed, using the equations (*), (**), and (***) we have that $f(\mathbf{a}) = f(\mathbf{b}_i + \Sigma_j \lambda_j \mathbf{p}_{i,j}) = f(\mathbf{b}_i) + \Sigma_j \lambda_j \mathbf{A}\mathbf{p}_{i,j} = \mathbf{A}\mathbf{b}_i + \mathbf{b} + \Sigma_j \lambda_j \mathbf{A}\mathbf{p}_{i,j} = \mathbf{A}(\mathbf{b}_i + \Sigma_j \lambda_j \mathbf{p}_{i,j}) + \mathbf{b} = \mathbf{A}\mathbf{a} + \mathbf{b}$. ■

3.2. ***The Temporal Logic FOPCTL***$^\star(\mathrm{PrA})$

We now define a version FOPCTL$^\star(\mathrm{PrA})$ of first-order and past-time extension of CTL* that is appropriate for reasoning about Presburger transition systems. The name 'FOPCTL$^\star(\mathrm{PrA})$' indicates that FOPCTL$^\star(\mathrm{PrA})$ contains past-time operators, first-order quantification over integers and its underlying temporal logic is CTL*. The logic FOPCTL$^\star(\mathrm{PrA})$ differs from standard CTL* with past mainly in the definition of atomic formulae. Whereas propositional variables are used in the propositional CTL*, we will use as atomic formulae in FOPCTL$^\star(\mathrm{PrA})$ Presburger definable predicates, interpreted on the set of configurations.

We introduce a countable set of individual variables, say VAR $= \{y_0, y_1, y_2 \ldots\}$, for quantification over counter values. Elements of VAR are distinct from the distinguished ones in $\{x_0, x_1, \ldots, x_n\}$ that are free variables, only interpreted by the values of counters on configurations (the control location being encoded by the interpretation of x_0). In order to match the dimension of the models where such formulae will be interpreted, the Presburger definable predicates must have a matching number of free variables, thus giving a family of logics FOPCTL$^\star(\mathrm{PrA})[n]$ parameterized by the dimension $n \geq 1$. When the dimension n is clear from the context, we just refer to FOPCTL$^\star(\mathrm{PrA})$.

Atomic formulae of FOPCTL$^\star(\mathrm{PrA})[n]$ are Presburger formulae of the form $\theta(\mathbf{x}, \mathbf{y})$ where $\mathbf{x} = x_0, x_1, \ldots, x_n$ and $\mathbf{y}$ is a vector of variables from VAR, regarded as parameters.

Formulae of FOPCTL$^\star(\mathrm{PrA})[n]$ are defined as follows:

$$\varphi \overset{\text{def}}{=} \theta(\mathbf{x}, \mathbf{y}) \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathtt{X}\varphi \mid \varphi\mathtt{U}\varphi \mid \mathtt{X}^{-1}\varphi \mid \varphi\mathtt{S}\varphi \mid \mathtt{A}\ \varphi \mid \exists\ y\ \varphi,$$

where $y \in$ VAR and $\mathbf{y}$ is a sequence of variables. We shall freely use standard abbreviations for the implication $\Rightarrow$, the existential path quantifier $\mathtt{E}$, the always operator $\mathtt{G}$, and the sometimes operator $\mathtt{F}$.

The LTL fragment of FOPCTL$^\star(\mathrm{PrA})$, denoted by FOLTL(Pr), consists of formulae of the form either $\mathtt{E}\ \phi'$ or $\mathtt{A}\ \phi'$ where ϕ' has no path quantifiers and no past-time operators. We define the *strict EF fragment* of FOPCTL$^\star(\mathrm{PrA})$ as the set of FOPCTL$^\star(\mathrm{PrA})$ formulae containing only the temporal operator $\mathtt{E\ F}$ and no nested occurrences of $\mathtt{E\ F}$. Hence, this fragment has no past-time operators either.

We will give semantics of FOPCTL$^\star(\mathrm{PrA})$ over Presburger transition systems. The satisfaction relation $\models$ is parameterized by an *environment* ρ that is a map VAR $\to \mathbb{Z}$, in order to interpret the free variables from VAR that occur in formulae (the map ρ will be omitted when not immediately relevant). For a PCS $\mathcal{C} = \langle \Sigma, Q, \delta, n \rangle$ with Presburger transition system $S_{\mathcal{C}} = \langle S, \to \rangle$, the satisfaction relation $\models^\rho$ is defined at position i of configuration path π as follows, where $\pi_{\leq i}$ denotes the initial part of π up to and including position i; the environment ρ will be omitted wherever it is not essential.

– $\pi, i \models^{\rho} \theta(\mathbf{x}, \mathbf{y})$ iff $\pi(i), \rho \models \theta(\mathbf{x}, \mathbf{y})$ in PrA, where $\pi(i)$ provides the interpretation of the variables $x_0, \ldots, x_n$ and ρ the interpretation for the variables in $\mathbf{y}$,

– $\pi, i \models \neg\varphi$ iff $\pi, i \not\models \varphi$,

– $\pi, i \models \varphi \wedge \varphi'$ iff $\pi, i \models \varphi$ and $\pi, i \models \varphi'$,

– $\pi, i \models \mathtt{X}\varphi$ iff $\pi, i+1 \models \varphi$,

– $\pi, i \models \varphi \mathtt{U} \varphi'$ iff there is some $j \geq i$ such that $\pi, j \models \varphi'$ and for each k, if $i \leq k < j$ then $\pi, k \models \varphi$,

– $\pi, i \models \mathtt{X}^{-1}\varphi$ iff $i > 0$ and $\pi, i-1 \models \varphi$,

– $\pi, i \models \varphi \mathtt{S} \varphi'$ iff there is some $j \leq i$ such that $\pi, j \models \varphi'$ and for each k, if $j < k \leq i$ then $\pi, k \models \varphi$,

– $\pi, i \models \mathtt{A}\ \varphi$ iff for every infinite configuration path π' such that $\pi'_{\leq i} = \pi_{\leq i}$ we have $\pi', i \models \varphi$,

– $\pi, i \models^{\rho} \exists y \varphi$ iff there is an integer $m \in \mathbb{Z}$ such that $\pi, i \models^{\rho[y \leftarrow m]} \varphi$ where $\rho[y \leftarrow m]$ is the environment obtained from ρ by forcing y to be interpreted by m.

Past-time operators are known to simplify the expressions of specifications, see e.g. (Laroussinie *et al.*, 2000). Here is an example of formula with a past-time operator:

$$\mathtt{A}\ \mathtt{G}\ (x_1 = x_2 \Rightarrow \mathtt{F}^{-1} x_3 = x_4)$$

The forthcoming translation will treat future-time and past-time temporal operators uniformly.

First-order quantification over counter values allows us to state many interesting properties in FOPCTL$^\star(\mathrm{PrA})$:

Determinism: For all the configurations reachable from the initial configuration, there is at most one successor configuration:

$$\mathtt{A}\ \mathtt{G} \bigwedge_{0 \leq i \leq n} \neg\exists y (\mathtt{E}\ \mathtt{X}(x_i = y) \wedge \mathtt{E}\ \mathtt{X}(x_i \neq y)).$$

Boundedness: The set of configurations reachable from the initial configuration is finite:

$$\exists y, y'\ \mathtt{A}\ \mathtt{G} \bigwedge_{1 \leq i \leq n} y \leq x_i \leq y'.$$

Increasing chain: On some path the first counter strictly increases at every step:

$$\mathtt{E}\ \mathtt{G} \exists y\ (y = x_1 \wedge \mathtt{X}(x_1 > y)).$$

3.3. *Model checking problems for FOPCTL*$^\star(\mathrm{PrA})[n]$

In the definition of model-checking problems below, the formulae in FOPCTL$^\star(\mathrm{PrA})[n]$ satisfy that none of the variables in VAR occur out of the scope of a quantification. Of course, variables related to counter values and locations (those in $\mathbf{x}$) occur freely, In that way, we can sometimes omit the environments when interpreting formulae with all variables in VAR bounded. We will call such formulae *semi-closed*. In that way, we do not need to specify an environment in the statement below.

1) LOCAL MODEL CHECKING: Given a PCS $\mathcal{C}$ with Presburger transition system $S_{\mathcal{C}} = \langle S, \rightarrow \rangle$, a configuration $\langle q, \mathbf{a} \rangle \in S$, and a semi-closed formula φ from $\mathrm{FOPCTL}^{\star}(\mathrm{PrA})[n]$, determine whether $\mathcal{C}, \langle q, \mathbf{a} \rangle \models \varphi$, meaning that for every path π such that $\pi(0) = \langle q, \mathbf{a} \rangle$, we have $\pi, 0 \models \varphi$.
The dual version of this problem with the existential quantification over paths can be defined in a similar fashion. In the rest of the paper, we only deal with the universal version but a similar treatment is possible for the existential version, too.

2) GLOBAL MODEL CHECKING: Given a PCS $\mathcal{C}$ with Presburger transition system $S_{\mathcal{C}} = \langle S, \rightarrow \rangle$, and a $\mathrm{FOPCTL}^{\star}(\mathrm{PrA})[n]$ formula φ, compute (as a Presburger formula) the set of configurations S such that for every path π with $\pi(0) \in S$, we have $\pi, 0 \models \varphi$.

3) VALIDITY CHECKING WITH AN INITIAL CONDITION: Given a PCS $\mathcal{C}$ with Presburger transition system $S_{\mathcal{C}} = \langle S, \rightarrow \rangle$, a Presburger formula $A_0(\mathbf{x})$ and a semi-closed $\mathrm{FOPCTL}^{\star}(\mathrm{PrA})[n]$ formula φ, check whether for every configuration $\langle q, \mathbf{a} \rangle$ satisfying $A_0(\mathbf{x})$, for every configuration $\langle q', \mathbf{a}' \rangle$ reachable from $\langle q, \mathbf{a} \rangle$, we have $\mathcal{C}, \langle q', \mathbf{a}' \rangle \models \varphi$.

Variants of these problems can be defined by considering subclasses of PCS or other specification languages.

4. Admissible Presburger Counter Systems

As we will show later, local model checking of $\mathrm{FOPCTL}^{\star}(\mathrm{PrA})$ over the whole class of PCSs is highly undecidable (by reduction from the recurring problem for nondeterministic Minsky machines (Minsky, 1967; Alur *et al.*, 1994)) even though reachability can be decided for many classes of counter systems, see e.g. (Ibarra *et al.*, 2000; Comon *et al.*, 1998; Finkel *et al.*, 2002; Dang *et al.*, 2003). In this section we introduce a subclass of *admissible* PCS in which model checking $\mathrm{FOPCTL}^{\star}(\mathrm{PrA})$ will be proved to be decidable in the next section.

DEFINITION 5. — *Given a relation $R \subseteq \mathbb{Z}^n \times \mathbb{Z}^n$ we define the* counting iteration of *R as the relation $R_{\mathbf{CI}} \subseteq \mathbb{Z}^n \times \mathbb{N} \times \mathbb{Z}^n$ such that $\langle \mathbf{a}, i, \mathbf{b} \rangle \in R_{\mathbf{CI}}$ iff $\langle \mathbf{a}, \mathbf{b} \rangle \in R^i$. R* has a Presburger counting iteration *if its counting iteration is Presburger definable.*

The *cycle relation* R^{λ} of a cycle λ in a PCS is obtained by composing the transition relations on the cycle. According to Section 2, a cycle λ can be viewed as a sequence $t_1, \ldots, t_{\alpha}$ of transitions of the form $t_i = q_i \xrightarrow{A_i} q_i'$ such that for $1 \leq i \leq \alpha - 1$, $q_{i+1} = q_i'$ and $q_1 = q_{\alpha}'$. We define the relation R^{t_i} as the set of pairs $\{\langle \langle q_i, \mathbf{a} \rangle, \langle q_i', \mathbf{a}' \rangle \rangle : \mathbf{a}, \mathbf{a}' \models_{\mathrm{PrA}} A_i(\mathbf{x}, \mathbf{x}')\}$. The relation R^{λ} is then the composition $R^{t_1} \circ \cdots \circ R^{t_{\alpha}}$. A cycle has the *Presburger counting iteration property* if its cycle relation has a Presburger counting iteration.

DEFINITION 6. — *A PCS $\mathcal{C}$ has the* Presburger counting iteration property *if every cycle in the control graph of $\mathcal{C}$ has that property.*

Observe that if a PCS $\mathcal{C}$ has the Presburger counting iteration property, we can effectively identify the Presburger formula associated with each cycle. It is sufficient to enumerate Presburger formulae $A(\mathbf{x}, i, \mathbf{y})$ and test whether

$$\forall \mathbf{x}, \mathbf{x}', i\ (A(\mathbf{x}, i, \mathbf{x}') \Rightarrow i \geq 0) \wedge (A(\mathbf{x}, 0, \mathbf{x}') \Leftrightarrow (\mathbf{x} = \mathbf{x}')) \wedge$$
$$(A(\mathbf{x}, i+1, \mathbf{x}') \Leftrightarrow (\exists \mathbf{x}''\ A(\mathbf{x}, i, \mathbf{x}'') \wedge A'(\mathbf{x}'', \mathbf{x}')))$$

is valid, where $A'(\mathbf{x}, \mathbf{y})$ is the effect of a given cycle. This is an instance of a more general result from (Leroux, 2006). Indeed, given a Presburger-definable binary relation $R \subseteq \mathbb{Z}^n \times \mathbb{Z}^n$, it is undecidable to determine whether the transitive and reflexive closure R^* is Presburger-definable too (Leroux, 2006). We also know that there exist Presburger counter systems of dimension 1 that do not have the Presburger counting iteration property (for instance, consider the update $x_1' = 2x_1$). In general, we expect that determining whether a counter system has a Presburger counting iteration is an undecidable problem by extending similar results from (Leroux, 2006). By contrast, given a total affine function $f(x) = \mathbf{A}x + \mathbf{b}$, by (Boigelot, 1998), $\{\langle \mathbf{x}, \mathbf{A}\mathbf{x} + \mathbf{b} \rangle : \mathbf{x} \in \mathbb{Z}^n\}^*$ is Presburger-definable iff $\{\mathbf{A}^n : n \in \mathbb{N}\}$ is finite. Following (Finkel *et al.*, 2002), $\{\mathbf{A}^n : n \in \mathbb{N}\}$ is finite iff $\{\langle \mathbf{x}, \mathbf{A}\mathbf{x} + \mathbf{b} \rangle : \mathbf{x} \in \mathbb{Z}^n\}$ has the Presburger counting iteration. Finiteness of the monoid generated from $\mathbf{A}$ has been also considered in (Emerson *et al.*, 1998). Indeed, the broadcast protocols introduced in (Emerson *et al.*, 1998) use monotone affine transition functions of the form $f(x) = \mathbf{A}x + \mathbf{b}$ where $\{\mathbf{A}^n : n \in \mathbb{N}\}$ is also finite. In (Emerson *et al.*, 1998), it is shown how to compute the least upper bound of $f^n(x)$ in order to construct coverability graphs. Nevertheless, this fact is not used in order to compute the *exact* value of the acceleration.

As pointed out in (Finkel *et al.*, 2002), *flatness* of the control graph is a key property enabling the symbolic computation of the reachability relation. That property ensures that there is only a finite number of 'schemes' of configuration paths (see details later on) in the PCS, and since one can effectively compute Presburger formulae associated with cycle relations, we obtain the following.

PROPOSITION 7. — *(Comon* et al.*, 1998; Finkel* et al.*, 2002) For every flat PCS satisfying the Presburger counting iteration property, one can effectively compute the reachability relation $\rightarrow^*$ for the transition system $S_{\mathcal{C}} = \langle S, \rightarrow \rangle$ by means of a formula in Presburger arithmetic.*

This proof of this folklore result is quite straightforward. Now, we will provide a sufficient condition for the Presburger counting iteration property. First, we need to recall a few definitions. The transitions in an affine PCS are of the form $s \xrightarrow{\mathbf{x}'=\mathbf{A}\mathbf{x}+\mathbf{b}} t$ where $\mathbf{A} \in \mathbb{Z}^{n \times n}$ and $\mathbf{b} \in \mathbb{Z}^n$.

A cycle λ has the *finite monoid property* if the multiplicative monoid of $\mathbf{A}_\lambda$ is finite where $\mathbf{A}_\lambda = \mathbf{A_1} \cdots \mathbf{A_N}$ and the cycle λ is labelled by the sequence of matrices $\mathbf{A_1} \cdots \mathbf{A_N}$.

DEFINITION 8. — *A PCS $\mathcal{C}$* has the finite monoid property *if every cycle in the control graph of $\mathcal{C}$ has that property.*

Let us remark that our definition of a PCS having the finite monoid property is weaker than the one in (Finkel *et al.*, 2002) in which a PCS has the finite monoid property if the multiplicative monoid generated from *all* the matrices occurring in the PCS is finite. Our weaker condition is sufficient to obtain the following result:

PROPOSITION 9. — *(Finkel* et al.*, 2002; Boigelot, 2003) Every flat and affine PCS with the finite monoid property has the Presburger counting iteration property.*

As a corollary of Propositions 7 and 9, the Presburger formula defining the reachability relation in every flat and affine PCS with the finite monoid property is effectively computable. By contrast, observe that in (Comon *et al.*, 1998), even though flatness is also assumed, the transition relations are not necessarily functional. Hence, the above-mentioned consequence appear to be incomparable with the main result from (Comon *et al.*, 1998). Furthermore, the systems defined in Definition 10 below are more general than the ones in (Comon *et al.*, 1998; Bozga *et al.*, 2009) since we allow a richer language on transitions.

Finally, we require functionality of the transition relation, in order to ensure effective enumeration within Presburger arithmetic of all configuration paths in the PCS. That condition is not always necessary and can be relaxed in various ways, but that will not be discussed in the paper. Let us mention that decidability still holds true if the transitions that do not belong to cycles are non-functional.

DEFINITION 10. — *An* admissible Presburger counter system (ACS) *is a flat, functional PCS, that has the Presburger counting iteration property.*

In particular, due to Proposition 9, every flat and affine PCS with the finite monoid property is admissible. As we will see further, relaxing any of the conditions for admissibility leads to undecidability, even of the simple reachability problem.

In order to conclude this section, it is worth recalling that acceleration of a loop is understood as the computation of the effect of the infinite iteration and to symbolically represent this effect with a regular language, e.g., with a finite-state automaton. The first reference to acceleration of loops in counter systems and its representation by formula appeared in the seminal paper (Boigelot *et al.*, 1994). The authors accelerate loops of counter systems labelled by an affine function $f(x) = \mathbf{A}x + \mathbf{b}$ where $\mathbf{A}$ is a diagonal matrix in $\{0,1\}^{n\times n}, \mathbf{b} \in \mathbb{Z}^n$ and the domain is given by a set of linear inequalities. In (Boigelot *et al.*, 1994), acceleration is represented by periodic sets that can be expressed by Presburger formulae. Since $\mathbf{A}^2 = \mathbf{A}$, the infinite iteration can be indeed represented by periodic sets. In (Boigelot, 1998; Boigelot, 2003), this result is extended to loops labelled by affine functions $f(x) = \mathbf{A}x + \mathbf{b}$ such that $\{\mathbf{A}^n : n \in \mathbb{N}\}$ is finite . A rather more complex but equivalent version is given whose domain is given by a set of linear inequalities (i.e., a domain defined by a Presburger formula without quantifiers and modulo). In (Finkel *et al.*, 2002), this is extended to Presburger-definable domains.

5. Model-Checking of FOPCTL$^\star(\mathrm{PrA})[n]$ on Admissible Counter Systems

Herein, we show decidability of model checking FOPCTL$^\star(\mathrm{PrA})$ over admissible Presburger counter systems. The main idea behind our decidability result is the following: in an ACS there are only finitely many 'path schemas', and although each of these generates a possibly infinite set of configuration paths, the configuration paths for each path schema can be uniformly encoded within Presburger arithmetic by finite vectors of integer parameters. Thus, the quantification over paths in FOPCTL$^\star(\mathrm{PrA})[n]$ can be simulated by quantification over tuples of natural numbers, eventually allowing translation of FOPCTL$^\star(\mathrm{PrA})[n]$ into PrA.

Throughout this section, let $\mathcal{C} = \langle \Sigma, Q, \delta \rangle$ be an ACS of dimension n. Recall that we also assume that there is at most one transition between any two locations, by taking the disjunction of all formulae labelling the transitions between every pair of locations.

5.1. *Control paths and configuration paths*

DEFINITION 11. — *A* control path *in $\mathcal{C}$ is any infinite path in the graph of $\mathcal{C}$. A* path segment *in $\mathcal{C}$ is a single transition $t \in \delta$ or a simple cycle in $\mathcal{C}$, that we represent as a finite sequence of locations. A* path schema *in $\mathcal{C}$ is a sequence $\langle \sigma_0, \ldots, \sigma_k \rangle$ of path segments in $\mathcal{C}$ such that:*

1) for every $0 \leq i \leq k-1$, the last location of σ_i is the first location of σ_{i+1},
2) no single transition σ_i occurs in a cycle σ_j for $j > i$,
3) the final path segment σ_k is a cycle.
4) for $i \neq j$, we have $\sigma_i \neq \sigma_j$.

Cycles in a path schema that are not the final segment are called interior cycles *of the schema.*

The idea behind the definition above is that it allows for a *unique* description of every control path in the graph of $\mathcal{C}$. Condition (4.) allows to get a concise description.

From now on we fix an enumeration $\lambda_1, \ldots, \lambda_M$ of all the cycles in $\mathcal{C}$ and assume that $M > 0$.

In Figure 2, we present an example of an ACS (the transitions on the figure that are not labelled are assigned arbitrary functional Presburger formulae). We give below examples of control paths, path segments and path schemata in the ACS with the following convention: a simple transition is encoded by a pair of the form $\langle q, q' \rangle$ and a cycle is encoded by a sequence $\langle q, \ldots, q' \rangle$ such that $q = q'$.

simple cycles: $\lambda_1 = \langle q_1, q_3, q_6, q_1 \rangle$ (see dotted arrows in Figure 2), $\lambda_2 = \langle q_4, q_5, q_4 \rangle$, $\lambda_3 = \langle q_7, q_7 \rangle$

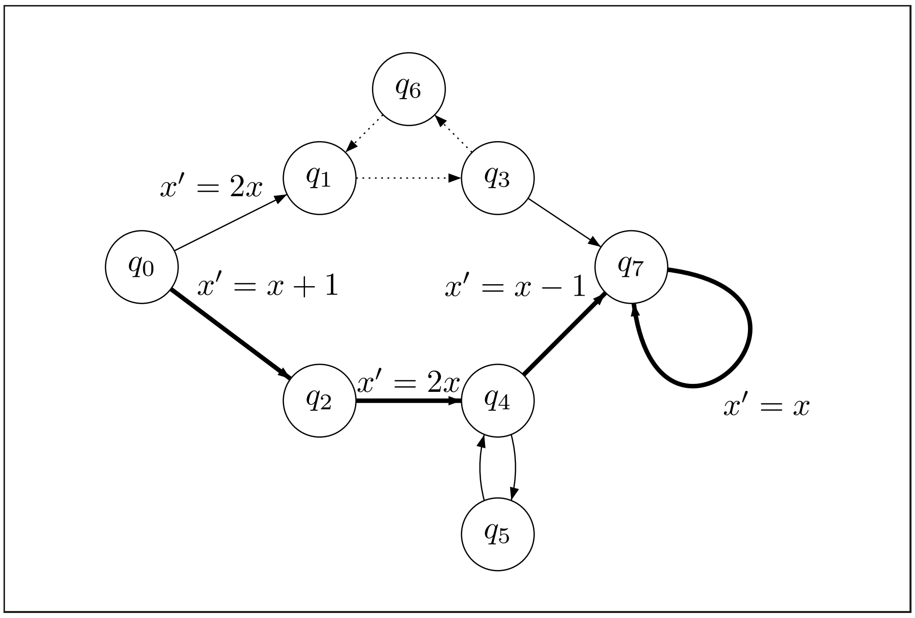


Figure 2. *A flat counter system*

control path: $q_0 q_2 q_4 q_7^\omega$ (see the bold arrows in Figure 2).

path segments: $\langle q_0, q_1\rangle$, $\langle q_1, q_3, q_6, q_1\rangle$, $\langle q_4, q_5, q_4\rangle$, $\langle q_5, q_4, q_5\rangle$, $\langle q_1, q_3\rangle$.

valid path schema: $\langle q_0, q_1\rangle, \langle q_1, q_3, q_6, q_1\rangle, \langle q_1, q_3\rangle, \langle q_3, q_7\rangle, \langle q_7, q_7\rangle$.

invalid path schema: $\langle q_0, q_1\rangle, \langle q_1, q_3\rangle, \langle q_3, q_6, q_1, q_3\rangle, \langle q_3, q_7\rangle, \langle q_7, q_7\rangle$ (Condition (2) in Definition 11 is violated).

Note that the last two path schemas above describe the same control path, but the latter violates condition 2 of the definition: the single transition $\langle q_1, q_3\rangle$ also occurs in a cycle that follows after it: $\langle q_3, q_6, q_1, q_3\rangle$.

Since an ACS is flat and has a finite number of locations, the following holds:

PROPOSITION 12. — *In every ACS $\mathcal{C}$ with at most one transition between two locations, the number of path schemata is bounded by $(N)^N$ where $N = |Q| + |\delta|$.*

PROOF 13. — The number of path segments is bounded by $|Q|$ (bound on the number of simple cycles) $+|\delta|$ (bound on the number of simple transitions). Hence, the number of path schemata is bounded by N^N. ■

Hereafter, we suppose that there are $P \geq 1$ path schemas in $\mathcal{C}$. A path schema with at least one interior cycle corresponds to infinitely many different control paths, since any interior cycle in the schema may be repeated an arbitrary number of times on the control path. The number of repetitions of a given cycle in a control path is called the *cycle count* of that cycle. Thus, every control path is completely characterized by its underlying path schema and the cycle counts for its interior cycles. The next definition formalizes this idea.

DEFINITION 14. — *Let the ACS $\mathcal{C}$ have $M > 0$ cycles and P path schemas. A* cycle count vector $\mathbf{c}$ *is a tuple $\langle c_1, \ldots, c_M \rangle \in \mathbb{N}^M$, where c_r represents the cycle count for the cycle λ_r. A* control path description α *is a pair $\alpha = \langle p, \mathbf{c} \rangle$ where $p \in \{1, \ldots, P\}$ denotes the path schema, $\mathbf{c}$ is the cycle count vector for the control path being described, $c_i > 0$ for every interior cycle λ_i and $c_i = 0$ for any cycle λ_i in $\mathcal{C}$ which is not interior in the path schema p. Hereafter a control path description, may be written as $\langle p, c_1, \ldots, c_M \rangle$. We write α_0 for the path schema associated with control path description α.*

Note that in the definition above p is simply the identifier of the control path α_0.

The following is immediate from the flatness condition on ACS.

PROPOSITION 15. — *For every control path in an ACS $\mathcal{C}$, there is a unique control path description.*

So, we can encode control paths by tuples of positive integers. Without risk of confusion, we identify every control path with its description. For example, in the system on Figure 2, the description of the control path $q_0 q_1 q_3 (q_6 q_1 q_3)^3 q_7^\omega$ with underlying path schema $\langle q_0, q_1 \rangle, \langle q_1, q_3, q_6, q_1 \rangle, \langle q_1, q_3 \rangle, \langle q_3, q_7 \rangle, \langle q_7, q_7 \rangle$, labelled by 1, is $\langle 1, \langle 3, 0, 0 \rangle \rangle$.

Every configuration path in an ACS is uniquely described by the pair $\langle \alpha, \langle q, \mathbf{a} \rangle \rangle$ where α is its control path and $\langle q, \mathbf{a} \rangle$ is the initial configuration. Conversely, due to the functionality of $\mathcal{C}$, every such pair $\langle \alpha, \langle q, \mathbf{a} \rangle \rangle$ with location of $\mathbf{a}$ corresponding to the first location of the path schema α_0, describes a unique path in the configuration graph starting at $\langle q, \mathbf{a} \rangle$, and progressing according to the transitions of the control path α. Note, however, that such a path may terminate and therefore not be considered as a configuration path.

In the example on Figure 2, from the control path $q_0 q_2 q_4 q_7^\omega$ and the initial configuration with $x = 3$, we obtain the configuration path

$$\langle q_0, 3 \rangle \langle q_2, 4 \rangle \langle q_4, 8 \rangle \langle q_7, 7 \rangle^\omega .$$

5.2. *Encoding the configurations along a path by a Presburger formula*

In this section we construct a Presburger formula that exactly describes the configuration path associated with a control path and initial configuration. As a corollary of Theorem 16 below, we obtain Proposition 7.

THEOREM 16. — *Given an ACS $\mathcal{C}$ of dimension n with $M > 0$ cycles, one can compute a Presburger formula $PathConfig_{\mathcal{C}}(\mathbf{v}, \mathbf{x}, i, \mathbf{y})$ such that for all $\alpha \in \mathbb{N}^{M+1}$, $\mathbf{a} \in \mathbb{Z}^{n+1}$, $m \in \mathbb{N}$ and $\mathbf{b} \in \mathbb{Z}^{n+1}$:*

$$\alpha, \mathbf{a}, m, \mathbf{b} \models PathConfig_{\mathcal{C}}(\mathbf{v}, \mathbf{x}, i, \mathbf{y})$$

iff α is a valid control path description and the m^{th} configuration of the configuration path $\langle \alpha, \mathbf{a} \rangle$ is $\mathbf{b}$ ($\mathbf{v}$, $\mathbf{x}$ and $\mathbf{y}$ are variable sequences and i is a variable).

PROOF 17. — Sketch: First, when a cycle λ has the Presburger counting iteration property, we write $\varphi_\lambda(\mathbf{x}, y, \mathbf{x}')$ to denote the Presburger formula encoding its counting iteration relation. In that case, there is also a Presburger formula $A_\lambda(\mathbf{x}, k, \mathbf{x}')$ that expresses that $\mathbf{x}'$ is obtained from $\mathbf{x}$ by following k transitions along the cycle λ.

Now, we will construct a formula *PathConfig*$_{\mathcal{C}}$ in accordance with the requirements of the theorem.

First, let P be the number of path schemas in $\mathcal{C}$. We consider each path schema p individually, constructing a formula $SchemaConfig_p(\mathbf{v}, \mathbf{x}, i, \mathbf{y})$ such that for all $\alpha \in \mathbb{N}^{M+1}$, $\mathbf{a} \in \mathbb{Z}^{n+1}$ (encoding a configuration), $m \in \mathbb{N}$ and $\mathbf{b} \in \mathbb{Z}^{n+1}$ it is the case that $\alpha, \mathbf{a}, m, \mathbf{b} \models SchemaConfig_p(\mathbf{v}, \mathbf{x}, i, \mathbf{y})$ iff α is a control path description, the path schema of the control path α is p, and the m^{th} configuration of the configuration path $\langle \alpha, \mathbf{a} \rangle$ is $\mathbf{b}$. Then our desired formula *PathConfig*$_{\mathcal{C}}$ will be the disjunction over all $SchemaConfig_p$ where p is a path schema in the system.

To define *SchemaConfig*$_p$ for a fixed path schema p, we proceed as follows. Suppose $\langle \sigma_0, \ldots, \sigma_k \rangle$ is the sequence of segments in p, and α is a control path with path schema p. Along the unique (if it exists) configuration path induced by α starting with configuration $\mathbf{a}$, we will identify some *landmark positions* and *landmark configurations*: for each segment σ_j (where $0 \leq j < k$) we would like to identify the position $t_j \in \mathbb{N}$ and the configuration $\mathbf{w_j} \in \mathbb{Z}^{n+1}$ immediately before the segment σ_j is traversed (or entered for the first time, if the segment is a cycle).

The landmarks associated with segment σ_0 are the initial position $t_0 = 0$ and the initial configuration $\mathbf{w_0} = \mathbf{a}$.

We define κ_j to be the number of positions in the configuration path that are covered by segment σ_j. If segment σ_j is a single transition then the number of positions covered by that segment is 1. Otherwise, the segment σ_j is some interior cycle λ_r with $|\lambda_r|$ transitions in the cycle. Recall that the number of times the cycle λ_r is traversed in the control path described by $\mathbf{v}$ is given by the cycle count $\mathbf{v}_r$. Then, the total number of positions in the configuration path covered by the segment σ_j is $\mathbf{v}_r|\lambda_r|$. Formally:

– if σ_j is a transition $\langle q, A(\mathbf{x}, \mathbf{x}'), q' \rangle$ then $\kappa_j \overset{\text{def}}{=} 1$,
– if σ_j is a cycle λ_r then $\kappa_j \overset{\text{def}}{=} \mathbf{v}_r|\lambda_r|$.

Having defined κ_j, we can now state that our landmark positions t_i will thus satisfy the following constraints: $t_0 = 0$ and $t_{j+1} = t_j + \kappa_j$ for $0 \leq j < k$.

Next, we consider the landmark configuration that corresponds to each landmark position. We would like to describe in a uniform way those configurations that appear while a specific segment is traversed. So, for the segment σ_j, we define a Presburger formula $\mathit{SegmentConfig}_j(\mathbf{x}, i, \mathbf{y})$ such that for all $\mathbf{a} \in \mathbb{Z}^{n+1}$, $m \in \mathbb{N}$, and $\mathbf{b} \in \mathbb{Z}^{n+1}$, it is the case that $\mathbf{a}, m, \mathbf{b} \models \mathit{SegmentConfig}_j(\mathbf{x}, i, \mathbf{y})$ iff the location of the configuration $\mathbf{a}$ appears as one of the locations in σ_j, and the configuration $\mathbf{b}$ is reached from configuration $\mathbf{a}$ after m transitions, according to the transition(s) of σ_j.

When segment σ_j is just a single transition, we define $\mathit{SegmentConfig}_j$ using the transition relation. Otherwise, if the segment is a simple cycle, we use the corresponding counting iteration relation.

Formally, we define *SegmentConfig*$_j$ as follows:

– if σ_j is a transition $t = \langle q, A(\mathbf{x}, \mathbf{x}'), q' \rangle$ then

$$\mathit{SegmentConfig}_j(\mathbf{x}, i, \mathbf{y}) \stackrel{\text{def}}{=} x_0 = q \wedge ((i = 0 \wedge \mathbf{x} = \mathbf{y}) \vee (i = 1 \wedge A(\mathbf{x}, \mathbf{y}))),$$

– if σ_j is a simple cycle λ then

$$\mathit{SegmentConfig}_j(\mathbf{x}, i, \mathbf{y}) \stackrel{\text{def}}{=} A_\lambda(\mathbf{x}, i, \mathbf{y}).$$

We define the string of quantifiers

$$\mathit{ExistLandmarks} \stackrel{\text{def}}{=} \exists t_0, \ldots, \exists t_k, \exists \mathbf{w_0}, \ldots, \exists \mathbf{w_k}$$

and define the formula

$$\begin{aligned}\mathit{LandmarkConstraints} \stackrel{\text{def}}{=}\ & (t_0 = 0 \wedge \mathbf{w_0} = \mathbf{x}) \\ & \bigwedge_{j=0}^{k-1} [(t_{j+1} = t_j + \kappa_j) \wedge \mathit{SegmentConfig}_j(\mathbf{w_j}, \kappa_j, \mathbf{w_{j+1}})].\end{aligned}$$

If the configuration path is infinite, we are assured that such landmarks exist, hence the formula that claims their existence will be true. Conversely, to ensure that the path in the Presburger transition system will be infinite, we will extend the formula to confirm the existence of configurations in all positions after the last landmark:

$$\mathit{CheckInfinite} \stackrel{\text{def}}{=} \forall t(t_k < t \rightarrow \exists \mathbf{z}\ \mathit{SegmentConfig}_k(\mathbf{x}, t - t_k, \mathbf{z})).$$

The final part of our construction of *SchemaConfig*$_p$ is to include a subformula that checks for the occurrence of a given configuration at a given position of the configuration path. We have to take some care to check whether position i occurs in a segment before the final cycle segment is entered, or inside the final cycle.

$$\begin{aligned}\mathit{CheckConfig} \stackrel{\text{def}}{=} & \bigwedge_{j=0}^{k-1} [(t_j \leq i \wedge i < t_{j+1}) \rightarrow \mathit{SegmentConfig}_j(\mathbf{w_j}, i - t_j, \mathbf{y})] \\ & \wedge [(t_k \leq i) \rightarrow \mathit{SegmentConfig}_k(\mathbf{w_k}, i - t_k, \mathbf{y})].\end{aligned}$$

The above formulae are now combined to give the configuration checking formula for path schema p

$$\begin{aligned}&\mathit{SchemaConfig}_p(\mathbf{v},\mathbf{x},i,\mathbf{y}) \overset{\text{def}}{=} (\xi_0 = p)\wedge\\ &\quad \mathit{ExistLandmarks}[\mathit{LandmarkConstraints} \wedge \mathit{CheckInfinite} \wedge \mathit{CheckConfig}].\end{aligned}$$

Finally we have:

$$\mathit{PathConfig}_{\mathcal{C}}(\mathbf{v},\mathbf{x},i,\mathbf{y}) \overset{\text{def}}{=} \bigvee_{p=1}^{M} \mathit{PathConfig}_p(\mathbf{v},\mathbf{x},i,\mathbf{y}).$$

■

We define two auxiliary formulae that will be used in the following proof. Firstly, we can check that a pair $(\mathbf{v},\mathbf{x})$ denotes a valid configuration path, by checking that the initial configuration of the path is correct and that the path is infinite:

$$\mathit{ValidPath}\,(\mathbf{v},\mathbf{x}) \overset{\text{def}}{=} \mathit{PathConfig}_{\mathcal{C}}(\mathbf{v},\mathbf{x},0,\mathbf{x}) \wedge \forall i \geq 0 \exists \mathbf{z} \mathit{PathConfig}_{\mathcal{C}}(\mathbf{v},\mathbf{x},i,\mathbf{z})$$

Secondly, for two configuration paths denoted by $(\mathbf{v},\mathbf{x})$ and $(\mathbf{v}',\mathbf{y})$ we would like to express that the paths agree on all configurations up to and including position i. To this end, we construct the formula

$$\begin{aligned}&\mathit{CommonPathPrefix}(\mathbf{v},\mathbf{x},\mathbf{v}',\mathbf{y},i) \overset{\text{def}}{=}\\ &\qquad \forall j \geq 0[j \leq i \Rightarrow \forall \mathbf{z}(\mathit{PathConfig}_{\mathcal{C}}(\mathbf{v},\mathbf{x},j,\mathbf{z}) \Leftrightarrow \mathit{PathConfig}(\mathbf{v}',\mathbf{y},j,\mathbf{z}))],\end{aligned}$$

This formula will be used when quantifying over paths with identical finite past.

5.3. *A decision procedure to verify an admissible counter system*

We are now ready to show that model-checking FOPCTL$^\star(\mathrm{PrA})[n]$ can be reduced to satisfiability in Presburger arithmetic.

THEOREM 18. — *Given an ACS $\mathcal{C}$ of dimension n with Presburger transition system $S_{\mathcal{C}} = \langle S, \rightarrow\rangle$, for every semi-closed FOPCTL$^\star(\mathrm{PrA})[n]$ formula φ, one can compute a Presburger formula $A_\varphi(\mathbf{x})$ such that for every $\langle q,\mathbf{a}\rangle \in S_{\mathcal{C}}$, $\langle q,\mathbf{a}\rangle \models A_\varphi(\mathbf{x})$ iff $\mathcal{C},\langle q,\mathbf{a}\rangle \models \varphi$ (no need for environment since φ is semi-closed).*

PROOF 19. — We show that, given an ACS $\mathcal{C}$, for every FOPCTL$^\star(\mathrm{PrA})[n]$ formula φ, one can define a Presburger formula $T(\langle \mathbf{v},\mathbf{x},i\rangle;\varphi)$ with free variables $\mathbf{v},\mathbf{x},i$ such that $\alpha,\mathbf{a},m \models T(\langle \mathbf{v},\mathbf{x},i\rangle;\varphi)$ iff for the configuration path π with control path α and initial configuration $\mathbf{a}$, we have that $\pi, m \models \varphi$, if such configuration path exists.

We define T recursively on φ as follows:

$T(\langle \mathbf{v}, \mathbf{x}, i\rangle; \theta(\mathbf{x}, \mathbf{y})) \overset{\text{def}}{=} \forall \mathbf{z}[\mathit{PathConfig}_{\mathcal{C}}(\mathbf{v}, \mathbf{x}, i, \mathbf{z}) \Rightarrow \theta(\mathbf{z}, \mathbf{y})]$;

$T(\langle \mathbf{v}, \mathbf{x}, i\rangle; \neg\varphi) \overset{\text{def}}{=} \neg T(\langle \mathbf{v}, \mathbf{x}, i\rangle; \varphi)$;

$T(\langle \mathbf{v}, \mathbf{x}, i\rangle; \varphi \wedge \varphi') \overset{\text{def}}{=} T(\langle \mathbf{v}, \mathbf{x}, i\rangle; \varphi) \wedge T(\langle \mathbf{v}, \mathbf{x}, i\rangle; \varphi')$;

$T(\langle \mathbf{v}, \mathbf{x}, i\rangle; \mathtt{X}\varphi) \overset{\text{def}}{=} \exists j[(j = i+1) \wedge T(\langle \mathbf{v}, \mathbf{x}, j\rangle; \varphi)]$;

$T(\langle \mathbf{v}, \mathbf{x}, i\rangle; \varphi \mathtt{U} \varphi') \overset{\text{def}}{=} \exists j((j \geq i \wedge [T(\langle \mathbf{v}, \mathbf{x}, j\rangle, \varphi') \wedge \forall k(i \leq k < j \Rightarrow T(\langle \mathbf{v}, \mathbf{x}, k\rangle, \varphi)]))$;

$T(\langle \mathbf{v}, \mathbf{x}, i\rangle; \mathtt{X}^{-1}\varphi) \overset{\text{def}}{=} i > 0 \wedge \exists j[(i = j+1) \wedge T(\mathbf{v}, \mathbf{x}, j, \varphi)]$;

$T(\langle \mathbf{v}, \mathbf{x}, i\rangle; \varphi \mathtt{S} \varphi') \overset{\text{def}}{=} \exists j((j \leq i \wedge [T(\langle \mathbf{v}, \mathbf{y}, j\rangle; \varphi') \wedge \forall k(j < k \leq i \Rightarrow T(\langle \mathbf{v}, \mathbf{x}, k\rangle; \varphi)]))$;

$T(\langle \mathbf{v}, \mathbf{x}, i\rangle; \mathtt{A}\ \varphi) \overset{\text{def}}{=} \forall \mathbf{v}', \mathbf{y}[\mathit{CommonPathPrefix}(\mathbf{v}, \mathbf{x}, \mathbf{v}', \mathbf{y}, i) \Rightarrow T(\langle \mathbf{v}', \mathbf{y}, i\rangle; \varphi)]$;

$T(\langle \mathbf{v}, \mathbf{x}, i\rangle; \exists\, y\ \varphi) \overset{\text{def}}{=} \exists\, y\ T(\langle \mathbf{v}, \mathbf{x}, i\rangle; \varphi)$.

The formula $A_{\varphi}(\mathbf{x})$ is defined as follows:

$$A_{\varphi}(\mathbf{x}) \overset{\text{def}}{=} \forall\ \mathbf{v}(\mathit{ValidPath}(\mathbf{v}, \mathbf{x}) \Rightarrow T(\langle \mathbf{v}, \mathbf{x}, 0\rangle; \varphi)).$$

■

Note that, for a fixed ACS, the size of $A_{\varphi}(\mathbf{x})$ is linear in the size of φ. However, when the ACS is not fixed, presently we have no way to measure the size of $A_{\varphi}(\mathbf{x})$ in function of the size of the ACS. Indeed, we have no measure on the size of the Presburger formulae witnessing the Presburger counting iteration property.

THEOREM 20. — *The following problems for FOPCTL**(PrA) *restricted to ACSs are decidable: local model checking, global model checking, validity checking with an initial configuration.*

PROOF 21. — Indeed, in order to decide the local model checking problem, it is sufficient to check whether $\langle q, \mathbf{a}\rangle \models A(\mathbf{x})$ holds true where the Presburger formula $A(\mathbf{x})$ is computed from the proof of Theorem 18. Global model checking can be solved by computing precisely the formula $A(\mathbf{x})$ and testing Presburger validity. Finally, by Proposition 7, there is a Presburger formula $A'(\mathbf{x}, \mathbf{x}')$ computing the reachability relation in the configuration graph of some ACS. In order to solve validity checking by an initial condition $A_0(\mathbf{x})$, it is sufficient to check Presburger validity of the formula $\forall\ \mathbf{x}, \mathbf{x}'\ (A_0(\mathbf{x}) \wedge A'(\mathbf{x}, \mathbf{x}') \Rightarrow A(\mathbf{x}'))$. ■

Theorem 20 can be extended to systems and temporal logics such that PrA is replaced by any decidable extension PrA^{+} of PrA, closed under first-order quantification and Boolean operators, obtained by adding new predicates. The notion of Presburger counter system is extended by allowing transitions labelled by elements of PrA^{+}. Similarly, FOPCTL$^{\star}(\mathrm{PrA}^{+})$ is obtained from FOPCTL$^{\star}(\mathrm{PrA})$ by allowing atomic formulae from PrA^{+}. The model-checking problems for FOPCTL$^{\star}(\mathrm{PrA}^{+})$ are defined as for FOPCTL$^{\star}(\mathrm{PrA})$. Finally, the notions of counting iteration property and admissible counter systems are defined with PrA^{+} instead of PrA.

THEOREM 22. — *The following problems for FOPCTL*$^{\star}(\mathrm{PrA}^{+})$ *restricted to ACSs are decidable: local model checking, global model checking, validity checking with an initial configuration.*

6. Testing the boundaries of decidable model checking in Presburger counter systems

Here we give some results and examples indicating that our result in Theorem 20 is close to optimal. Before that, call a PCS *piecewise-affine* whenever each transition is labelled by a disjunction of expressions of the form $\theta(\mathbf{x}) \wedge \mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{b}$.

PROPOSITION 23. — *The reachability problem is not decidable in any of the following classes:*

1) all flat affine PCSs;
2) all affine PCSs with the finite monoid property (even counter automata);
3) all flat piecewise-affine PCSs with a single location.

PROOF 24. —

(1) Follows from results in (Cortier, 2002) about very basic control graphs but having cycles without the Presburger counting iteration property.

(2) Follows from undecidability of the halting problem for Minsky machines (Minsky, 1967).

(3) Follows from (Minsky, 1967), too. As a matter of fact, any counter automaton can be encoded as a flat piecewise-affine PCS with a single location q_0. Indeed, suppose that $q \xrightarrow{x:=x+1} q'$ is a transition in the counter automaton with the integer n [resp. n'] attached to q [resp. q'], then the unique transition in the piecewise-affine PCS is of the form $q_0 \xrightarrow{(x_0=n\wedge x_0'=n'\wedge x'=x+1)\vee\ldots} q_0$. There is an obvious correspondence between the transitions in the original counter automaton and the number of disjuncts in the Presburger formula labelling the unique transition. ■

To show how close to optimal our class of ACSs is, we give below an undecidability result for a fixed PCS $\mathcal{C}_u$ that is almost an ACS, but not flat. It is obtained from an ACS by only adding a reset transition while preserving the Presburger counting iteration property and functionality (see Figure 3).

$\mathcal{C}_u$ is of dimension 4, with counters x_1, x_2 and x_3, x_0 is the additional counter representing the location, and "id" denotes the identity function on the counters x_1, x_2 and x_3.

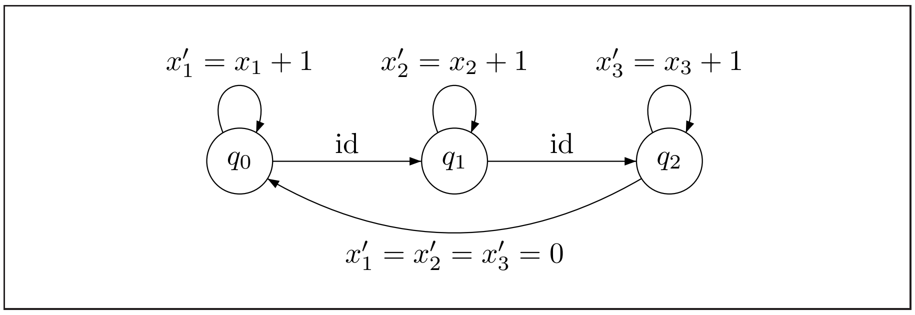


Figure 3. *Almost an admissible counter system*

THEOREM 25. — *Local model-checking on $\mathcal{C}_u$ with FOLTL(Pr)*[3] *is Σ^1_1-hard (highly undecidable).*

PROOF 26. — The proof is by reducing the recurrence problem for nondeterministic 2-counter machines that is shown Σ^1_1-hard in (Alur *et al.*, 1994). A nondeterministic 2-counter machine M consists of two counters C_1 and C_2, and a sequence of $n \geq 1$ instructions. The $\mathbf{k}$-th instruction is written as one of the following:

$\mathbf{k}$: $C_i := C_i + 1$; goto $\mathbf{k}_1$ or goto $\mathbf{k}_2$.

$\mathbf{k}$: if $C_i = 0$ then goto $\mathbf{k}_0$ else $C_i := C_i - 1$; goto $\mathbf{k}_1$ or goto $\mathbf{k}_2$.

We represent the configurations of M by triples $\langle c_1, c_2, l\rangle$ where $1 \leq l \leq n$, $c_1 \geq 0$ and $c_2 \geq 0$. A computation of M is a finite sequence of related configurations, starting with the initial configuration $\langle 0, 0, 1\rangle$ (location encoded as last element). The recurrence problem can be stated as the existence of an infinite execution that passes through the instruction 1 infinitely often. We shall build a formula φ of FOLTL(Pr)[3] such that M visits 1 infinitely often iff $\langle q_2, \langle 0, 0, 1\rangle\rangle \models \varphi$. The formula φ is of the form

$$\mathtt{E}\,(\mathtt{GF}(x_3 = 1) \wedge \bigwedge_{1 \leq \mathbf{k} \leq n} \mathtt{G}\varphi'_{\mathbf{k}}),$$

where $\varphi'_{\mathbf{k}}$ encodes the $\mathbf{k}$-th instruction. For instance, the $\mathbf{k}$-th instruction "$C_1 := C_1 + 1$; goto $\mathbf{k}_1$ or goto $\mathbf{k}_2$" is encoded by

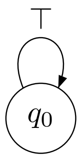
$$\forall y, z\, (x_1 = y \ \wedge\ x_2 = z \ \wedge\ x_3 = \mathbf{k} \ \wedge\ \mathtt{X}(x_0 = 0)) \Rightarrow$$

$$\mathtt{X}(\neg(\mathtt{X}(x_0 = 0))\,\mathtt{U}\,(\mathtt{X}(x_0 = 0) \ \wedge\ \overbrace{x_1 = y + 1}^{\text{increase } \mathrm{C}_1} \wedge x_2 = z \ \wedge\ (x_3 = \mathbf{k}_1 \vee x_3 = \mathbf{k}_2))).$$

Other instructions can be encoded similarly. ■

It is worth mentioning that $\mathcal{C}_u$ can be simulated by an ACS ('flattened') in a sense preserving the reachability sets, and therefore the strict EF fragment of FOLTL(Pr)[3] has a decidable local model-checking problem for $\mathcal{C}_u$.

Furthermore, by using the idea in the proof of Theorem 25 one can show that FOLTL(PrA)[3] has an undecidable local model-checking problem for the PCS described in Figure 4 (that is flat, has the Presburger counting iteration but is not functional) with variables x_1, x_2, x_3, and x_0 representing the location, where $\top$ denotes the truth constant.



This PCS has a unique transition that accepts any update of the counters x_1, x_2 and x_3. Hence, any sequence $\mathbb{N} \rightarrow \{q_0\} \times \mathbb{Z}^3$ is an infinite configuration path of this PCS. By way of example, as done above, the $\mathbf{k}$-th instruction "$C_1 := C_1 + 1$; goto $\mathbf{k}_1$ or goto $\mathbf{k}_2$" is encoded by

$$\forall\, y, z\, (x_1 = y \;\wedge\; x_2 = z \;\wedge\; x_3 = \mathbf{k}) \;\Rightarrow$$
$$\mathtt{X}(x_1 = y + 1 \wedge x_2 = z \wedge (x_3 = \mathbf{k}_1 \vee x_3 = \mathbf{k}_2)).$$

7. Decidable Extension of FOPCTL$^\star(\mathrm{PrA})[n]$ with CQDD Patterns

We present below an extension of FOPCTL$^\star(\mathrm{PrA})[n]$ for which model-checking over ACS can still be encoded into Presburger satisfiability.

In (Wolper, 1983) Wolper extends linear-time temporal logic LTL to an extended temporal logic that has the same expressive power as Büchi automata. In this section, we similarly extend the set of path formulae from FOPCTL$^\star(\mathrm{PrA})[n]$ by allowing temporal operators defined by another class of language acceptors, namely the CQDD (*constrained queue-content decision diagrams*) (Bouajjani *et al.*, 1999). This formalism has been introduced for symbolically representing infinite sets of configurations in FIFO automata – our use of CQDD is different. Non-regular languages can be defined with CQDD; moreover, the model-checking problem for LTL augmented with operators defined from CQDD is undecidable (Demri *et al.*, 2009), unlike the extension with regular languages (Wolper, 1983). The proof of that result is inspired from the undecidability proof of propositional dynamic logic (PDL) augmented with programs over the context-free language $\{a_1^n \cdot a_2 \cdot a_1^n : n \geq 0\}$ (see (Harel *et al.*, 2000, Chapter 9)). This context-free language can be easily recognized by a CQDD. By contrast, we show that the model-checking problem for FOPCTL$^\star(\mathrm{PrA})[n]$ extended with CQDD-based operators is decidable over ACS. Decidability is regained due to the flatness restriction in CQDD. Hence, in this section we show evidence that we can take advantage of flatness both in models *and* in formulae.

Before introducing the formal definition for CQDDs, let us mention that CQDD are finite-state automata attached to Presburger formulae that provide constraints on the number of times transitions are taken. Moreover, the underlying graphs of CQDDs is flat by definition.

A *CQDD* is a structure $\mathcal{A} = \langle \Sigma, S, S_0, E, l, A(y_1, \ldots, y_m), F\rangle$ such that:

– Σ is a finite set of symbols (the alphabet),
– S is a finite set of states,
– $S_0 \subseteq S$ is the set of initial states,
– $E \subseteq S \times \Sigma \times S$ is a set of transitions of cardinality m and $\langle S, \Sigma, E\rangle$ is flat,
– $F \subseteq S$ is a set of final (or accepting) states,
– l is a bijection from E to $\{1, \ldots, m\}$,
– $A(y_1, \ldots, y_m)$ is a Presburger formula.

An *accepting run* for the word $\sigma = a_0a_1a_2 \ldots a_{k-1}$ is a sequence $q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \ldots \xrightarrow{a_{k-1}} q_k$ such that

– $q_0 \in S_0$, $q_k \in F$ (the standard acceptance conditions for finite-state automata),
– for every $i \in \{0, \ldots, k-1\}$, $\langle q_i, a_i, q_{i+1}\rangle \in E$,
– $n_1, \ldots, n_m \models A(y_1, \ldots, y_m)$ in Presburger arithmetic, where each n_i is the number of occurrences of the transition $l^{-1}(i)$ in the sequence (alternatively, $(n_1, \ldots, n_m)$ is the Parikh image of σ).

The word σ is also said to be accepted by the automaton $\mathcal{A}$. We write $\mathrm{L}(\mathcal{A})$ to denote the set of words accepted by $\mathcal{A}$.

Figure 4 presents a CQDD with its constraint on the number of occurrences (each transition is related to a unique letter and to a unique variable in the constraint).

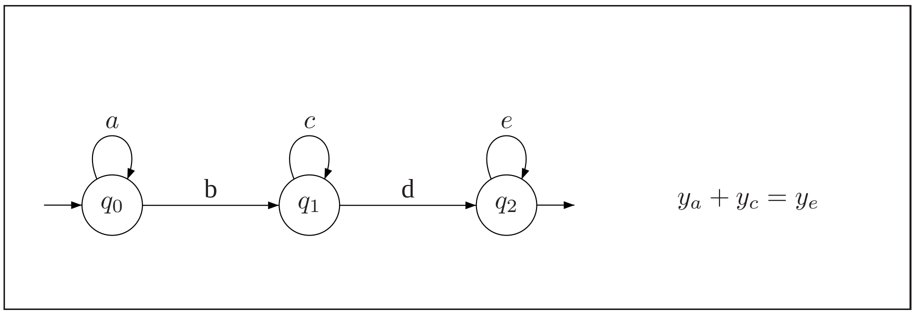


Figure 4. *A CQDD*

Let $\mathcal{A} = \langle \Sigma, S, S_0, E, l, A(y_1, \ldots, y_m), F\rangle$ be a CQDD with the letters from Σ linearly ordered: $a_1 < \ldots < a_k$. The extension EFOPCTL*(Pr)$[n]$ of the logic FOPCTL$^\star(\mathrm{PrA})[n]$ consists in considering formulae of the form $\mathcal{A}(\phi_1, \ldots, \phi_n)$ defined as follows:

– $\pi, i \models \mathcal{A}(\phi_1, \ldots, \phi_n)$ iff either $\epsilon \in \mathrm{L}(\mathcal{A})$,
or there is a finite word $a_{i_1}a_{i_2} \ldots a_{i_n} \in \mathrm{L}(\mathcal{A})$ such that for every $1 \leq j \leq n$, $\pi, i + (j-1) \models \phi_{i_j}$.

Thus, $\pi, i \models \mathcal{A}(\phi_1, \ldots, \phi_n)$ holds when a finite pattern induced from $\mathrm{L}(\mathcal{A})$ satisfies the respective arguments on the suffix path starting from position i. Note the correspondence between the letters $a_1, \ldots, a_k$ and the arguments $\phi_1, \ldots, \phi_n$. These automata-based operators are defined like those in (Wolper, 1983) except that the languages of finite words we consider are not exactly the regular languages. Of the regular languages only the bounded ones are allowed, and some context-free languages can be obviously defined.

For instance, in EFOPCTL*(Pr)$[n]$ we can state that there is a path and some $m \neq 0$ such that ϕ_1 holds true at the m first positions, then ϕ_2 holds true at the m next positions and then neither ϕ_1 nor ϕ_2 holds true forever. It is known that ETL is more expressive that LTL (Wolper, 1983) and this result could be lifted between FOPCTL$^\star(\mathrm{PrA})[n]$ and EFOPCTL*(Pr)$[n]$. However, at the present moment, we do not have a formal proof of this. Theorem 18 can be extended by allowing CQDD-based operators.

THEOREM 27. — *Given an ACS $\mathcal{C}$ of dimension n with Presburger transition system $S_{\mathcal{C}} = \langle S, \rightarrow \rangle$, for every EFOPCTL*(Pr)$[n]$ formula φ, one can compute a Presburger formula $A_\varphi(\mathbf{x})$ such that for every $\langle q, \mathbf{a} \rangle \in S_{\mathcal{C}}$, $\langle q, \mathbf{a} \rangle \models A_\varphi(\mathbf{x})$ iff $\mathcal{C}, \langle q, \mathbf{a} \rangle \models \varphi$.*

PROOF 28. — First, one can show that any language recognized by a CQDD $\mathcal{A}$ can be recognized by an ACS augmented with an alphabet. In such enriched ACS, the transitions between control states are of the form $q \xrightarrow{A(\mathbf{x},\mathbf{x}'),a} q'$ where a is a letter from a finite alphabet. Any transition $t = q \xrightarrow{a} q'$ in the CQDD is translated in the enriched ACS by a transition of the form $q \xrightarrow{x_t := x_t + 1, a} q'$ where x_t is a variable attached to the transition t. To any final state q of the CQDD, we associate the transition $q \xrightarrow{A(x_{t_1}, \ldots, x_{t_m}), \epsilon} q_{new}$ the CQDD having m distinct transitions and q_{new} being a new control state. There is a natural correspondence between the accepting runs of the CQDD and paths in the enriched ACS from initial states and q_{new}. In that way, the final constraint $A(y_1, \ldots, y_m)$ on the m transitions can be simulated in some ACS by increasing the ith counter whenever the ith transition is visited in the accepting run and checking the final constraint amounts to adding a final transition with identity function and domain precisely the values of the counters satisfying $A(y_1, \ldots, y_m)$. Hence, the proof technique from Theorem 16 can be used again. Typically, the following formulae can be defined in Presburger Arithmetic:

– By replacing valid control paths with accepting runs, there is a formula $PathConfig_{\mathcal{A}}(\mathbf{v}, i, j)$ stating that the ith transition of the accepting run $\mathbf{v}$ is a_j.
– The formula $Length(\mathbf{v}, l)$ states that the accepting run $\mathbf{v}$ has l transitions.
– The formula $AcceptingRun(\mathbf{v})$ states that $\mathbf{v}$ encodes an accepting run. Typically, we also use a path schema (starting from an initial state) and a cycle count vector which is possible because of the flat structure of $\mathcal{A}$.

Once these formulae are defined, it remains to define $T(\langle \mathbf{v}, \mathbf{x}, i \rangle; \mathcal{A}(\phi_1, \ldots, \phi_k))$ as follows (we omit the obvious case when $\varepsilon \in \mathrm{L}(\mathcal{A})$):

$$\exists\, \mathbf{v}', l,\ AcceptingRun(\mathbf{v}') \wedge Length(\mathbf{v}', l) \wedge$$

$$((l = 0) \vee \forall\, 1 \le i' \le l, \bigwedge_{c \in \{1,\ldots,k\}} PathConfig_{\mathcal{A}}(\mathbf{v}', i', c) \Rightarrow T(\langle \mathbf{v}, \mathbf{x}, i{+}i'{-}1\rangle; \phi_c)).$$

Then, the formula $A_{\varphi}(\mathbf{x})$ can be defined as in the proof of Theorem 18. ■

As a corollary, local model-checking problem for EFOPCTL*(Pr)$[n]$ over ACS is decidable.

COROLLARY 29. — *The following problems for EFOPCTL*(Pr)$[n]$ are decidable: local model checking, global model checking, validity checking with an initial configuration.*

8. Concluding Remarks

In this paper we have established decidability of various model-checking problems for FOPCTL$^{\star}(\mathrm{PrA})$ and related CTL*-like languages over Presburger arithmetic on a class of counter systems, by translation into Presburger arithmetic. Indeed, encoding quantification over paths can be performed by quantification over tuples of natural numbers. Hence, we have improved the decidability boundary for model-checking ACS with CTL*-like languages. The decidability of model-checking is currently open on extensions with fixed-point operators (e.g., Presburger μ-calculus) or monadic second-order quantification over ACS.

Another direction for further work is to analyze and extend further the class of ACS. For instance, giving up the functionality assumption on transitions that do not belong to a cycle preserves decidability, while it is open whether giving up the full functionality assumption still preserves decidability in the absence of first-order quantification. Similarly, the complexity of local model checking ACS with quantifier-free Presburger transition formulae over FOPCTL$^{\star}(\mathrm{PrA})$ is not fully characterized.

There are several related questions that have at least theoretical interest, which we have no addressed in the paper. For instance, how are the configuration graphs of ACSs placed relative to Caucal's hierarchy (Caucal, 2003)? It is not difficult to construct examples of ACS, like the one shown in Figure 5, with configuration graphs which are not pushdown graphs (Muller *et al.*, 1985). In the ACS displayed in Figure 5 the transitions from q_2 to q_3 and q_4 are only enabled when $x = 0$. It is easy to see that the configuration graph generated from the initial state $(0, 0, 0, 0)$ has infinitely many non-isomorphic ends, and therefore, by Muller-Schupp's theorem (Muller *et al.*, 1985) it is not a pushdown graph.

We currently do not know whether all ACS generate prefix-recognizable configuration graphs, or any graphs from higher levels of Caucal's hierarchy. Of course, decidability of MSO in a configuration graph does not imply decidability of FOPCTL$^{\star}(\mathrm{PrA})$, but it could suggest further strengthening of the results in the current paper.

Finally, the results in this paper can be extended to non-admissible counter systems, which are behaviorally equivalent in a suitable sense to ACSs. Typically, such

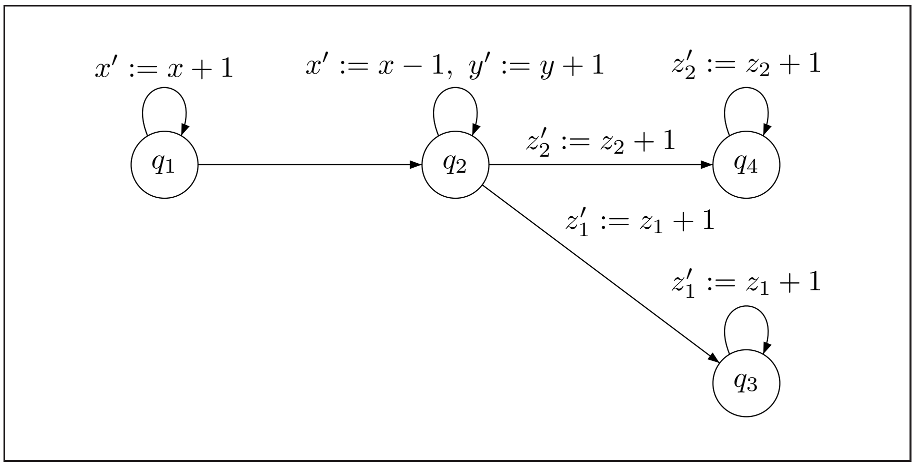


Figure 5. *A simple ACS with a non-pushdown graph and non-terminating paths*

equivalence can be achieved by '*flattening*' of the control graph. Extensions of the scope of model checking methods for FOPCTL$^\star(\mathrm{PrA})$ by means of flatability and other bisimulation equivalences to ACSs will be studied in a sequel paper.

9. References

Alur R., Henzinger T., "A really temporal logic", *Journal of the Association for Computing Machinery*, vol. 41, num. 1, pp. 181–204, 1994.

Annichini A., Bouajjani A., Sighireanu M., "TReX: a tool for reachablity analysis of complex systems", *CAV'01*, vol. 2102 of *Lecture Notes in Computer Science*, Springer, pp. 368–372, 2001.

Bardin S., Finkel A., Leroux J., "FASTer Acceleration of Counter Automata in Practice", *TACAS'04*, vol. 2988 of *Lecture Notes in Computer Science*, Springer, pp. 576–590, March, 2004.

Bardin S., Finkel A., Leroux J., Petrucci L., "FAST: Fast Acceleration of Symbolic Transition Systems", *CAV'03*, vol. 2725 of *Lecture Notes in Computer Science*, Springer, pp. 118–121, 2003.

Bardin S., Finkel A., Leroux J., Schnoebelen P., "Flat acceleration in symbolic model checking", *ATVA'05*, vol. 3707 of *Lecture Notes in Computer Science*, Springer, pp. 474–488, 2005.

Bardin S., Finkel A., Lozes E., Sangnier A., "From Pointer Systems to Counter Systems Using Shape Analysis", *AVIS'06*, 2006a.

Bardin S., Leroux J., Point G., "FAST Extended Release", *CAV'06*, vol. 4144 of *Lecture Notes in Computer Science*, Springer, pp. 63-66, 2006b.

Blumensath A., “Axiomatising Tree-Interpretable Structures”, *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, Springer-Verlag, pp. 596–607, 2002.

Blumensath A., Grädel E., “Finite Presentations of Infinite Structures: Automata and Interpretations”, *Theory of Computing Systems*, vol. 37, pp. 641 – 674, 2004.

Boigelot B., Symbolic methods for exploring infinite state spaces, PhD thesis, Université de Liège, 1998.

Boigelot B., “On iterating linear transformations over recognizable set of integers”, *Theoretical Computer Science*, vol. 309, num. 1–3, pp. 413–468, 2003.

Boigelot B., Wolper P., “Symbolic Verification with Periodic Sets”, *CAV’94*, vol. 818 of *Lecture Notes in Computer Science*, Springer, pp. 55–67, 1994.

Borosh I., Treybig L., “Bounds on positive integral solutions of linear diophantine equations”, , vol. 55, pp. 299–304, 1976.

Bouajjani A., Bozga M., Habermehl P., Iosif R., Moro P., Vojnar T., “Programs with lists are counter automata”, *CAV’06*, vol. 4144 of *Lecture Notes in Computer Science*, Springer, pp. 517–531, 2006.

Bouajjani A., Echahed R., Habermehl P., “On the verification problem of nonregular properties for nonregular processes”, *LICS’95*, pp. 123–133, 1995.

Bouajjani A., Esparza J., Maler O., “Reachability Analysis of Pushdown Automata: Application to Model Checking”, *CONCUR’97*, vol. 1243 of *LNCS*, Springer, pp. 135–150, 1997.

Bouajjani A., Habermehl P., “Symbolic Reachability Analysis of FIFO-channel systems with nonregular sets of configurations”, *Theoretical Computer Science*, vol. 221, num. 1–2, pp. 211–250, 1999.

Bozga M., Iosif R., Lakhnech Y., “Flat parametric counter automata”, *Fundamenta Informaticae*, vol. 91, num. 2, pp. 275–303, 2009.

Bruyère V., Dall’Olio E., Raskin J., “Durations, Parametric Model-Checking in Timed Automata with Presburger Arithmetic”, *STACS’03*, vol. 2607 of *Lecture Notes in Computer Science*, Springer, pp. 687–698, 2003.

Bultan T., Gerber R., Pugh W., “Symbolic model checking of infinite state systems using Presburger arithmetic”, *CAV’97*, vol. 1254 of *Lecture Notes in Computer Science*, Springer, pp. 400–411, 1997.

Burkart O., Caucal D., Moller F., Steffen B., “Verification of infinite structures.”, *Handbook of Process Algebra*, Elsevier, pp. 545–623, 2001.

Caucal D., “On infinite transition graphs having a decidable monadic theory”, *Theoretical Computer Science*, vol. 290, pp. 79–115, 2003.

Čerāns K., “Deciding Properties of Integral Relational Automata”, *ICALP*, vol. 820 of *Lecture Notes in Computer Science*, Springer, pp. 35–46, 1994.

Comon H., Cortier V., “Flatness is not a weakness”, *CSL’00*, vol. 1862 of *Lecture Notes in Computer Science*, Springer, pp. 262–276, 2000.

Comon H., Jurski Y., “Multiple counters automata, safety analysis and Presburger analysis”, *CAV’98*, vol. 1427 of *Lecture Notes in Computer Science*, Springer, pp. 268–279, 1998.

Cortier V., “About the Decision of Reachability for Register Machines”, *Theoretical Informatics and Applications*, vol. 36, num. 4, pp. 341–358, 2002.

Courcelle B., “Graph rewriting: An algebraic and logic approach”, *in* J. V. Leeuwen (ed.), *Handbook of Theoretical Computer Science, Volume B, Formal models and semantics*, Elsevier, pp. 193–242, 1990.

Dang Z., Pietro P. S., Kemmerer R., “Presburger Liveness Verification of Discrete Timed Automata”, *Theoretical Computer Science*, vol. 299, pp. 413–438, 2003.

Demri S., “LTL over integer periodicity constraints”, *Theoretical Computer Science*, vol. 360, num. 1–3, pp. 96–123, 2006.

Demri S., Finkel A., Goranko V., van Drimmelen G., “Towards a model-checker for counter systems”, *Proceedings of the 4th International Symposium on Automated Technology for Verification and Analysis (ATVA’06)*, vol. 4218 of *Lecture Notes in Computer Science*, Springer, pp. 493–507, 2006.

Demri S., Gastin P., *Modern Applications of Automata Theory*, IIsc Research Monographs, World Scientific, chapter Specification and Verification using Temporal Logics, 2009. To appear.

Emerson A., Halpern J., “‘Sometimes‘ and ’Not Never’ revisited: on branching versus Linear time temporal logic”, *Journal of the Association for Computing Machinery*, vol. 33, pp. 151–178, 1986.

Emerson A., Namjoshi K., “On Model Checking for Non-Deterministic Infinite-State Systems”, *LICS’98*, IEEE, pp. 70–80, 1998.

Esparza J., Finkel A., Mayr R., “On the verification of broadcast protocols”, *LICS’99*, pp. 352–359, 1999.

Finkel A., Leroux J., “How to compose Presburger accelerations: Applications to broadcast protocols”, *FST&TCS’02*, vol. 2256 of *Lecture Notes in Computer Science*, Springer, pp. 145–156, 2002.

Finkel A., Lozes E., Sangnier A., “Towards Model-Checking Programs with Lists”, *Infinity in Logic and Computation*, vol. 5489 of *Lecture Notes in Artificial Intelligence*, Springer, 2009. To appear.

Finkel A., Sutre G., “Decidability of reachability problems for classes of two counters automata”, *STACS’00*, vol. 2256 of *Lecture Notes in Computer Science*, Springer, pp. 346–357, 2000.

Finkel A., Willems B., Wolper P., “A Direct Symbolic Approach to Model Checking Pushdown Systems (Extended Abstract)”, *INFINITY’97*, vol. 9 of *ENTCS*, Elsevier Science, 1997.

Fribourg L., Olsén H., “Proving safety properties of infinite state systems by compilation into Presburger arithmetic”, *CONCUR’97*, vol. 1243 of *Lecture Notes in Computer Science*, Springer, pp. 213–227, 1997.

Ginsburg S., Spanier E., “Semigroups, Presburger formulas and languages”, *Pacific Journal of Mathematics*, vol. 16, num. 2, pp. 285–296, 1966.

Harel D., Kozen D., Tiuryn J., *Dynamic Logic*, MIT Press, 2000.

Ibarra O., “Reversal-bounded multicounter machines and their decision problems”, *Journal of the Association for Computing Machinery*, vol. 25, num. 1, pp. 116–133, 1978.

Ibarra O., Su J., Dang Z., Bultan T., Kemmerer A., “Counter Machines: Decidable Properties and Applications to Verification Problems”, *MFCS’00*, vol. 1893 of *Lecture Notes in Computer Science*, Springer, pp. 426–435, 2000.

Khoussainov B., Nerode A., “Automatic presentations of structures”, *Logic and Computation Complexity*, vol. 1995 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 367–392, 1995.

Lagarias J., “The $3x+1$ problem and its generalizations”, *The American Mathematical Monthly*, vol. 92, num. 1, pp. 3–23, 1985.

Laroussinie F., Schnoebelen P., “Specification in CTL + Past for Verification in CTL”, *Information and Computation*, vol. 156, pp. 236–263, 2000.

Leroux J., Algorithmique de la vérification des systèmes à compteurs. Approximation et accélération. Implémentation de l’outil FAST., PhD thesis, ENS de Cachan, France, 2003.

Leroux J., Regular acceleration for number decision diagrams, Technical Report num. 1385-06, LABRI, January, 2006.

Leroux J., Sutre G., “Flat counter systems are everywhere!”, *ATVA’05*, vol. 3707 of *Lecture Notes in Computer Science*, Springer, pp. 489–503, 2005.

Minsky M., *Computation, Finite and Infinite Machines*, Prentice Hall, 1967.

Muller D., Schupp P., “The theory of ends, pushdown automata , and second-order logic”, *Theoretical Computer Science*, vol. 37, pp. 51–75, 1985.

Papadimitriou C., “On the Complexity of Integer Programming”, *JACM*, vol. 28, num. 4, pp. 765–768, 1981.

Potapov I., “From Post Systems to the Reachability Problems for Matrix Semigroups and Multicounter Automata”, *DLT’04*, vol. 3340 of *Lecture Notes in Computer Science*, Springer, pp. 345–356, 2004.

Presburger M., “Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt”, *Comptes Rendus du premier congrès de mathématiciens des Pays Slaves, Warszawa*, pp. 92–101, 1929.

Schuele T., Schneider K., “Global vs. Local Model Checking: A Comparison of Verification Techniques for Infinite State Systems”, *SEFM’04*, IEEE, pp. 67–76, 2004.

Walukiewicz I., “Pushdown processes: games and model-checking”, *Information and Computation*, vol. 164, num. 2, pp. 234–263, 2001.

Wolper P., “Temporal logic can be more expressive”, *Information and Computation*, vol. 56, pp. 72–99, 1983.