Witness Runs for Counter Machines*

Clark Barrett[1], Stéphane Demri[1,2], and Morgan Deters[1]

[1] New York University, USA
[2] LSV, CNRS, France

Abstract. In this paper, we present recent results about the verification of counter machines by using decision procedures for Presburger arithmetic. We recall several known classes of counter machines for which the reachability sets are Presburger-definable as well as temporal logics with arithmetical constraints. We discuss issues related to flat counter machines, path schema enumeration, and the use of SMT solvers.

1 Introduction

Infinite-state systems. Model-checking is a standard approach to verifying properties of computing systems [CGP00] and it is known that dealing with infinity or unboundedness of computational structures leads easily to undecidable verification problems. Such problems include testing boundedness (checking whether a counter in a counter machine takes a finite amount of values) and those dealing with model-checking temporal formulae in which atomic formulae can state properties about unbounded values (e.g., arithmetical constraints about counter values). Roughly speaking, techniques for the verification of infinite-state systems stem from exact methods in which potentially infinite sets of configurations are finitely represented symbolically to semi-algorithms that are designed to behave well in practice. When exact methods can produce decision procedures, this is because an underlying finite structure can be identified in the verification problem. For instance, the set of reachable configurations can be effectively represented symbolically, typically by a formula in Presburger arithmetic, for which satisfiability is known to be decidable [Pre29]. The use of Presburger arithmetic for formal verification has been advocated in [SJ80]. Finiteness can also occur in a more subtle way, as in well-structured transition systems [FS01], for which termination is guaranteed thanks to underlying well-quasi-orderings, see e.g. [Kos82, OW05].

Counter machines. Counter machines are well-known infinite-state systems that have many applications in formal verification. Their ubiquity stems from their use as operational models for several purposes, including for instance for broadcast protocols [FL02], for programs with pointer variables (see [BBH+06]) and for logics for data words, see e.g. [BL10]. However, numerous model-checking

* Work partially supported by the EU Seventh Framework Programme under grant agreement No. PIOF-GA-2011-301166 (DATAVERIF).

P. Fontaine, C. Ringeissen, and R.A. Schmidt (Eds.): FroCos 2013, LNAI 8152, pp. 120–150, 2013.

problems for counter machines, such as reachability, are known to be undecidable. Many subclasses of counter machines admit a decidable reachability problem, such as reversal-bounded counter automata [Iba78] and flat counter automata [CJ98, Boi99, FL02]. These two classes of machines admit reachability sets effectively definable in Presburger arithmetic (assuming some additional conditions, unspecified herein). In general, computing the transitive closures of integer relations is a key step to solve verification problems on counter machines, see e.g. [BW94, CJ98, Fri00, FL02, BIK09].

Flatness. Flat counter machines are counter machines in which each control state belongs to at most one simple cycle (i.e., a cycle without any repetition of edges). Several classes of such flat operational devices have been identified and reachability sets have been shown effectively Presburger-definable for many of them, see e.g. [FO97, CJ98, Boi99, FL02, BIK09]. This provides a decision procedure for the reachability problem, given a prover for Presburger arithmetic validity. Effective semilinearity boils down to check that the effect of a loop can be characterized by a formula in Presburger arithmetic (or in any decidable fragment of first-order arithmetic). The results for flat counter machines can be then obtained by adequately composing formulae for loops and for finite paths. However, this approach, briefly described in this paper, suffers from at least two drastic limitations. First, flatness in counter machines remains a strong restriction on the control graph, though this has been relaxed by considering flattable counter machines, see e.g. [BFLS05, LS05, Ler13] and Section 3.5, where a machine may not itself be flat, but is known to have a flat unfolding with the same reachability set. The second limitation is due to the fact that reachability questions are not the only interesting ones and the verification of properties expressed in dedicated temporal logics is often desirable, see e.g. [DFGvD10].

In this paper, we present a selection of results about the verification of counter machines, at times assuming flatness, from reachability problems to model-checking problems with temporal logics. We follow an approach similar to [Fri00] to translate verification problems into Presburger arithmetic satisfiability. We focus on flattable counter machines and how to compute flat unfoldings by enumerating path schemas while invoking SMT solvers to optimize this enumeration. This part of the paper presents preliminary results, and it will be the subject of a dedicated paper.

Satisfiability Modulo Theories. Deciding Presburger arithmetic fragments is essential to verify properties of programs; see e.g. [Sho79] and [SJ80] for an early use of Presburger arithmetic for formal verification. Most well-known SMT solvers deal with quantifier-free linear integer formulae, also known as quantifier-free linear integer arithmetic ('QF_LIA' in the parlance of SMT-LIB [BST12]). For instance, this includes Z3 [dMB08], CVC4 [BCD$^+$11] and Alt-Ergo [Con12], to cite a few of them. However, dealing with quantifiers is usually a difficult task for SMT solvers that are better tailored to theory reasoning. Many general-purpose SMT solvers (including CVC3, CVC4, Z3, Yices, Alt-Ergo) do accept formulas with quantifiers and they handle them in roughly the same way, through

heuristic instantiation. Z3 is unique in that it implements several quantifier-elimination procedures as preprocessing steps, including a procedure for Presburger arithmetic ('LIA' in SMT-LIB). It is worth also mentioning automata-based tools dealing with satisfiability such as MONA [BKR96], LASH [BJW01] or TAPAS [LP09]. Even though Presburger arithmetic admits quantifier elimination, it is known that eliminating quantifiers can be computationally expensive (see e.g., [RL78, Grä88]). Recent developments propose a promising, lazy approach for quantifier elimination [Mon10].

2 Machines with Registers

In this section, we briefly present Presburger arithmetic (PA), the class of Presburger counter systems, and standard subclasses.

2.1 Presburger Arithmetic in a Nutshell

Presburger arithmetic (PA) has been introduced by M. Presburger in [Pre29] where it is shown decidable by quantifier elimination. This decidability result on the theory of addition is regarded today as a key result in theoretical computer science.

Let VAR $= \{\mathsf{x}, \mathsf{y}, \mathsf{z}, \ldots\}$ be a countably infinite set of *variables*. *Terms* are expressions of the form $a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + k$ where $a_1, \ldots, a_n$ are constant coefficients in $\mathbb{N}$, k is in $\mathbb{N}$ and the x_i's are variables. Variables and terms come with their interpretations when the variables are interpreted by natural numbers. A *valuation* $\mathfrak{v}$ is a map VAR $\to \mathbb{N}$ and it can be extended to the set of all terms as follows: $\mathfrak{v}(k) = k$, $\mathfrak{v}(a\mathsf{x}) = a \times \mathfrak{v}(\mathsf{x})$ and $\mathfrak{v}(\mathtt{t} + \mathtt{t}') = \mathfrak{v}(\mathtt{t}) + \mathfrak{v}(\mathtt{t}')$ for all terms $\mathtt{t}$ and $\mathtt{t}'$. *Formulae* are defined by the grammar below:

$$\phi ::= \ \mathtt{t} \leq \mathtt{t}' \mid \ \neg\phi \ \mid \ \phi \wedge \phi \ \mid \ \exists\, \mathsf{x}\ \phi$$

where $\mathtt{t}$ and $\mathtt{t}'$ are terms and $\mathsf{x} \in$ VAR. A formula ϕ is in the *linear fragment* $\stackrel{\text{def}}{\Leftrightarrow}$ ϕ is a Boolean combination of atomic formulae of the form $\mathtt{t} \leq \mathtt{t}'$. The semantics for formulae in (PA) is defined with the help of the satisfaction relation $\models$ that determines the conditions for the satisfaction of a formula under a given valuation (we omit the Boolean clauses):

- $\mathfrak{v} \models \mathtt{t} \leq \mathtt{t}' \ \stackrel{\text{def}}{\Leftrightarrow}\ \mathfrak{v}(\mathtt{t}) \leq \mathfrak{v}(\mathtt{t}')$,
- $\mathfrak{v} \models \exists\, \mathsf{x}\ \phi \ \stackrel{\text{def}}{\Leftrightarrow}$ there is $n \in \mathbb{N}$ such that $\mathfrak{v}[\mathsf{x} \mapsto n] \models \phi$ where $\mathfrak{v}[\mathsf{x} \mapsto n]$ is equal to $\mathfrak{v}$ except that x is mapped to n.

Any formula $\phi(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ whose free variables are among $\mathsf{x}_1, \ldots, \mathsf{x}_n$, with $n \geq 1$, defines a set of n-tuples $[\![\phi(\mathsf{x}_1, \ldots, \mathsf{x}_n)]\!] \stackrel{\text{def}}{=} \{\langle \mathfrak{v}(\mathsf{x}_1), \ldots, \mathfrak{v}(\mathsf{x}_n)\rangle \in \mathbb{N}^n : \mathfrak{v} \models \phi\}$ which contains all the tuples that make true the formula ϕ by ignoring the irrelevant interpretation of the bound variables and by fixing an arbitrary ordering between the variables. For instance, $[\![\mathsf{x}_1 < \mathsf{x}_2]\!] = \{\langle n, n'\rangle \in \mathbb{N}^2 : n < n'\}$. Let ϕ be a formula $\phi(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ with $n \geq 1$ free variables $\mathsf{x}_1, \ldots, \mathsf{x}_n$. We say that

$[\![\phi]\!]$ is a *Presburger set*. The *satisfiability problem* for (PA) is a decision problem that takes as input a formula ϕ and asks whether there is a valuation $\mathfrak{v}$ such that $\mathfrak{v} \models \phi$. If such a valuation exists, we say that ϕ is *satisfiable*.

Theorem 1 (Presburger Arithmetic Decidability). *[Pre29] The satisfiability problem for (PA) is decidable.*

The satisfiability problem can be solved in triple exponential time [Opp78] by analyzing the quantifier elimination procedure described in [Coo72]. It was shown 2ExpTime-hard in [FR74] and to be in 2ExpSpace in [FR79]. An exact complexity characterization is provided in [Ber80] (double exponential time on alternating Turing machines with linear amounts of alternations).

2.2 Presburger Counter Systems

The systems introduced below are finite-state automata augmented with registers, also known as counters (variables interpreted as natural numbers). Transitions are labelled by arithmetical constraints on counters defined in (PA). A *Presburger counter system* $\mathtt{C} = \langle Q, n, \delta \rangle$ is a structure (see e.g. [DFGvD10, Ler12]) such that

- Q is a nonempty finite set of *control states*,
- $n \geq 1$ is the *dimension* of the system, i.e. the number of counters, we assume that the counters are represented by the variables $\mathtt{x}_1, \ldots, \mathtt{x}_n$,
- δ is the *transition relation* defined as a finite set of triples of the form $\langle q, \phi, q' \rangle$, where q, q' are control states and ϕ is a Presburger formula whose free variables are among $\mathtt{x}_1, \ldots, \mathtt{x}_n, \mathtt{x}'_1, \ldots, \mathtt{x}'_n$. Prime variables are intended to be interpreted as the next values of the unprimed variables.

Figure 1 contains a Presburger counter system $\mathtt{C}$ such that inc(i) [resp. dec(i)] stands for the formula that increments [resp. decrements] the counter $\mathtt{x}_i$ and keeps unchanged the other counters. Formulae zero(i) tests if counter $\mathtt{x}_i$ is equal to zero but it has no effect on the counters.

Elements $t = \langle q, \phi, q' \rangle \in \delta$ are called *transitions* and are often represented by $q \xrightarrow{\phi} q'$. A *configuration* of the Presburger counter system $\mathtt{C} = \langle Q, n, \delta \rangle$ is a pair $\langle q, \boldsymbol{x} \rangle \in Q \times \mathbb{N}^n$. Given two configurations $\langle q, \boldsymbol{x} \rangle$, $\langle q', \boldsymbol{x'} \rangle$ and a transition $t = q \xrightarrow{\phi} q'$, we write $\langle q, \boldsymbol{x} \rangle \xrightarrow{t} \langle q', \boldsymbol{x'} \rangle$ whenever $\mathfrak{v} \models \phi$ (in (PA)) and for every $i \in [1, n]$, $\mathfrak{v}(\mathtt{x}_i) \stackrel{\text{def}}{=} \boldsymbol{x}(i)$ and $\mathfrak{v}(\mathtt{x}'_i) \stackrel{\text{def}}{=} \boldsymbol{x'}(i)$. Given a Presburger counter system $\mathtt{C}$, its *transition system* $\mathfrak{T}(\mathtt{C}) = \langle \mathfrak{S}, \rightarrow \rangle$ is a graph with $\mathfrak{S} = Q \times \mathbb{N}^n$ and $\rightarrow \subseteq \mathfrak{S} \times \mathfrak{S}$ such that $\langle \langle q, \boldsymbol{x} \rangle, \langle q', \boldsymbol{x'} \rangle \rangle \in \rightarrow \stackrel{\text{def}}{\Leftrightarrow}$ there exists a transition $t \in \delta$ such that $\langle q, \boldsymbol{x} \rangle \xrightarrow{t} \langle q', \boldsymbol{x'} \rangle$. As usual, $\xrightarrow{*}$ denotes the reflexive and transitive closure of the binary relation $\rightarrow$. The binary relation $\xrightarrow{*}$ is also called the *reachability relation* of $\mathtt{C}$ and it is sometimes written $\mathtt{Reach}_{\mathtt{C}}$. Similarly, we write $\mathtt{Reach}_{\mathtt{C}}(\langle q, \boldsymbol{x} \rangle)$ to denote the *reachability set* $\{\langle q', \boldsymbol{x'} \rangle \in \mathfrak{S} : \langle q, \boldsymbol{x} \rangle \xrightarrow{*} \langle q', \boldsymbol{x'} \rangle\}$. A *run* ρ is a nonempty (possibly infinite) sequence $\rho = \langle q_0, \boldsymbol{x_0} \rangle, \ldots, \langle q_k, \boldsymbol{x_k} \rangle, \ldots$ of configurations such that two consecutive configurations are in the relation $\rightarrow$ from $\mathfrak{T}(\mathtt{C})$.

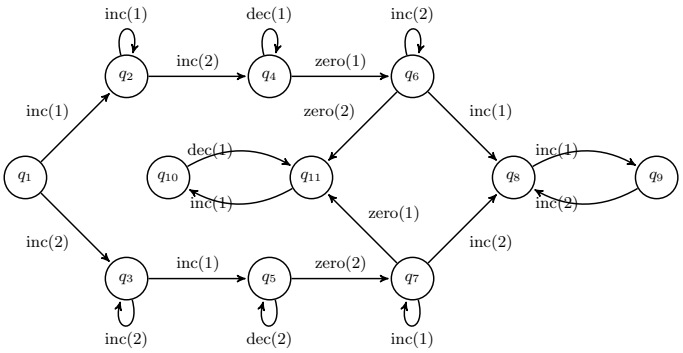

Fig. 1. A Presburger counter system

Most verification problems on Presburger counter systems are known to be undecidable since they include Minsky machines [Min67, Chapter 11] (see also [Min61, Section 3]) that are Turing-complete, even if restricted to two counters [Min67, Chapter 14]. The introduction of *program machines with registers* in [Min67], nowadays best known as Minsky machines, has been motivated by proposing an alternative to Turing machines that is closely related to programs.

2.3 Decision Problems

In this section, we recall several standard decision problems about Presburger counter systems. They are mainly related to reachability questions (problems related to temporal logics are introduced in Section 4).

Reachability problem:

Input: a Presburger counter system $\mathtt{C}$ and configurations $\langle q_0, \boldsymbol{x_0}\rangle$ and $\langle q_f, \boldsymbol{x_f}\rangle$.
Question: is there a finite run from $\langle q_0, \boldsymbol{x_0}\rangle$ to $\langle q_f, \boldsymbol{x_f}\rangle$?

Control state reachability problem:

Input: a Presburger counter system $\mathtt{C}$, a configuration $\langle q_0, \boldsymbol{x_0}\rangle$ and a state q_f.
Question: is there a finite run with initial configuration $\langle q_0, \boldsymbol{x_0}\rangle$ and whose final configuration has control state q_f?

Other verification problems on Presburger counter systems are worth mention, though not discussed herein, including the *control state repeated reachability problem* and the *termination problem*.

2.4 Some Classes of Presburger Counter Systems

In this section, we introduce classes of Presburger counter systems by restricting the general definition provided above. Additional requirements can be of

distinct nature: restriction on syntactic resources (number of counters, etc.), restriction on the control graph (e.g., flatness) and semantical restrictions (reversal-boundedness, etc.).

Counter systems. A *counter system* $\mathtt{C} = \langle Q, n, \delta \rangle$ is a Presburger counter system such that for each transition $t = q \xrightarrow{\phi} q' \in \delta$, ϕ can be written as $\phi_g \wedge \phi_u$, where ϕ_g (guard) is a Boolean combination of atomic formulae of the form $\mathtt{t} \leq \mathtt{t}'$ built over $\mathtt{x}_1, \ldots, \mathtt{x}_n$ and ϕ_u (update) is a formula of the form $\bigwedge_{i \in [1,n]} \mathtt{x}'_i = \mathtt{x}_i + \boldsymbol{b}(i)$ where $\boldsymbol{b} \in \mathbb{Z}^n$. The transition t is also written $q \xrightarrow{\langle \phi_g, \boldsymbol{b} \rangle} q'$. Minsky machines [Min67] are counter systems such that each update can change at most one counter and the guards are restricted to $\top$ and to zero-tests. The Presburger counter system in Figure 1 is indeed a counter system with the above meaning.

A *vector addition system with states* [KM69] (VASS for short) is a counter system such that all the transitions are of the form $q \xrightarrow{\langle \top, \boldsymbol{b} \rangle} q'$. So, a VASS can be represented by a tuple $\mathcal{V} = \langle Q, n, \delta \rangle$ where Q is the finite set of control states and δ is a finite subset of $Q \times \mathbb{Z}^n \times Q$. A famous result states that the reachability problem for VASS is decidable [May84, Kos82, Ler09]. It has been the subject of the book [Reu90] and its proof requires many non-trivial steps involving graph theory, logic and theory of well-quasi-orderings. Nevertheless, the exact complexity of the reachability problem is open: we know it is ExpSpace-hard [Lip76] and no primitive recursive upper bound exists. In [Ler09], existence of semilinear separators in case of non-reachability in VASS leads to promising developments.

A Note to the Reader. Counter systems are the main class of Presburger counter systems considered in this document. However, we are aware that the current term might be confusing: when we really want to mean the full class of systems, we will use the more general term 'Presburger counter system.'

Reversal-bounded counter systems. In this section, we consider counter systems for which the atomic formulae in guards are of the form $\mathtt{t} \leq k$ or $\mathtt{t} \geq k$ with $k \in \mathbb{Z}$ and $\mathtt{t}$ is of the form $\sum_i a_i \mathtt{x}_i$ with the a_i's in $\mathbb{Z}$. There is no real restriction with the class introduced earlier except that we require that atomic formulae occur in a certain way.

A *reversal* for a counter occurs in the run of some counter system when there is an alternation from nonincreasing mode to nondecreasing mode and vice-versa. Below, we propose a slight generalization from [BD11] that captures the notion of reversal-boundedness from [Iba78]; reversal-boundedness applies to counters but also to terms occurring in guards. Let $\mathtt{C} = \langle Q, n, \delta \rangle$ be a counter system and $\mathtt{T}$ be a finite set of terms including $\{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$. From a run $\rho = \langle q_0, \boldsymbol{x_0} \rangle, \langle q_1, \boldsymbol{x_1} \rangle, \ldots$ of $\mathtt{C}$, in order to describe the behavior of counters and terms varying along ρ, we define a sequence of *mode vectors* $\mathfrak{md}_0, \mathfrak{md}_1, \ldots$ (of the same length as ρ) such that each $\mathfrak{md}_i$ has profile $\mathtt{T} \to \{\nearrow, \searrow\}$. Intuitively, each value in a mode vector records whether a term is currently in an increasing phase or in an decreasing phase (this includes the counters themselves as in standard reversal-boundedness). Given $\mathtt{t} = \sum_i a_i \mathtt{x}_i$ and a counter vector $\boldsymbol{x}$, we write

$\boldsymbol{x}(\mathtt{t})$ to denote the integer $\sum a_i \boldsymbol{x}(i)$. We are now ready to define the sequence $\mathtt{md}_0, \mathtt{md}_1, \ldots$ By convention, $\mathtt{md}_0$ is the constant map $\nearrow$. For every $j \geq 0$ and $\mathtt{t} \in \mathtt{T}$, we have $\mathtt{md}_{j+1}(\mathtt{t}) \stackrel{\text{def}}{=} \mathtt{md}_j(\mathtt{t})$ when $\boldsymbol{x}_j(\mathtt{t}) = \boldsymbol{x}_{j+1}(\mathtt{t})$, $\mathtt{md}_{j+1}(\mathtt{t}) \stackrel{\text{def}}{=} \nearrow$ when $\boldsymbol{x}_{j+1}(\mathtt{t}) - \boldsymbol{x}_j(\mathtt{t}) > 0$ and $\mathtt{md}_{j+1}(\mathtt{t}) \stackrel{\text{def}}{=} \searrow$ when $\boldsymbol{x}_{j+1}(\mathtt{t}) - \boldsymbol{x}_j(\mathtt{t}) < 0$. Let $Rev_{\mathtt{t}} = \{j \in [0, \mathrm{len}(\rho)-1] : \mathtt{md}_j(\mathtt{t}) \neq \mathtt{md}_{j+1}(\mathtt{t})\}$; we say that ρ is r-$\mathtt{T}$-*reversal-bounded* for some $r \geq 0 \stackrel{\text{def}}{\Leftrightarrow}$ for all $\mathtt{t} \in \mathtt{T}$, $\mathrm{card}(Rev_{\mathtt{t}}) \leq r$. Given a counter system $\mathtt{C}$, we write $\mathtt{T}_{\mathtt{C}}$ to denote the set of terms $\mathtt{t}$ occurring in atomic formulae of the form $\mathtt{t} \sim k$ with $\sim \in \{\leq, \geq\}$ augmented with the counters in $\{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$. An initialized counter system $\langle \mathtt{C}, \langle q, \boldsymbol{x} \rangle \rangle$ is *reversal-bounded* $\stackrel{\text{def}}{\Leftrightarrow}$ there is $r \geq 0$ such that every run from $\langle q, \boldsymbol{x} \rangle$ is r-$\mathtt{T}_{\mathtt{C}}$-reversal-bounded. When $\mathtt{T}$ is reduced to $\{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$, $\mathtt{T}$-reversal-boundedness is equivalent to reversal-boundedness from [Iba78]. Note that the counter system in Figure 1 is $\{\mathtt{x}_1, \mathtt{x}_2\}$-reversal-bounded from any initial configuration of the form $\langle q_1, \boldsymbol{x_0} \rangle$.

Compared to the subclasses considered so far, reversal-bounded counter systems are augmented with an initial configuration so that existence of the bound r is relative to the initial configuration. Secondly, this class is not defined from the class of counter systems by imposing syntactic restrictions but rather semantically. The main property related to reversal-bounded counter systems is the result below.

Theorem 2. *[Iba78, BD11] Given a counter system* $\mathtt{C}$*,* $r \geq 0$ *and control states* q, q'*, one can effectively compute a Presburger formula* $\phi_{q,q'}(\mathtt{x}_1, \ldots, \mathtt{x}_n, \mathtt{y}_1, \ldots, \mathtt{y}_n)$ *such that for all valuations* $\mathfrak{v}$*, we have* $\mathfrak{v} \models \phi$ *iff there is an* r-$\mathtt{T}_{\mathtt{C}}$*-reversal-bounded run from* $\langle q, \langle \mathfrak{v}(\mathtt{x}_1), \ldots, \mathfrak{v}(\mathtt{x}_n) \rangle \rangle$ *to* $\langle q', \langle \mathfrak{v}(\mathtt{y}_1), \ldots, \mathfrak{v}(\mathtt{y}_n) \rangle \rangle$*.*

So, bounding the number of reversals in runs allows to characterize the reachability sets by computing Presburger formulae. This approach can be generalized to richer models, see e.g., [HR87, FS08, HL11].

Affine Presburger counter systems. Now, we present the class of *affine Presburger counter systems* that substantially extends the class of counter systems by allowing any guard that can be defined in (PA) and by giving the possibility to have affine updates. A partial function f from $\mathbb{N}^n$ to $\mathbb{N}^n$ is *affine* $\stackrel{\text{def}}{\Leftrightarrow}$ there exist a matrix $\mathtt{A} \in \mathbb{Z}^{n \times n}$ and $\boldsymbol{b} \in \mathbb{Z}^n$ such that for every $\boldsymbol{a} \in \mathrm{dom}(f)$, we have $f(\boldsymbol{a}) = \mathtt{A}\boldsymbol{a} + \boldsymbol{b}$. f is *Presburger-definable* $\stackrel{\text{def}}{\Leftrightarrow}$ the graph of f is a Presburger set (binary relation).

A Presburger counter system $\mathtt{C} = \langle Q, n, \delta \rangle$ is *affine* when for every transition $q \xrightarrow{\phi} q' \in \delta$, $[\![\phi]\!]$ is affine and there is a triple $\langle \phi_g, \mathtt{A}, \boldsymbol{b} \rangle$ such that ϕ_g (guard) is a formula in (PA) with free variables among $\mathtt{x}_1, \ldots, \mathtt{x}_n$ and $[\![\phi]\!] = \{\langle \boldsymbol{x}, \boldsymbol{x'} \rangle \in \mathbb{N}^{2n} : \boldsymbol{x'} = \mathtt{A}\boldsymbol{x} + \boldsymbol{b}$ and $\boldsymbol{x} \in [\![\phi_g]\!]\}$. The formula ϕ_g represents the guard of the transition and the pair $\langle \mathtt{A}, \boldsymbol{b} \rangle$ is the deterministic update function. Such a triple $\langle \phi_g, \mathtt{A}, \boldsymbol{b} \rangle$ is called an *affine update* and we also write $[\![\langle \phi_g, \mathtt{A}, \boldsymbol{b} \rangle]\!]$ to denote $[\![\phi]\!]$. Observe that one can decide whether a Presburger formula ϕ satisfies that $[\![\phi]\!]$ is affine [DFGvD10, Proposition 3]. Furthermore, counter systems are affine counter systems in which the only matrix is identity. This class of Presburger counter systems has been introduced in [FL02].

Observe that given $t = q \xrightarrow{\langle \phi_g, \mathtt{A}, \boldsymbol{b} \rangle} q'$, there is a Presburger formula $\varphi(\overline{\mathsf{x}}, \overline{\mathsf{x}'})$ such that for every $\mathfrak{v}$, we have $\mathfrak{v} \models \varphi$ iff $\langle q, \langle \mathfrak{v}(\mathsf{x}_1), \ldots, \mathfrak{v}(\mathsf{x}_n) \rangle\rangle \xrightarrow{t} \langle q', \langle \mathfrak{v}(\mathsf{x}'_1), \ldots, \mathfrak{v}(\mathsf{x}'_n) \rangle\rangle$. Here is the witness formula that encodes the one-step relation:

$$\phi_g(\overline{\mathsf{x}}) \wedge \bigwedge_{i \in [1,n]} (\mathsf{x}'_i = \sum_j \mathtt{A}(i,j)\mathsf{x}_j + \boldsymbol{b}(i))$$

Note that the composition of affine updates is still an affine update.

Presburger counter systems with octagonal constraints. A *Presburger counter systems with octagonal constraints* is such that for each transition $q \xrightarrow{\phi} q' \in \delta$, the formula ϕ is a conjunction of atomic formulae of the form $\pm\mathsf{y} \pm \mathsf{z} \leq k$ where y, z are variables among $\mathsf{x}_1, \ldots, \mathsf{x}_n, \mathsf{x}'_1, \ldots, \mathsf{x}'_n$, $k \in \mathbb{Z}$ and $\pm\mathsf{y}$ stands for either y or $-\mathsf{y}$ (same applies for $\pm\mathsf{z}$). Constraints of the form $\pm\mathsf{y} \pm \mathsf{z} \leq k$ are called *octagons* and have been considered in [BGI09]. Note that octagons include constraints of the form $\mathsf{y} \leq \mathsf{z} + k$ or $\mathsf{y} \leq k$ considered in [CJ98]. Unlike the counter systems, in Presburger counter systems with octagonal constraints the transitions do not necessarily lead to functional updates. Here is an example of formula labelling a transition: $\phi = (\mathsf{x}_1 + 1 < \mathsf{x}'_1) \wedge (\mathsf{x}_2 - 3 = \mathsf{x}'_2)$. In [CJ98], Presburger counter systems with octagonal constraints with only constraints of the form $\mathsf{y} \leq \mathsf{z} + k$ or $\mathsf{y} \leq k$ have been studied and a major result established in [CJ98] states that the effect of any loop can be effectively defined in (PA).

Imperfect counter automata. Counter automata are defined as VASS except that we accept also zero-tests on counters as guards. Below, we briefly consider variants of counter automata in which counter values can be decremented without notification (a loss) or counter values can be incremented without notification (a gain) – but not the two possibilities in the same model. A *lossy counter automaton* is a counter automaton such that for all $q \in Q$ and for all $i \in [1, n]$, $q \xrightarrow{\mathrm{dec}(i)} q$ (which allows us to simulate losses). The control state reachability problem for lossy counter automata is decidable and actually lossy counter automata form a subclass of lossy channel systems, see e.g. [Sch02] and the reachability problem for lossy channel systems is decidable [AJ96, FS01]. For instance, they can be used to model lossy channel systems for which the ordering of the messages is not relevant. In that case, each counter can store how many messages of a given type are present in the channel. Lossy counter automata have been introduced in [May03]. Similarly, a *gainy counter automaton* is a counter automaton such that for all $q \in Q$ and for all $i \in [1, n]$, $q \xrightarrow{\mathrm{inc}(i)} q \in \delta$ (which allows us to simulate gains). The control state reachability problem for gainy counter automata can be shown decidable by making a correspondence with reset VASS (VASS in which it is possible to reset counter values) but the problem is nonprimitive recursive, see e.g. [Sch02, Sch10]. Even though Presburger counter systems with imperfect computations are not further discussed in the paper, they form an interesting class of systems related to many other verification problems.

In order to conclude this section, it is worth noting that there exist plenty of other classes of Presburger counter systems for which reachability problems can be solved by using (PA) (see e.g., subclasses of Petri nets). However, since Presburger counter systems are Turing-complete, designing new (tractable) subclasses remains an ongoing process. In the next sections, we focus on presenting proof techniques to solve reachability problems for some of the classes.

3 Loops, Path Schemas and Flatness

3.1 Computing Loop Effects in (PA)

Let $\mathtt{C} = \langle Q, n, \delta \rangle$ be a Presburger counter system. A *path* $\mathtt{p}$ of $\mathtt{C}$ is a finite sequence of transitions from δ corresponding to a path in its control graph. We write *first*($\mathtt{p}$) [resp. *last*($\mathtt{p}$)] to denote the first [resp. last] control state of a path. A *loop* $\mathtt{l}$ is a non-empty path $\mathtt{p}$ such that *first*($\mathtt{p}$) = *last*($\mathtt{p}$) and we write $\mathtt{effect}(\mathtt{l})$ to denote the *effect* of the loop $\mathtt{l}$ defined as below:

$$\{\langle \boldsymbol{x}, \boldsymbol{x'} \rangle \in \mathbb{N}^n \times \mathbb{N}^n : \langle \mathit{first}(\mathtt{l}), \boldsymbol{x} \rangle \xrightarrow{\mathtt{l}} \langle \mathit{last}(\mathtt{l}), \boldsymbol{x'} \rangle\}$$

Similarly, we write $\mathtt{effect}^{<\omega}(\mathtt{l})$ to denote the *repeated effect* of the loop $\mathtt{l}$:

$$\{\langle \boldsymbol{x}, \boldsymbol{x'} \rangle \in \mathbb{N}^n \times \mathbb{N}^n : \langle \mathit{first}(\mathtt{l}), \boldsymbol{x} \rangle \xrightarrow{\mathtt{l}^i} \langle \mathit{last}(\mathtt{l}), \boldsymbol{x'} \rangle,\ i \geq 0\}$$

The reachability problem for loops can be then defined as follows: given a loop $\mathtt{l}$ from a Presburger counter system $\mathtt{C}$ of dimension n and two counter value vectors $\boldsymbol{x_0}$, $\boldsymbol{x_f}$ in $\mathbb{N}^n$, is $\langle \boldsymbol{x_0}, \boldsymbol{x_f} \rangle \in \mathtt{effect}^{<\omega}(\mathtt{l})$? Repeated effect is simply called *acceleration* in [FL02, Section 3].

Note that even though $\mathtt{effect}(\mathtt{l})$ can be defined by a Presburger formula, this does not imply that it is the case for $\mathtt{effect}^{<\omega}(\mathtt{l})$ too. Indeed, if the binary relation R is Presburger set, then this does not imply that its reflexive and transitive closure R^* is a Presburger set too. For instance, if $R = \{\langle \alpha, 2\alpha \rangle \in \mathbb{N}^2 : \alpha \in \mathbb{N}\}$ then $R^* = \{\langle \alpha, 2^\beta \alpha \rangle \in \mathbb{N}^2 : \alpha, \beta \in \mathbb{N}\}$ is not Presburger-definable. By contrast, if $S = \{\langle \alpha, \alpha + 1 \rangle \in \mathbb{N}^2 : \alpha \in \mathbb{N}\}$ then $S^* = \{\langle \alpha, \beta \rangle \in \mathbb{N}^2 : \alpha < \beta,\ \alpha, \beta \in \mathbb{N}\}$ is a Presburger set. The question of deciding whether the reflexive and transitive closure of a Presburger-definable binary relation is Presburger-definable is known to be intimately related to the fact that reachability relations from Presburger counter systems are Presburger-definable, which leads to decidability when effectiveness is guaranteed too. Indeed, consider the following loop with $q_1 = q_k$:

$$q_1 \xrightarrow{\phi_1(\mathsf{x}_1, \ldots, \mathsf{x}'_n)} q_2 \xrightarrow{\phi_2(\mathsf{x}_1, \ldots, \mathsf{x}'_n)} \cdots \xrightarrow{\phi_{k-1}(\mathsf{x}_1, \ldots, \mathsf{x}'_n)} q_{k-1} \xrightarrow{\phi_k(\mathsf{x}_1, \ldots, \mathsf{x}'_n)} q_k.$$

The effect of the loop can be represented by the Presburger formula below:

$$\psi(\bar{\mathsf{x}}, \bar{\mathsf{x}'}) \stackrel{\text{def}}{=} \exists\ \bar{\mathsf{y}_1}, \ldots, \bar{\mathsf{y}_k}\ \ \phi_1(\bar{\mathsf{x}}, \bar{\mathsf{y}_1}) \wedge \phi_2(\bar{\mathsf{y}_1}, \bar{\mathsf{y}_2}) \wedge \cdots \wedge \phi_k(\bar{\mathsf{y}_k}, \bar{\mathsf{x}'})$$

where $\bar{\mathsf{x}}$, $\bar{\mathsf{x}'}$, $\bar{\mathsf{y}_1}$, ..., $\bar{\mathsf{y}_k}$ are sequences of variables of length n.

In order to decide the reachability problem on the loop, it is essential to represent symbolically $\texttt{effect}^{<\omega}(\texttt{l})$. The best we can hope for is that $\texttt{effect}^{<\omega}(\texttt{l})$ is a Presburger set. This motivates the definition below.

Definition 3. *Given $R \subseteq \mathbb{N}^{2n}$, we define the* counting iteration of R *as the relation $R_{\mathbf{CI}} \subseteq \mathbb{N}^n \times \mathbb{N} \times \mathbb{N}^n$ such that $\langle \boldsymbol{x}, i, \boldsymbol{y} \rangle \in R_{\mathbf{CI}} \stackrel{def}{\Leftrightarrow} \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in R^i$ (i compositions of R). R has a* Presburger counting iteration *if $R_{\mathbf{CI}}$ is a Presburger set.*

If R has a Presburger counting iteration, then there exists a Presburger formula $\varphi(\bar{\mathsf{x}}, \mathsf{z}, \bar{\mathsf{y}})$ such that $[\![\varphi]\!] = R_{\mathbf{CI}}$. Consequently, the relation R^* is Presburger-definable since $[\![\exists\ \mathsf{z}\ \varphi]\!] = R^*$. Observe that $\{\langle \alpha, \alpha + 1 \rangle \in \mathbb{N}^2 : \alpha \in \mathbb{N}\}$ has a Presburger counter iteration witnessed by a Presburger formula of the form $\mathsf{x}' = \mathsf{x} + \mathsf{y}$.

Definition 4 (The property ($\star$) of Presburger counter systems). *A class of Presburger counter systems is said to satisfy the property ($\star$) when, for every loop* $\texttt{l}$, $\texttt{effect}(\texttt{l})$ *has the Presburger counting iteration and its Presburger formula is computable.*

Note in particular that this means that for every loop $\texttt{l}$, the set $\{\langle \boldsymbol{x}, i, \boldsymbol{x}' \rangle : \langle \mathit{first}(\texttt{l}), \boldsymbol{x} \rangle \xrightarrow{\texttt{l}^i} \langle \mathit{last}(\texttt{l}), \boldsymbol{x}' \rangle, i \geq 0\}$ is effectively definable by a Presburger formula $\varphi^\star_{\texttt{l}}$ (with $2n + 1$ free variables).

3.2 Finitary Path Schemas

A *path schema* $\texttt{P}$ is a regular expression built over the alphabet of transitions such that its language represents an overapproximation of the set of labels obtained from finite runs following the transitions of $\texttt{P}$ (counter values are ignored). This notion has been extensively used since [FO97, Fri00, FL02] and this provides a natural transition since path schemas are made of loops and paths. More precisely, a *finitary path schema* $\texttt{P}$ is of the form $\texttt{p}_1\texttt{l}_2^*\texttt{p}_3\texttt{l}_4^* \ldots \texttt{l}_{k-1}^*\texttt{p}_k$ where (1) $\texttt{l}_2, \ldots, \texttt{l}_{k-1}$ are loops and (2) $\texttt{p}_1\texttt{l}_2\texttt{p}_3\texttt{l}_4 \ldots \texttt{p}_k$ is a path. The *length* of a path schema, written $\mathrm{len}(\texttt{P})$, is defined as the number of letter occurrences in the regular expression defining the path schema (no substructure sharing). Let $\mathrm{Lan}(\texttt{P})$ denote the set of finite words in δ^* which belong to the language defined by $\texttt{P}$. Note that some elements of $\mathrm{Lan}(\texttt{P})$ may not correspond to any actual run because of constraints on counter values. Finally, we say that a run ρ starting in a configuration $\langle q_0, \boldsymbol{x_0} \rangle$ *respects* a path schema $\texttt{P}$ if the sequence of transitions generating ρ belongs to $\mathrm{Lan}(\texttt{P})$.

Path schemas are used as a means to encode the structure of a potentially infinite set of runs. That is why, we will pay a special attention to avoid considering distinct path schemas $\texttt{P}$ and $\texttt{P}'$ such that $\mathrm{Lan}(\texttt{P}) \subseteq \mathrm{Lan}(\texttt{P}')$. Containment problem for regular expressions is PSPACE-complete but co-NP-complete for regular expressions defining bounded languages, see e.g. [HRS76]. Any set $\mathrm{Lan}(\texttt{P})$ defines a bounded language.

That is why, in the following we only consider path schemas such that the loops $\mathtt{l}$ are not multiples of smaller loops (i.e., $\mathtt{l} = (\mathtt{l}')^i$ with $i \geq 2$) and no path $\mathtt{p}$ contains a loop as a factor (which bounds the length of such paths). Such loops are called *simple loops*. In the following, such path schemas are called *good*. It is easy to see that every finite run respects a good path schema.

Lemma 5. *When ($\star$) holds, $\{\langle \boldsymbol{x}, \boldsymbol{x}' \rangle : \langle q, \boldsymbol{x} \rangle \xrightarrow{*} \langle q', \boldsymbol{x}' \rangle$ respects $\mathtt{P}\}$ is effectively definable by a Presburger formula $\varphi_{\mathtt{P}}$ (with $2n$ free variables).*

Effective semilinearity boils down to check that the effect of a loop can be characterized by a formula in Presburger arithmetic (or in any decidable fragment of first-order arithmetic). The above result can be then obtained by adequately composing formulae for the loops and for the finite paths.

By way of example, note that the effect of the self-loop $q \xrightarrow{\mathsf{x}'=2\mathsf{x}} q$ is not definable in Presburger arithmetic since $\{2^i : i \geq 0\}$ is not Presburger-definable. By contrast, the effect of the self-loop

$$q \xrightarrow{\mathsf{x}'_1=\mathsf{x}_1+2 \wedge \mathsf{x}'_2=\mathsf{x}_2+3 \wedge \phi(\mathsf{x}_1,\mathsf{x}_2)} q$$

for any Presburger formula $\phi(\mathsf{x}_1, \mathsf{x}_2)$ is Presburger-definable since $\{\langle \boldsymbol{x}, i, \boldsymbol{x}' \rangle : \langle q, \boldsymbol{x} \rangle \xrightarrow{\mathtt{l}^i} \langle q, \boldsymbol{x}' \rangle\}$ can be defined with the formula below:

$$\varphi(\mathsf{x}_1, \mathsf{x}_2, i, \mathsf{x}'_1, \mathsf{x}'_2) \stackrel{\text{def}}{=} \mathsf{x}'_1 = \mathsf{x}_1 + 2i \wedge \mathsf{x}'_2 = \mathsf{x}_2 + 3i \wedge \forall\, \mathsf{y}\ (0 \leq \mathsf{y} < i) \Rightarrow \phi(\mathsf{x}_1 + 2\mathsf{y}, \mathsf{x}_2 + 3\mathsf{y})$$

Here are concrete classes to apply Lemma 5.

Theorem 6

(I) *Presburger counter systems with octagonal constraints enjoy ($\star$) [BGI09] (see also [CJ98] for a substantial result on a subclass).*
(II) *Counter systems enjoy ($\star$) (folklore result, see e.g. [Fri00, DDS12]).*

An implementation of the transitive closure of octagonal relations is done in the tool FLATA, see e.g., [BGI09].

3.3 Flat Presburger Counter Systems

A Presburger counter system $\mathtt{C}$ is *flat* $\stackrel{\text{def}}{\Leftrightarrow}$ every control state belongs to at most one simple cycle (i.e., a loop in which each transition occurs at most once). As far as we can judge, the term 'flat' in that sense has been introduced in [FO97, CJ98, Fri00]. The Presburger counter system in Figure 1 is flat.

Lemma 7. *Every flat Presburger counter system has a finite number of good path schemas that is at most exponential in its size.*

Of course, this is not the only way to get a finite amount of path schemas, for instance when from an initial configuration, termination is guaranteed but here the finite number of path schemas is structurally guaranteed.

Theorem 8. *Let $\mathcal{C}$ be a class of Presburger counters that enjoys ($\star$). Then, for every flat Presburger counter system from $\mathcal{C}$, the reachability relation $\mathtt{Reach}_{\mathtt{C}}$ is Presburger-definable.*

This is at the heart of the decidability results for verifying safety and reachability properties on flat Presburger counter systems from [CJ98, FL02, BIK09] whereas for the verification of temporal properties, it is much more difficult to get sharp complexity characterization, see e.g. [DDS12].

Corollary 9. *Let $\mathcal{C}$ be a class of Presburger counters that enjoys ($\star$). The reachability problem for $\mathcal{C}$ is decidable.*

The corollary can be obtained as follows. Consider the instance $\mathtt{C}$, $\langle q_0, \boldsymbol{x_0}\rangle$ and $\langle q_f, \boldsymbol{x_f}\rangle$. We have seen that we can compute the Presburger formula ϕ that encodes the reachability relation in $\mathtt{C}$. It remains to check satisfiability of the formula $(\bigwedge_{i=1}^{i=n}(\mathtt{x}_i = \boldsymbol{x_0}(i) \wedge \mathtt{x}'_i = \boldsymbol{x_f}(i))) \wedge \phi$ assuming free variables in ϕ are $\mathtt{x}_1, \ldots, \mathtt{x}_n, \mathtt{x}'_1, \ldots, \mathtt{x}'_n$. This can be done thanks to Theorem 1.

3.4 Finite Monoid Property in Affine Presburger Counter Systems

Below, we present a class of affine Presburger counter systems with Presburger-definable loop effects even though the class does not necessarily enjoy the property ($\star$). Given $\mathtt{A} \in \mathbb{Z}^{n\times n}$, let $\mathtt{A}^*$ be the monoid generated from $\mathtt{A}$ with $\mathtt{A}^* = \{\mathtt{A}^i : i \in \mathbb{N}\}$. The identity element is naturally the identity matrix $\mathtt{A}^0 = I$. Given a matrix $\mathtt{A} \in \mathbb{Z}^{n\times n}$, checking whether the monoid generated by $\mathtt{A}$ is finite, is decidable [MS77]. By way of example, with $\mathtt{A} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$, we have

$$\mathtt{A}^2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \quad \mathtt{A}^3 = \begin{pmatrix} 1 & 0 \\ 3 & 1 \end{pmatrix} \quad \ldots \mathtt{A}^m = \begin{pmatrix} 1 & 0 \\ m & 1 \end{pmatrix}$$

So $\mathtt{A}^*$ is not finite. Finiteness of the monoid generated from $\mathtt{A}$ is interesting because of the lemma below.

Lemma 10. *[BW98, Boi99, FL02] Let $R \subseteq \mathbb{N}^n \times \mathbb{N}^n$ be a binary relation of dimension n defined by the triple $\langle \phi_g, \mathtt{A}, \boldsymbol{b}\rangle$ such that $R = \{\langle \boldsymbol{x}, \boldsymbol{x'}\rangle \in \mathbb{N}^{2n} : \boldsymbol{x'} = \mathtt{A}\boldsymbol{x} + \boldsymbol{b}$ and $\boldsymbol{x} \in [\![\phi_g]\!]\}$. If $\mathtt{A}^*$ is finite, then R has a Presburger counting iteration.*

It is worth adding that one can also effectively compute the Presburger formula encoding the relation R^*. A recent work unifying [CJ98, FL02, BGI09, BIL09] by considering all the families of formulae labelling transitions from these works can be found in [BIK09].

A loop in an affine counter system has the *finite monoid property* $\stackrel{\text{def}}{\Leftrightarrow}$ its corresponding affine update $\langle \phi_g, \mathtt{A}, \boldsymbol{b}\rangle$, possibly obtained by composition of several affine updates, satisfies that $\mathtt{A}^*$ is finite. Let us introduce below the class of admissible counter systems.

Definition 11. *An affine Presburger counter system $\mathtt{C}$ is* admissible *iff*

1. *there is at most one transition between two control states (always possible as soon as disjunction is allowed in guards),*
2. *its control graph is flat,*
3. *each simple loop has the finite monoid property.*

The restriction to admissible counter systems mainly takes advantage of Lemma 10 as shown below.

Theorem 12. *[FL02] Let* $\mathtt{C}$ *be an admissible Presburger counter system and* $q, q' \in Q$*. One can effectively compute a Presburger formula* ϕ *such that for every valuation* $\mathfrak{v}$*, we have* $\mathfrak{v} \models \phi$ *iff* $\langle q, \langle \mathfrak{v}(\mathtt{x}_1), \ldots, \mathfrak{v}(\mathtt{x}_n)\rangle\rangle \xrightarrow{*} \langle q', \langle \mathfrak{v}(\mathtt{x}'_1), \ldots, \mathfrak{v}(\mathtt{x}'_n)\rangle\rangle$.

3.5 Flattable Presburger Counter Systems

As observed in [CJ98, FL02, Ler03, BIL09], flatness is very often essential to get effective semilinear reachability sets (but of course this is not a necessary condition, see e.g. [HP79]). However, flat Presburger counter systems are seldom natural in real-life applications. That is why, a relaxed version of flatness has been considered in [FO97, Fri00, LS05, DFGvD10] so that an initialized Presburger counter system $\langle \mathtt{C}, \langle q_0, \boldsymbol{x_0}\rangle\rangle$ is *flattable* whenever there is a partial unfolding of $\langle \mathtt{C}, \langle q_0, \boldsymbol{x_0}\rangle\rangle$ that is flat and has the same reachability set as $\langle \mathtt{C}, \langle q_0, \boldsymbol{x_0}\rangle\rangle$. In that way, reachability questions on $\langle \mathtt{C}, \langle q_0, \boldsymbol{x_0}\rangle\rangle$ can still be decided even in the absence of flatness. $\langle \mathtt{C}, \langle q_0, \boldsymbol{x_0}\rangle\rangle$ is *initially flattable* [LS05] iff there is a *a finite* set of path schemas such that the configurations reachable from $\langle q_0, \boldsymbol{x_0}\rangle$ are those reachable by firing the sequences of transitions from one of those path schemas (not every such sequence leads to a run). For instance, reversal-bounded initialized counter systems are initially flattable [LS05]. The fact that $\langle \mathtt{C}, \langle q_0, \boldsymbol{x_0}\rangle\rangle$ is flattable means that as far as reachability is concerned, a finite set of path schemas captures the full reachability relation. Note that flat counter systems are (structurally) flattable but in general it is non-trivial to compute such a finite set of path schemas, see also Section 5. This problem is also known as the problem of finite *good accelerations* [FL02, Section 5].

4 Verifying Temporal Properties

Reachability problems asks for the existence of runs reaching some configuration or control state in some specific way. Often, it is desirable to check how events are temporally organized along a run and to specify such properties temporal logics have been advocated since [Pnu77]. Furthermore, we wish to include in the logical language the possibility to express directly constraints between variables of the program, whence giving up the standard abstraction made with propositional variables. When the variables are typed, they may be interpreted in some specific domain like integers, real numbers, strings and so on; reasoning in such theories can be performed thanks to satisfiability modulo theories proof techniques, see e.g., [BSST08] and [GNRZ07] in which SMT solvers are used for model-checking infinite-state systems. Hence, a proposition like "$\mathtt{x}$ is greater than the next value

of y" can be encoded in such extended temporal logics by $\mathtt{x} > \mathtt{X}\mathtt{y}$ but this time the models are sequences of configurations. This means that each position comes with a control state and a valuation for variables. Hence, the basic idea behind the design of the logic Presburger LTL is to refine the language of atomic formulae and to allow the possibility to compare counter values at successive positions of the run of the counter systems. Similar motivations can be found in the introduction of concrete domains in description logics, that are logic-based formalisms for knowledge representation, see e.g. [Lut04].

4.1 Presburger LTL

We define below a version of linear-time temporal logic LTL dedicated to Presburger counter systems in which the atomic formulae are Presburger formulae about counter values, the temporal operators are those of LTL and first-order quantification over natural numbers is allowed, although we shall use it in a restricted way. Similarly, in [MP95], a mixture of first-order logic and LTL is shown sufficient to precisely state verification problems for the class of reactive systems.

We introduce a countable set of *integer variables*, say $\mathrm{VAR}^{\mathrm{P}} = \{\mathtt{y}_1, \mathtt{y}_2, \ldots\}$, for quantification over natural numbers. Elements of $\mathrm{VAR}^{\mathrm{P}}$ are distinct from the *counter variables* in $\mathrm{VAR} = \{\mathtt{x}_1, \mathtt{x}_2, \ldots\}$ that are free variables, only interpreted by the counter values on configurations. We also consider a countably infinite set $\mathtt{Q} = \{q_1, q_2, \ldots\}$ of control state symbols. The Presburger LTL formulae are defined as follows: $\phi ::= \psi \mid q \mid \phi \wedge \phi \mid \neg\phi \mid \mathtt{X}\phi \mid \phi \mathtt{U} \phi \mid \exists\, \mathtt{y}\ \phi$, where ψ is a Presburger formula with free variables in $\mathrm{VAR}^{\mathrm{P}} \cup \mathrm{VAR}$ from the linear fragment of (PA) and $q \in \mathtt{Q}$. The symbols $\mathtt{X}$ and $\mathtt{U}$ are respectively the classical operators next-time and until from LTL.

The models of Presburger LTL formulae are infinite runs from Presburger counter systems whose set of control states is included in the countable set $\mathtt{Q}$. A *model ρ of dimension n for* Presburger LTL is an element of $(Q \times \mathbb{N}^n)^\omega$ for some finite subset $Q \subseteq \mathtt{Q}$. An *environment* $\mathcal{E}$ is a partial map $\mathrm{VAR}^{\mathrm{P}} \to \mathbb{N}$. The *empty environment* is denoted by $\emptyset$. The satisfiability relation $\models$ is defined as follows between a model ρ of dimension n, a position $i \geq 0$, an environment $\mathcal{E}$ and a formula in which the free variables are among $\mathrm{VAR}^{\mathrm{P}} \cup \{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$.

The relation $\models_{\mathcal{E}}$ is defined on runs $\rho = \langle q_0, \boldsymbol{x_0}\rangle, \ldots, \langle q_k, \boldsymbol{x_k}\rangle, \ldots$ such that:

- $\rho, i \models_{\mathcal{E}} q \stackrel{\text{def}}{\Leftrightarrow} q = q_i$,
- When ψ is a Presburger formula from the linear fragment with free variables included in $\mathrm{VAR}^{\mathrm{P}} \cup \{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$, we have $\rho, i \models_{\mathcal{E}} \psi \stackrel{\text{def}}{\Leftrightarrow} \mathfrak{v}_i \models \psi$ in Presburger arithmetic where $\mathfrak{v}_i$ is a conservative extension of $\mathcal{E}$ such that for every $j \in [1, n]$, $\mathfrak{v}_i(\mathtt{x}_j) = \boldsymbol{x_i}(j)$,
- $\rho, i \models_{\mathcal{E}} \neg\phi \stackrel{\text{def}}{\Leftrightarrow} \rho, i \not\models_{\mathcal{E}} \phi$,
- $\rho, i \models_{\mathcal{E}} \phi_1 \wedge \phi_2 \stackrel{\text{def}}{\Leftrightarrow} \rho, i \models_{\mathcal{E}} \phi_1$ and $\rho, i \models_{\mathcal{E}} \phi_2$,
- $\rho, i \models_{\mathcal{E}} \mathtt{X}\phi \stackrel{\text{def}}{\Leftrightarrow} \rho, i+1 \models_{\mathcal{E}} \phi$,
- $\rho, i \models_{\mathcal{E}} \phi_1 \mathtt{U} \phi_2 \stackrel{\text{def}}{\Leftrightarrow}$ there is $j \geq i$ such that $\rho, j \models_{\mathcal{E}} \phi_2$ and $\rho, k \models_{\mathcal{E}} \phi_1$ for all $i \leq k < j$.
- $\rho, i \models_{\mathcal{E}} \exists\, \mathtt{y}\ \phi$ iff there is a natural number $m \in \mathbb{N}$ such that $\rho, i \models_{\mathcal{E}[\mathtt{y} \mapsto m]} \phi$.

As usual, we pose $\mathtt{F}\phi \stackrel{\text{def}}{=} \top\mathtt{U}\phi$ and $\mathtt{G}\phi \stackrel{\text{def}}{=} \neg\mathtt{F}\neg\phi$. Semantics with finite runs instead of infinite runs can be defined similarly. A *semi-closed formula* is an Presburger LTL formula such that no integer variable from $\mathrm{VAR}^{\mathrm{P}}$ is free. By construction, the counter variables $\mathtt{x}_1, \ldots, \mathtt{x}_n$ are always free and are interpreted as the current counter values. In the decision problems defined below, we only consider semi-closed formulae and therefore there is no need to specify an environment in the statements.

For instance, one can express that the first counter strictly increases at every step: $\mathtt{G}\,\exists\,\mathtt{y}\;(\mathtt{y} = \mathtt{x}_1 \wedge \mathtt{X}(\mathtt{x}_1 > \mathtt{y}))$. Similarly, the first counter takes a finite number of values along the run can be expressed by $\exists\,\mathtt{y}\;\mathtt{G}(\mathtt{x}_1 \leq \mathtt{y})$.

Let us start by presenting the *satisfiability problem* for Presburger LTL:

Input: A Presburger LTL semi-closed formula ϕ with free counter variables $\mathtt{x}_1$, $\ldots$, $\mathtt{x}_n$.
Question: Is there a model $\rho \in (\mathtt{Q} \times \mathbb{N}^n)$ of dimension n such that $\rho, 0 \models_\emptyset \phi$?

Observe that for satisfiability checking, it is not necessary that the model is derived from a Presburger counter system. Let us turn to *existential model-checking problem* for Presburger LTL:

Input: A Presburger counter system $\mathtt{C} = \langle Q, n, \delta\rangle$, an initial configuration $\langle q_0, \boldsymbol{x}_0\rangle$ and a semi-closed formula ϕ in Presburger LTL.
Question: Is there an infinite run ρ starting at $\langle q_0, \boldsymbol{x}_0\rangle$ such that $\rho, 0 \models_\emptyset \phi$?

Temporal logics with Presburger constraints has been developed, for instance, in [BEH95, CC00, BDR03]. Some of them have quite expressive decidable fragments. Undecidability of the existential model-checking problem for Presburger LTL can be shown using the undecidability of the halting problem for Minsky machines, see e.g., [CC00]. Still, using SMT solvers can be done for checking bounded reachability problems, see e.g., [BFM$^+$10].

In the rest of this section, we present fragments of Presburger LTL obtained by restricting first-order quantification over natural numbers. First, let us observe that if we restrict ourselves to formulae in which temporal operators are not in the scope of first-order quantification, then we get a fragment of Presburger LTL that is very similar to plain LTL. Indeed, atomic formulae are arithmetical constraints between counter values and they can be understood as high-level propositional variables; whence the automata-based approach for LTL can be easily adapted to this fragment. In that fragment, the arithmetical constraints are only local and in the construction of Büchi automata, the existence of transitions between states depends on the satisfiability status of Presburger formulae. Below, we provide restrictions in which the temporal operators may occur in the scope of first-order quantification.

Comparing successive counter values. Given a Presburger formula $\psi(\mathtt{z}_1, \ldots, \mathtt{z}_k)$, we shall write $\psi(\mathtt{X}^{i_1}\mathtt{x}_{j_1}, \ldots, \mathtt{X}^{i_k}\mathtt{x}_{j_k})$ to denote the formula below

$$(\exists\,\mathtt{y}_1, \ldots, \mathtt{y}_k\;\mathtt{X}^{i_1}(\mathtt{y}_1 = \mathtt{x}_{j_1}) \wedge \cdots \wedge \mathtt{X}^{i_k}(\mathtt{y}_k = \mathtt{x}_{j_k}) \wedge \psi(\mathtt{y}_1, \ldots, \mathtt{y}_k),$$

where $\mathtt{y}_1, \ldots, \mathtt{y}_k$ are new variables distinct from the free variables that are present in $\psi(\mathtt{z}_1, \ldots, \mathtt{z}_k)$. It is easy to see that $\psi(\mathtt{X}^{i_1}\mathtt{x}_{j_1}, \ldots, \mathtt{X}^{i_k}\mathtt{x}_{j_k})$ is interpreted as the formula $\psi(\mathtt{z}_1, \ldots, \mathtt{z}_k)$ in which each variable $\mathtt{z}_a$ takes the value of $\mathtt{x}_{j_a}$ at the i_ath next configuration. For instance, $\mathtt{x}_1 = \mathtt{X}\mathtt{x}_2$ specifies that the next value of $\mathtt{x}_2$ is equal to the current value of $\mathtt{x}_1$. Similarly, $\mathtt{G}(\mathtt{x}_1 = \mathtt{X}\mathtt{x}_1)$ states that counter 1 has a constant value along the model. In Section 4.2, we present a simple fragment of Presburger LTL by allowing first-order quantification only for formulae of the form $\psi(\mathtt{X}^{i_1}\mathtt{x}_{j_1}, \ldots, \mathtt{X}^{i_k}\mathtt{x}_{j_k})$ and Presburger formulae (at the atomic level) are quantifier-free too. It is worth observing that we use '$\mathtt{X}$' as the next-time temporal operator whereas '$\mathtt{X}\mathtt{x}$' refers to the value of $\mathtt{x}$ at the next position.

Freeze operator. In order to verify properties on Presburger counter systems, we want also to be able to compare counter values. For that, it is possible to define the so-called 'freeze operator' with formulae of the form $\downarrow_r^j \phi$ interpreted as $\exists\, \mathtt{y}_r\ (\mathtt{y}_r = \mathtt{x}_j \wedge \phi)$ that store counter values. There are counterpart formulae of the form $\uparrow_r^j$ interpreted as $\mathtt{y}_r = \mathtt{x}_j$ that perform equality tests. Intuitively, the modality $\downarrow_r^j$ is used to store the value of the counter j into the register r; the atomic formula $\uparrow_r^j$ holds true if the value stored in the register r is equal to the current value of the counter j. The formula $\mathtt{G}(\downarrow_1^1 \mathtt{XG}\neg \uparrow_1^1)$ states that the first counter has distinct values at distinct positions. Freeze operator has been introduced in numerous works, sometimes with different motivations, see e.g. [Hen90, Gor94, Fit02, LP05]. It is also sometimes used implicitly as for the temporal semantics for imperative programs that may use first-order temporal logics, see e.g. [MP92]. For instance, the statement that the program variable $\mathtt{x}$ never decreases below its initial value can be expressed by the formula below that uses a form of freeze operator: $\exists \mathtt{y}\ (\mathtt{x} = \mathtt{y}) \ \wedge\ \mathtt{G}(\mathtt{x} \geq \mathtt{y})$. Recent results on satisfiability and model-checking problems can be found in [FS09, DLS10].

4.2 The Logic CLTL with Finite Window

As mentioned earlier, we shall define the logic CLTL as a strict fragment of Presburger LTL such that first-order quantification at the level of temporal formulae is restricted to macro formulae of the form $\psi(\mathtt{X}^{i_1}\mathtt{x}_{j_1}, \ldots, \mathtt{X}^{i_k}\mathtt{x}_{j_k})$. Consequently, there is no more quantification over integer variables from $\mathrm{VAR}^{\mathrm{P}}$ and no variable in $\mathrm{VAR}^{\mathrm{P}}$ occurs in CLTL formulae. The logic CLTL has atomic formulae from the linear fragment of (PA) except that variables are replaced by expressions of the form $\mathtt{X}^i\mathtt{x}$ where $\mathtt{x} \in \mathrm{VAR}$ is a variable and $\mathtt{X}^i$ is understood as a sequence of i consecutive symbols $\mathtt{X}$. The expression $\mathtt{X}^i\mathtt{x}$ is interpreted as the value of $\mathtt{x}$ at the ith next state. Given a CLTL formula ϕ, we define its $\mathtt{X}$-length $\mathrm{len}(\phi)_{\mathtt{X}}$ as the maximal number i such that an expression of the form $\mathtt{X}^i\mathtt{x}$ occurs in ϕ. Intuitively, the $\mathtt{X}$-length defines the size of a frame/window of consecutive states that can be compared. The models of CLTL are pairs of sequences $\sigma = \langle \sigma_1, \sigma_2 \rangle$ such that $\sigma_1 : \mathbb{N} \to (\mathrm{VAR} \to \mathbb{N})$, $\sigma_2 : \mathbb{N} \to Q$ for a finite subset $Q \subseteq \mathtt{Q}$. The satisfaction relation is defined as for LTL except at the atomic level:

- $\sigma, i \models q$ iff $\sigma_2(i) = q$,
- $\sigma, i \models \psi(\mathtt{X}^{l_1}\mathsf{x}_1, \ldots, \mathtt{X}^{l_n}\mathsf{x}_n)$ iff $\langle \sigma_1(i+l_1)(\mathsf{x}_1), \ldots, \sigma_1(i+l_n)(\mathsf{x}_n)\rangle \in [\![\psi]\!]$.
- $\sigma, i \models \mathtt{X}\phi$ iff $\sigma, i+1 \models \phi$,
- $\sigma, i \models \phi \mathtt{U} \phi'$ iff there is $j \geq i$ such that $\sigma, j \models \phi'$ and for every $i \leq l < j$, we have $\sigma, l \models \phi$.

As usual, a formula $\phi \in \mathrm{CLTL}$ is satisfiable whenever there exists a model σ such that $\sigma, 0 \models \phi$. We write CLTL^l_n to denote the restriction of CLTL to formulae with at most n variables and $\mathtt{X}$-length less or equal to l (below the value ω is used for some syntactic resource when there is no restriction). CLTL^0 denote the fragment in which the arithmetical constraints deal only with current counter values. Note that there is a logspace reduction from the satisfiability problem for CLTL to the satisfiability problem for CLTL_ω restricted to formulae of $\mathtt{X}$-length at most 1 (CLTL^1_ω), see e.g. [DLN07]. The proof is done by renaming terms and requires an unbounded amount of variables in CLTL^1_ω. For instance, the expressions $\mathsf{x}_1, \ldots, \mathtt{X}^3\mathsf{x}_1$ are encoded by $\mathtt{G}(\mathsf{x}'' = \mathtt{X}\mathsf{x}' \wedge \mathsf{x}' = \mathtt{X}\mathsf{x} \wedge \mathsf{x} = \mathtt{X}\mathsf{x}_1)$ (assuming that x, x' and x'' are new variables) and each occurrence of $\mathtt{X}\mathsf{x}_1$ [resp. $\mathtt{X}^2\mathsf{x}_1$, $\mathtt{X}^3\mathsf{x}_1$] is replaced by x [resp. x', x'']. For reductions between satisfiability problems, the introduction of new variables is harmless.

The halting problem for Minsky machines can be easily reduced to the satisfiability problem for CLTL or to the existential model-checking problem for CLTL, leading to simple undecidability proofs. In the sequel, we show how to restrict the class of counter systems or the logical language in order to regain decidability.

Given a fragment $\mathfrak{F}$ of (PA) (not necessarily restricted to the linear fragment as for CLTL), we write $\mathrm{CLTL}(\mathfrak{F})$ to denote the variant of CLTL in which atomic formulae built over the quantifier-free linear fragment of (PA) are replaced by formulae from $\mathfrak{F}$ (the definition of the satisfaction relation is update accordingly without significant changes). Similarly, we write $\mathrm{CLTL}^l_n(\mathfrak{F})$ ($n \geq 1$, $l \geq 0$) to denote the restriction of $\mathrm{CLTL}(\mathfrak{F})$ to formulae such that the variables are among $\{\mathsf{x}_1, \ldots, \mathsf{x}_n\}$ and the $\mathtt{X}$-length is bounded by l.

Fragment $\mathfrak{F}_0$ is defined as follows:

$$\mathfrak{F}_0 \ni \phi ::= \mathsf{x}_i < \mathsf{x}_j \mid \mathsf{x}_i = \mathsf{x}_j \mid \mathsf{x}_i \leq k$$

where $k \in \mathbb{N}$. Fragment $\mathfrak{F}_1$ is defined as follows:

$$\mathfrak{F}_1 \ni \phi ::= \mathsf{x}_i \sim \mathsf{x}_j + d \mid \mathsf{x}_i \sim d$$

where $d \in \mathbb{Z}$ and $\sim \in \{<, >, \leq, \geq, =\}$. For instance, $\mathsf{x}_1 = \mathtt{X}^8\mathsf{x}_2 + 1 \in \mathrm{CLTL}^8_2(\mathfrak{F}_1)$ and $\mathtt{XXX}(\mathtt{X}\mathsf{x}_1 \geq 27) \in \mathrm{CLTL}^1_1(\mathfrak{F}_0)$. The logic CLTL defined in [CC00] is precisely $\mathrm{CLTL}^1_\omega(\mathfrak{F}_1)$. By way of example, let us quote a few interesting results.

Theorem 13. *(I) Satisfiability problem for* $\mathrm{CLTL}(\mathfrak{F}_0)$ *is* PSPACE*-complete, see e.g. [DD07, DG08, ST11]. (II) Satisfiability problem for* $\mathrm{CLTL}^1_1(\mathfrak{F}_1)$ *is* PSPACE*-complete [DG09]. (III) Satisfiability problem for* $\mathrm{CLTL}^2_1(\mathfrak{F}_1)$ *or for* $\mathrm{CLTL}^1_2(\mathfrak{F}_1)$ *is undecidable [DG09], see also [CC00].*

4.3 Model-Checking Linear-Time Properties

Flat counter systems. Below, we state several recent results about model-checking problems for subclasses of counter systems for which the complexity is relatively low. We recall that guards in counter systems belong to the linear fragment and the updates are in $\mathbb{Z}^n$.

Theorem 14. *[DDS12] Model-checking flat counter systems with* CLTL^0 *is* NP*-complete (also holds with past-time temporal operators).*

The NP upper bound is obtained as follows given a flat counter system with initial configuration $\langle q_0, \boldsymbol{x_0}\rangle$ and a formula ϕ in CLTL^0:

- Guess a good infinitary path schema $\mathtt{P}$ from $\langle q_0, \boldsymbol{x_0}\rangle$. Infinitary path schemas are of the form $\mathtt{p}_1\mathtt{l}_2^*\mathtt{p}_3\mathtt{l}_4^*\ldots\mathtt{l}_{k-1}^*\mathtt{p}_k\mathtt{l}_k^\omega$.
- Guess an *unfolded path schema* $\mathtt{P}'$ from $\mathtt{P}$ by eliminating disjunctions in guards and counter values (but at the cost of adding new atomic propositions). Unfolding a path schema amounts to copying loops a (polynomial) number of times while adding atomic propositions or constraints in guards to guarantee that each visit in a new loop satisfies the same guards.
- Build an existential Presburger formula that encodes all the runs respecting $\mathtt{P}'$ from $\langle q_0, \boldsymbol{x_0}\rangle$ (all the quantified variables are loop counters).
- Guess a run respecting $\mathtt{P}'$ and check whether it satisfies ϕ symbolically. Each loop may be visited an exponential number of times, but a stuttering theorem allows the symbolic model-checking algorithm to perform efficiently.

Efficient solvers for quantifier-free (PA) are required to make feasible the above algorithm. By contrast, we get a little higher complexity with linear μ-calculus or first-order logic (FOL).

Theorem 15. *[DDS13] Model-checking flat counter systems with linear* μ*-calculus or with FOL both with arithmetical constraints is* PSpace*-complete.*

It is unclear what are the counterpart results for flat Presburger counter systems with octagonal constraints.

LTL on VASS. Structural restrictions seem more efficient to reduce the computational complexity of temporal model-checking. By way of comparison, model-checking vector addition systems with states with linear μ-linear calculus (without arithmetical constraints) is already ExpSpace-complete [Hab97].

Let us present a fragment of Presburger LTL introduced in [Jan90] such that the atomic formulae are either control states or atomic formulae of the form $\mathtt{x}_i \geq k$ or $\neg(\mathtt{x}_i \geq k)$ with $k \in \mathbb{N}$. The temporal logic with fairness TLF is defined as a logic on VASS for which formulae are defined by the grammar $q \mid \mathtt{x}_i \geq k \mid \neg(\mathtt{x}_i \geq k) \mid \phi \vee \phi \mid \phi \wedge \phi \mid \mathtt{GF}\phi$, where $q \in \mathtt{Q}$ and $k \in \mathbb{N}$. Observe that TLF formulae are not closed under negations and the temporal properties are intersection or union of fairness conditions. Decidability of (existential) model-checking problem for TLF restricted to VASS is established in [Jan90] by

reduction into the reachability problem for VASS. Fairness conditions on VASS can be also found in [GS92]. Moreover, it is worth noting that the operator $\mathtt{F}$ cannot be expressed in TLF, otherwise undecidability would hold. Indeed, in [HR89] a linear-time temporal logic (on Petri nets) is shown undecidable with the temporal operator $\mathtt{F}$, Boolean connectives and atomic formulae of the form $\mathtt{x}_i \geq k$ and "transition t is the next one in the run." Finally, other logical formalisms interpreted on VASS runs can be found in [Esp94, AH11, BS11] where complexity/decidability results are established.

Bounding the number of reversals. Results about model-checking reversal-bounded counter systems with LTL equipped with arithmetical constraints can be found in [BD11, HL11]. Below, we recall the definition for the reversal-bounded model-checking problem (RBMC). Its peculiarity is that the input initialized counter systems are not necessarily reversal-bounded but the input contains an explicit bound r about the maximal number of reversals within a run. Moreover, given a formula ϕ in CLTL, we write $\mathtt{T}_\phi$ to denote the terms of the form $\mathtt{t} \sim k$ occurring in it. The problem RBMC is defined as follows:

Input: a counter system $\mathtt{C}$, an initial configuration $\langle q_0, \boldsymbol{x}_0 \rangle$, a formula CLTL ϕ (with atomic formulae of the form $\mathtt{t} \sim k$), a bound $r \in \mathbb{N}$ (in binary),
Question: Is there an infinite run ρ from $\langle q, \boldsymbol{x} \rangle$ such that $\rho, 0 \models \phi$ and ρ is r-$\mathtt{T}$-reversal-bounded with $\mathtt{T} = \mathtt{T}_\mathtt{C} \cup \mathtt{T}_\phi$?

The computational complexity for RBMC can be precisely characterized; the upper bound can be obtained by a refined analysis on runs, see e.g. [GI81, BD11].

Theorem 16. *[BD11, HL11] RBMC is* NExpTime*-complete.*

Actually, one can also establish that global model-checking is possible for RBMC [BD11], i.e., the set of initial configurations for which there is a run satisfying a given temporal formula from CLTL is effectively Presburger-definable.

4.4 A Quick Look at a Branching-Time Extension

The logic Presburger LTL is interpreted on linear runs but it is possible to extend it to its CTL* version by interpreting the formulae on the underlying transition systems of the Presburger counter systems and by adding quantifications over paths, see e.g. [BG06, DFGvD10]. The formulae for Presburger CTL* are defined as follows: $\phi ::= \psi \mid q \mid \phi \wedge \phi \mid \neg\phi \mid \mathtt{X}\phi \mid \phi \mathtt{U} \phi \mid \mathtt{A}\ \phi \mid \exists\, \mathtt{y}\ \phi$ where ψ is a Presburger formula with free variables included in $\mathrm{VAR}^\mathrm{p} \cup \mathrm{VAR}$ from the linear fragment and $q \in \mathtt{Q}$. Semantics for Presburger CTL* is provided via models that are transition systems obtained from Presburger counter systems. Again, the satisfaction relation $\models$ is parameterized by an *environment* $\mathcal{E}$. Given a Presburger counter system $\mathtt{C} = \langle Q, n, \delta \rangle$ with transition system $\mathfrak{T}(\mathtt{C}) = \langle \mathfrak{S}, \rightarrow \rangle$, the satisfaction relation $\models_\mathcal{E}$ is defined at position i of the run as for Presburger LTL except for quantifications over paths: $\rho, i \models_\mathcal{E} \mathtt{A}\ \ \phi \stackrel{\text{def}}{\Leftrightarrow}$ for all infinite runs ρ' starting at configuration $\rho(i)$, we have $\rho', 0 \models_\mathcal{E} \phi$. First-order quantification over counter

values allows us to state many interesting properties in Presburger CTL* such as determinism (for all the configurations reachable from the initial configuration, there is at most one successor configuration):

$$\mathtt{A}\ \mathtt{G}(\bigwedge_{i\in[1,n]} \neg\exists \mathsf{y}(\mathtt{E}\ \mathtt{X}(\mathsf{x}_i = \mathsf{y}) \wedge \mathtt{E}\ \mathtt{X}(\mathsf{x}_i \neq \mathsf{y}))) \wedge (\bigwedge_{q\in Q} \neg(\mathtt{E}\ \mathtt{X}q \wedge \mathtt{E}\ \mathtt{X}\neg q)).$$

Similarly, boundedness (the set of configurations reachable from the initial configuration is finite) can be stated with $\exists \mathsf{y}, \mathsf{y}'\ \mathtt{A}\ \mathtt{G} \bigwedge_{i\in[1,n]} \mathsf{y} \leq \mathsf{x}_i \leq \mathsf{y}'$.

Model-checking problem for Presburger CTL* is defined as follows: given a Presburger counter system $\mathtt{C}$ with transition system $\mathfrak{T}(\mathtt{C}) = \langle \mathfrak{S}, \rightarrow \rangle$, an initial configuration $\langle q_0, \boldsymbol{x}_0 \rangle$, and a semi-closed formula ϕ from Presburger CTL*, determine whether for every run ρ from $\langle q_0, \boldsymbol{x}_0 \rangle$, we have $\rho, 0 \models \phi$.

Theorem 17. *[DFGvD10] Model-checking admissible Presburger counter systems with Presburger CTL* is decidable.*

The proof provides a reduction into satisfiability in (PA) by encoding the runs by tuples of natural numbers. Indeed, every admissible Presburger counter system is flat and therefore it has a finite amount of good path schemas. Runs respecting a path schema can be encoded as tuples of natural numbers by counting how many times the loops are visited. Temporal formulae are then encoded by internalizing the semantics into (PA) itself. Other decidability and complexity results can be found in [BP12, CKL13]. Nevertheless, it remains open whether modal μ-calculus (with atomic arithmetical constraints on counter values) can be shown decidable on admissible Presburger counter systems.

5 Path Schema Enumeration

In this section, we explain the interest of designing algorithms for the enumeration of finitary path schemas and how to prune the search space by subsumption. We present a preliminary version of an algorithm for enumerating path schemas. More details and developments will be provided in a forthcoming paper. Only finitary path schemas are discussed in this section but infinitary ones could be generated in a similar way. Moreover, we assume that we are dealing with a class of Presburger counter systems satisfying the property ($\star$), recalling Definition 4 on page 129, such as the class of counter systems where the guards are Boolean combinations of linear constraints and the updates in $\mathbb{Z}^n$ are those from VASS.

5.1 Why Path Schema Enumeration?

As is well-known, Presburger counter systems are Turing-complete and it is undecidable to check the existence of a run satisfying a given property (even for very basic ones). However, approximating the Presburger counter systems by looking at a subclass of runs provides a means to produce answers in some cases. For example, a finite set of path schemas is a simple way to represent

a (potentially) infinite set of runs. Being able to generate path schemas in a structured and controlled fashion while using structural properties of the control graph as well as arithmetical constraints of counter values will be helpful to test the existence of runs satisfying some property.

A wish list for generating path schemas. Even though it is not difficult to generate path schemas in a fair and complete way by tracing the transitions, the details of the enumeration are quite important but often underestimated, see e.g.. [BFLS05, DFGvD10] (see some exception in [Ler03]). First, we want an enumeration strategy that is efficient in practice. Previous work has left open the question of efficient enumeration of path schemas, as the results have been of a theoretical nature, and path schemas didn't have to be enumerated explicitly. Second, we want an enumeration strategy that will find a finite set of path schemas that fully captures the behavior of a Presburger counter system, if possible. As noted earlier, some classes of Presburger counter systems are known to have a finite number of good path schemas. For flat Presburger counter systems, the number of good path schemas is exponential in the size of the control graph whereas for flattable initialized Presburger counter systems, the number of path schemas is finite but with no specific bound on this number. Finally, we want an enumeration strategy in which we have a clear way of detecting whether we have enumerated sufficiently many path schemas to capture the behavior of the Presburger counter system.

Enumerating path schemas can also be viewed as a way to underapproximate the set of runs; this is similar to a standard approach to consider subclasses of runs by bounding some features and to search for 'bounded runs' that may satisfy a desirable or undesirable property. Examples include reversal-bounded counter machines (which have a bound on the number of reversals) [HR87, BD11, HL11], context-bounded model-checking (where there is a bound on the number of context switches) [QR05], and of course bounded model-checking (BMC) (where there is a bound on the distance of the reached positions), see e.g. [BCC$^+$03].

5.2 Pruning the Search Space: Path Schema Subsumption

Let $\mathtt{C} = \langle Q, n, \delta \rangle$ be a Presburger counter system and $\mathtt{P}_1, \ldots, \mathtt{P}_N, \mathtt{P}$ be finitary path schemas such that $\mathit{first}(\mathtt{P}_1) = \cdots = \mathit{first}(\mathtt{P}_N) = \mathit{first}(\mathtt{P})$ and $\mathit{last}(\mathtt{P}_1) = \cdots = \mathit{last}(\mathtt{P}_N) = \mathit{last}(\mathtt{P})$, i.e., all the path schemas start and end by the same control states. One can think of $\mathtt{P}_1, \ldots, \mathtt{P}_N$ as the path schemas already in our database whereas $\mathtt{P}$ is a new path schema for which we have to decide whether we keep it or not. Such a path schema $\mathtt{P}$ must be consistent, i.e., there exists a finite run that respects it. The path schema $\mathtt{P}$ is *consistent* w.r.t. the initial condition $\phi_{\mathrm{init}}(\mathtt{y}_1, \ldots, \mathtt{y}_n)$ iff the formula $\exists\, \mathtt{x}_1, \ldots, \mathtt{x}_n\ \exists\, \mathtt{x}'_1, \ldots, \mathtt{x}'_n\ \phi_{\mathrm{init}}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \wedge \varphi_{\mathtt{P}}(\mathtt{x}_1, \ldots, \mathtt{x}_n, \mathtt{x}'_1, \ldots, \mathtt{x}'_n)$ is valid. Then, comes subsumption. The set $\{\mathtt{P}_1, \ldots, \mathtt{P}_N\}$ *subsumes* $\mathtt{P}$ *w.r.t. the initial condition* $\phi_{\mathrm{init}}(\mathtt{y}_1, \ldots, \mathtt{y}_n)$ (and with respect to reachability) $\overset{\text{def}}{\Leftrightarrow}$ the formula below is valid:

$$\forall\, \mathtt{x}_1, \ldots, \mathtt{x}_n\ \forall\, \mathtt{x}'_1, \ldots, \mathtt{x}'_n\ (\phi_{\mathrm{init}}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \wedge \varphi_{\mathtt{P}}(\mathtt{x}_1, \ldots, \mathtt{x}_n, \mathtt{x}'_1, \ldots, \mathtt{x}'_n)) \Rightarrow$$

$$\bigvee_{i\in[1,N]} \exists\, \mathsf{z}_1,\ldots,\mathsf{z}_n\ \phi_{\text{init}}(\mathsf{z}_1,\ldots,\mathsf{z}_n) \wedge \varphi_{\mathtt{P}_i}(\mathsf{z}_1,\ldots,\mathsf{z}_n,\mathsf{x}'_1,\ldots,\mathsf{x}'_n)$$

For the class of counter systems, the consistency problem is NP-complete, and the subsumption problem can be expressed in the fragment of (PA) with at most one quantifier alternation. The above subsumption notion can be also formulated as follows. A path schema $\mathtt{P}$ enriched with $\phi_{\text{init}}(\mathsf{y}_1,\ldots,\mathsf{y}_n)$ defines a set of finite runs such that the initial counter values satisfy $\phi_{\text{init}}(\mathsf{y}_1,\ldots,\mathsf{y}_n)$ and the run respects $\mathtt{P}$. Moreover, such a pair defines a set of counter values—those that have been reached at the end of the runs, say $[\mathtt{P}]_{\phi_{\text{init}}} \stackrel{\text{def}}{=} \{\boldsymbol{x_f} : \langle q_0, \boldsymbol{x_0}\rangle \xrightarrow{*} \langle q_f, \boldsymbol{x_f}\rangle$ respects $\mathtt{P}$ and $\phi_{\text{init}}(\boldsymbol{x_0})\}$. Counter values are therefore extracted from runs. Now, subsumption can be formulated as follows: $[\mathtt{P}]_{\phi_{\text{init}}} \subseteq [\mathtt{P}_1]_{\phi_{\text{init}}} \cup \cdots \cup [\mathtt{P}_N]_{\phi_{\text{init}}}$. There exist other means to extract witness counter values from runs. A *pattern* ϕ_{pat} is a formula from Presburger LTL without first-order quantification (see Section 4) and with free occurrences of the integer variables $\mathsf{y}_1, \ldots. \mathsf{y}_\alpha$ that are therefore interpreted rigidly. By way of example, we consider the version of Presburger LTL on finite runs. Let us define the set of tuples $[\mathtt{P}]_{\phi_{\text{pat}},\phi_{\text{init}}}$ obtained by extracting the parameter values from runs respecting $\mathtt{P}$ and whose initial configuration satisfies ϕ_{init} (see the semantics in Section 4):

$$[\mathtt{P}]_{\phi_{\text{pat}},\phi_{\text{init}}} \stackrel{\text{def}}{=} \{\mathcal{E} : \rho = \langle q_0, \boldsymbol{x_0}\rangle \xrightarrow{*} \langle q_f, \boldsymbol{x_f}\rangle \text{ respects } \mathtt{P},\ \ \phi_{\text{init}}(\boldsymbol{x_0})\ \&\ \rho, 0 \models_{\mathcal{E}} \phi_{\text{pat}}\}$$

Note that $[\mathtt{P}]_{\phi_{\text{init}}}$ above corresponds to $[\mathtt{P}]_{\phi_{\text{pat}},\phi_{\text{init}}}$ with

$$\phi_{\text{pat}} \stackrel{\text{def}}{=} \mathtt{F}(\mathsf{x}_1 = \mathsf{y}_1 \wedge \cdots \wedge \mathsf{x}_n = \mathsf{y}_n \wedge \neg \mathtt{X}\top)$$

Let us define the generalized path schema subsumption problem: $\{\mathtt{P}_1,\ldots,\mathtt{P}_N\}$ *subsumes* $\mathtt{P}$ *with respect to the initial condition* $\phi_{\text{init}}(\mathsf{y}_1,\ldots,\mathsf{y}_n)$ *and the property/pattern* ϕ_{pat} $\stackrel{\text{def}}{\Leftrightarrow}$ $[\mathtt{P}]_{\phi_{\text{pat}},\phi_{\text{init}}} \subseteq [\mathtt{P}_1]_{\phi_{\text{pat}},\phi_{\text{init}}} \cup \cdots \cup [\mathtt{P}_N]_{\phi_{\text{pat}},\phi_{\text{init}}}$ (of course, a Presburger counter system is also part of the instance).

Lemma 18. *For any class of Presburger counter systems satisfying (⋆), there is a reduction from the generalized path schema subsumption problem to the validity problem for (PA).*

The proof consists in encoding the runs satisfying a path schema by tuples and then to use the standard translation from LTL to first-order logic. The initial condition ϕ_{init} and atomic formulae in PLTL formulae are already Presburger formulae, so do not require special treatment in the translation process.

5.3 How to Deal with Quantifiers

Note that Presburger formulae built to perform subsumption tests contain quantifiers. Most well-known Satisfiability Modulo Theories (SMT) solvers can deal with quantifier-free formulae, also known as linear arithmetic (LIA). For instance, this includes Z3 [dMB08], CVC4 [BCD+11], and Alt-Ergo [Con12], to cite a few of them; see also Pugh's Omega test [Pug92].

However, as observed earlier, dealing with quantifiers is usually a difficult task for SMT solvers. Fortunately, quantifiers can be eliminated but this may be expensive computationally. Cooper's elimination procedure [Coo72], when considering $\exists\, \mathtt{x}\, \psi$ with quantifier-free ψ, does not assume that ψ is in disjunctive normal form (a disjunction of conjuncts, with conjuncts made of literals). This is a remarkable difference with the original algorithm presented in [Pre29]. Indeed, transforming a propositional formula into an equivalent formula in disjunctive normal form may cause an exponential blow-up. A more advanced improvement of Cooper's procedure can be found in [RL78]; recent developments propose a lazy approach to quantifier elimination [Mon10].

5.4 An Algorithm that Builds Cycle Schemas and Path Schemas

In this section, we present an algorithm to generate path schemas from a Presburger counter system. It proceeds by building path schemas of larger and larger sizes. An outer loop ensures that all path schemas of some constant size $k-1$ have been built before the generation of path schemas of size k is attempted. The algorithm is inherently iterative, and its first k iterations enumerate all path schemas of size less than or equal to k.

Path schemas are generated by building upon smaller path schemas. Given a path schema of size $k-1$, the algorithm extends it by adding a transition; the result is a new path schema of size k. Path schemas may also be extended by cycles, and for this, the algorithm detects and maintains *cycle schemas* (see below). These cycle schemas are detected by using smaller path schemas.

Preliminary definitions: cycle schemas and suffixes. A *cycle schema* L is a path schema starting and ending by the same control state. The set of cycles generated by a cycle schema L is precisely $\mathrm{Lan}(L)$. We write $\mathrm{Lan}^{\circlearrowleft}(L)$ to denote the smallest set of paths such that $\mathrm{Lan}(L) \subseteq \mathrm{Lan}^{\circlearrowleft}(L)$ and if $t_1 \cdots t_\alpha \in \mathrm{Lan}^{\circlearrowleft}(L)$, then $t_2 \cdots t_\alpha t_1 \in \mathrm{Lan}^{\circlearrowleft}(L)$. The set $\mathrm{Lan}^{\circlearrowleft}(L)$ can be also obtained from $\mathrm{Lan}(L)$ by considering all possible rotations of loops.

Path schemas can be concatenated assuming that constraints on control states are respected. Let $\mathtt{P} = \mathtt{P}_1 \cdot \mathtt{P}_2$ be a path schema obtained by concatenating two path schemas such that (1) $\mathtt{P}_2$ starts by q' and is of length at least one, (2) $\mathtt{P}_2$ ends by q, (3) there is a transition $t = q \xrightarrow{\phi} q'$. Obviously, $\mathtt{P}_2 \cdot t$ is a cycle schema. $\mathtt{P}_2$ is called a *suffix* of $\mathtt{P}$. Similarly, let $\mathtt{P} = \mathtt{P}_1 \cdot (\mathtt{l})^* \cdot \mathtt{P}_2$ be a path schema with $\mathtt{l} = \mathtt{p}_1 \cdot \mathtt{p}_2$, such that (1) $\mathtt{p}_2$ starts by q', (2) $\mathtt{P}_2$ ends by q, (3) there is a transition $t = q \xrightarrow{\phi} q'$. Obviously, $\mathtt{p}_2 \cdot (\mathtt{l})^* \cdot \mathtt{P}_2 \cdot t$ is another cycle schema. $\mathtt{p}_2 \cdot (\mathtt{l})^* \cdot \mathtt{P}_2$ is called an *augmented suffix* of $\mathtt{P}$. By definition, an augmented suffix is any suffix obtained in one of the two above-mentioned ways. A *simple suffix* is a suffix without a loop, i.e., a non-empty sequence of transitions being the suffix of a path schema.

ps-complexity. A path $\mathtt{p} \in \delta^*$ has *ps-complexity* $k \stackrel{\text{def}}{\Leftrightarrow}$ there is a path schema $\mathtt{P}$ of length k such that $\mathtt{p} \in \mathrm{Lan}(\mathtt{P})$ and no path schema $\mathtt{P}'$ of strictly smaller

size verifies $\mathtt{p} \in \mathrm{Lan}(\mathtt{P}')$. In a sense, the ps-complexity of a path measures how concisely the path can be represented with the help of path schemas generated from the Presburger counter system. Similarly, a run ρ has *ps-complexity* $k \stackrel{\text{def}}{\Leftrightarrow}$ there is a path schema $\mathtt{P}$ of length k such that ρ respects $\mathtt{P}$ and for no path schema $\mathtt{P}'$ of strictly smaller size, ρ respects $\mathtt{P}'$. The *relative length* of a loop $\mathtt{l}$ with respect to control state q_0 is equal to the length of $\mathtt{l}$ plus the minimal distance between the initial control state q_0 and a control state occurring in $\mathtt{l}$ (can be infinite, can be equal to the length of $\mathtt{l}$ if q_0 occurs in $\mathtt{l}$). Again, no constraints about counter values are involved at this stage.

Subsumption test. Our algorithm is parameterized by a subsumption test. When a path schema is subsumed by the set of path schemas previously discovered, it is not itself enumerated. This leads to less redundant results, and to less work being done by the algorithm at larger k. It also leads to an easy termination test: during a level k, if no new path schemas are enumerated, then the algorithm has already enumerated a finite set of path schemas that "capture" the behavior of the system (w.r.t. the subsumption test).

The algorithm. Below, we present the algorithm in which $PS(k)$ is the set of path schemas of size k discovered. $CS(k)$ is the set of cycle schemas discovered with relative length k. We write $PS^+(k) \stackrel{\text{def}}{=} \bigcup_{k' \leq k} PS(k')$ and $CS^+(k) \stackrel{\text{def}}{=} \bigcup_{k' \leq k} CS(k')$. The input is a Presburger counter system $\mathtt{C}$, with initial state q_0 and initial condition $\phi_{\text{init}}(\mathtt{x}_1, \ldots, \mathtt{x}_n)$. The input contains a path schema eligibility test (called *test*) so that $PS(k) \stackrel{test}{\longleftarrow} \mathtt{P}$ is a shorthand for: `if` $test(\phi_{\text{init}}, \mathtt{P}, PS^+(k))$ `then` $PS(k) \longleftarrow PS(k) \cup \{\mathtt{P}\}$. When the test returns true, a new path schema is included in $PS(k)$ (typically by performing a subsumption check). The output of the algorithm is $PS^+(k)$.

1. $PS(0)$ is initialized to the empty path schema starting at control state q_0
2. $k \longleftarrow 1$
3. `while` $PS^+(k-1) \neq \emptyset$ `do`
   (a) `for each` $\mathtt{P} \in PS^+(k-1)$ and $t \in \delta$ s.t. $first(t) = last(\mathtt{P})$ `do`
   { *look for path schemas ending with a transition* }
   i. `if` there exists no simple suffix S of $\mathtt{P} \cdot t$ s.t. $first(S) = last(S)$ `then`
   $PS(k) \stackrel{test}{\longleftarrow} \mathtt{P} \cdot t$ { *add path schema* $\mathtt{P} \cdot t$ *to level* k }
   ii. `for each` augmented suffix S of $\mathtt{P}$ s.t. $first(S \cdot t) = last(S \cdot t)$ `do`
   { *add cycle schema* $S \cdot t$ *to level* $\mathrm{len}(S \cdot t)$ }
   $CS(\mathrm{len}(S \cdot t)) \longleftarrow CS(\mathrm{len}(S \cdot t)) \cup \{S \cdot t\}$
   (b) { *add path schemas ending with a cycle* }
   `for each` $L \in CS^+(k)$ and prime cycle $\mathtt{l} \in \mathrm{Lan}^{\circlearrowleft}(L)$ s.t. $\mathrm{len}(\mathtt{l}) \leq k$ `do`
   `for each` $\mathtt{P} \in PS(k - \mathrm{len}(\mathtt{l}))$ s.t. $last(\mathtt{P}) = first(\mathtt{l})$ `do`
   $PS(k) \stackrel{test}{\longleftarrow} \mathtt{P} \cdot \mathtt{l}$ { *add* $\mathtt{P} \cdot \mathtt{l}$ *to level* k }
   (c) $k \longleftarrow k + 1$; `endwhile`
4. `return` $PS^+(k)$

Properties of the algorithm. The algorithm has several nice properties such as being parameterized by an eligibility test and it produces only good path schemas (see line 3(a)(i)). It is natural to wonder about the purpose of the eligibility test. Whenever a new path schema is built by the algorithm, we do not systematically insert it in the working set of path schemas (represented by $PS(k)$ or $PS^{+}(k)$). Indeed, it may happen that there is no run that respects it when starting by the initial control state q_0 and when satisfying the initial constraint on counter values. In that case, there is no point to include it in the working set of path schemas. When path schemas are generated with the purpose to abstract a potentially infinite set of runs, only consistent path schemas are inserted. Similarly, a new path schema may be subsumed by the working set of path schemas and if the property to be checked on runs can be safely pruned, such a new path schema can be discarded. The eligibility test allows to parameterize the algorithm by any kind of Boolean function to test whether a new path schema can be inserted or not. On the other hand, it might be useful to enumerate path schemas regardless the arithmetical constraints on counter values, which corresponds to consider the algorithm when the test always returns true. Consequently, the eligibility test provides a means to eliminate new path schemas depending on the purpose of the path schema generation.

Theorem 19 below states that the algorithm generates all the cycle schemas and path schemas when constraints on counter values are ignored. A nice way to ignore such values is to assume that the test returns always true, which amounts also to view the counter system as a standard labelled transition system.

Theorem 19 (Completeness). *Consider the algorithm in which the main test returns true. After completing the kth step of the main while loop:*

(†) *For every loop* $\mathtt{l}$ *of relative length at most k, there is a cycle schema $L \in CS^{+}(k)$ such that* $\mathtt{l} \in \mathrm{Lan}^{\circlearrowleft}(L)$.

(††) *For every path* $\mathtt{p}$ *starting at control state q_0 of ps-complexity at most k, there is a path schema* $\mathtt{P} \in PS^{+}(k)$ *such that* $\mathtt{p} \in \mathrm{Lan}(\mathtt{P})$.

With subsumption on counter values, a complete version of the algorithm can be obtained if cycles are generated independently of cycle schemas. At the time of writing this paper, we have designed such an algorithm, based on the one presented above. If cycle schemas are used to generate cycles during the course of the algorithm, then the enumeration procedure is known to be incomplete in the sense of case (††) in Theorem 19; that is, some path schemas may be missed at step k that are necessary to describe a path of ps-complexity k. However, this does not prevent us from using this algorithm for certain applications where completeness is less important, as useful path schemas might still be generated. Implementation and tests will be part of future work.

6 Conclusion

In this paper, we have recalled several classes of Presburger counter systems for which reachability sets are computable Presburger sets. Though this is a

desirable property to provide decision procedures on such machines, it is not sufficient if model-checking temporal properties are required; indeed, we may need to specify how intermediate configurations occur (see e.g. [DDS12]). For instance, the exact complexity of model-checking temporal properties for flat admissible Presburger counter systems is still open.

We have recalled several results from the literature and we emphasize that the generation of path schemas is a key problem for formal verification of Presburger counter systems. This is not really new (see e.g. [FO97, Boi99, Ler03, LS05]) but it is becoming an important issue, at least as important as the design of optimal decision procedures as far as worst-case complexity is concerned. The paper has been designed to put some light on this problem. However, an efficient generation of path schemas means that redundant path schemas should be eliminated as early as possible in the enumeration process. A comparison with the algorithm for acceleration technique in FAST [Ler03] or LASH [Boi99] will be in order.

We have introduced the notion of subsumption to take care of redundancy and again subsumption can be checked by testing the satisfiability of a quantified Presburger formula. This is a real challenge to deal with such quantified formulae in the framework of path schemas enumeration since most SMT solvers do not behave so nicely with quantified formulae, see e.g. [dMB08, BCD$^+$11]. Part of our future work is dedicated to design a path schema generation algorithm that invokes SMT solvers for quantified Presburger formulae.

Acknowledgements. The second author thanks the colleagues involved in fruitful collaborations about Presburger counter systems along the years; including members of the ANR Project REACHARD, R. Lazić (Warwick University), A. Dhar and A. Sangnier (LIAFA), and members of the ACSys group at NYU.

References

[AH11] Atig, M.F., Habermehl, P.: On Yen's path logic for Petri nets. IJFCS 22(4), 783–799 (2011)

[AJ96] Abdulla, P., Jonsson, B.: Verifying programs with unreliable channels. I & C 127(2), 91–101 (1996)

[BBH$^+$06] Bouajjani, A., Bozga, M., Habermehl, P., Iosif, R., Moro, P., Vojnar, T.: Programs with Lists Are Counter Automata. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 517–531. Springer, Heidelberg (2006)

[BCC$^+$03] Biere, A., Cimatti, A., Clarke, E.M., Strichman, O., Zhu, Y.: Bounded model checking. Advances in Computers 58, 118–149 (2003)

[BCD$^+$11] Barrett, C., Conway, C.L., Deters, M., Hadarean, L., Jovanović, D., King, T., Reynolds, A., Tinelli, C.: CVC4. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 171–177. Springer, Heidelberg (2011)

[BD11] Bersani, M.M., Demri, S.: The complexity of reversal-bounded model-checking. In: Tinelli, C., Sofronie-Stokkermans, V. (eds.) FroCoS 2011. LNCS (LNAI), vol. 6989, pp. 71–86. Springer, Heidelberg (2011)

[BDR03] Bruyère, V., Dall'Olio, E., Raskin, J.-F.: Durations, parametric model-checking in timed automata with Presburger arithmetic. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 687–698. Springer, Heidelberg (2003)

[BEH95] Bouajjani, A., Echahed, R., Habermehl, P.: On the verification problem of nonregular properties for nonregular processes. In: LICS 1995, pp. 123–133 (1995)

[Ber80] Berman, L.: The complexity of logical theories. TCS 11, 71–78 (1980)

[BFLS05] Bardin, S., Finkel, A., Leroux, J., Schnoebelen, P.: Flat acceleration in symbolic model checking. In: Peled, D.A., Tsay, Y.-K. (eds.) ATVA 2005. LNCS, vol. 3707, pp. 474–488. Springer, Heidelberg (2005)

[BFM+10] Bersani, M., Frigeri, A., Morzenti, A., Pradella, M., Rossi, M., San Pietro, P.: Bounded reachability for temporal logic over constraint systems. In: TIME 2010, pp. 43–50. IEEE (2010)

[BG06] Bozzelli, L., Gascon, R.: Branching-time temporal logic extended with qualitative Presburger constraints. In: Hermann, M., Voronkov, A. (eds.) LPAR 2006. LNCS (LNAI), vol. 4246, pp. 197–211. Springer, Heidelberg (2006)

[BGI09] Bozga, M., Gîrlea, C., Iosif, R.: Iterating octagons. In: Kowalewski, S., Philippou, A. (eds.) TACAS 2009. LNCS, vol. 5505, pp. 337–351. Springer, Heidelberg (2009)

[BIK09] Bozga, M., Iosif, R., Konečný, F.: Fast acceleration of ultimately periodic relations. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 227–242. Springer, Heidelberg (2010)

[BIL09] Bozga, M., Iosif, R., Lakhnech, Y.: Flat parametric counter automata. FI 91(2), 275–303 (2009)

[BJW01] Boigelot, B., Jodogne, S., Wolper, P.: On the use of weak automata for deciding linear arithmetic with integer and real variables. In: Goré, R.P., Leitsch, A., Nipkow, T. (eds.) IJCAR 2001. LNCS (LNAI), vol. 2083, pp. 611–625. Springer, Heidelberg (2001)

[BKR96] Biehl, M., Klarlund, N., Rauhe, T.: MONA: Decidable arithmetic in practice. In: Jonsson, B., Parrow, J. (eds.) FTRTFT 1996. LNCS, vol. 1135, pp. 459–462. Springer, Heidelberg (1996)

[BL10] Bojańczyk, M., Lasota, S.: An extension of data automata that captures XPath. In: LICS 2010, pp. 243–252. IEEE (2010)

[Boi99] Boigelot, B.: Symbolic methods for exploring infinite state spaces. PhD thesis, Université de Liège (1999)

[BP12] Bozzelli, L., Pinchinat, S.: Verification of gap-order constraint abstractions of counter systems. In: Kuncak, V., Rybalchenko, A. (eds.) VMCAI 2012. LNCS, vol. 7148, pp. 88–103. Springer, Heidelberg (2012)

[BS11] Blockelet, M., Schmitz, S.: Model checking coverability graphs of vector addition systems. In: Murlak, F., Sankowski, P. (eds.) MFCS 2011. LNCS, vol. 6907, pp. 108–119. Springer, Heidelberg (2011)

[BSST08] Barrett, C., Sebastiani, R., Seshia, S., Tinelli, C.: Satisfiability Modulo Theories. Frontiers in Artificial Intelligence and Applications, vol. 185, ch. 26, pp. 825–885. IOS Press (2008)

[BST12] Barrett, C., Stump, A., Tinelli, C.: The SMT-LIB standard: Version 2.0 (September 2012), `http://smtlib.cs.uiowa.edu/papers/smt-lib-reference-v2.0-r12.09.09.pdf`

[BW94] Boigelot, B., Wolper, P.: Verification with Periodic Sets. In: Dill, D.L. (ed.) CAV 1994. LNCS, vol. 818, pp. 55–67. Springer, Heidelberg (1994)

[BW98] Boigelot, B., Wolper, P.: Verifying systems with infinite but regular state spaces. In: Vardi, M.Y. (ed.) CAV 1998. LNCS, vol. 1427, pp. 88–97. Springer, Heidelberg (1998)

[CC00] Comon, H., Cortier, V.: Flatness is not a weakness. In: Clote, P.G., Schwichtenberg, H. (eds.) CSL 2000. LNCS, vol. 1862, pp. 262–276. Springer, Heidelberg (2000)

[CGP00] Clarke, E., Grumberg, O., Peled, D.: Model checking. MIT Press (2000)

[CJ98] Comon, H., Jurski, Y.: Multiple counter automata, safety analysis and Presburger Arithmetic. In: Vardi, M.Y. (ed.) CAV 1998. LNCS, vol. 1427, pp. 268–279. Springer, Heidelberg (1998)

[CKL13] Carapelle, C., Kartzow, A., Lohrey, M.: Satisfiability of CTL* with constraints. In: D'Argenio, P.R., Melgratti, H. (eds.) CONCUR 2013. LNCS, vol. 8052, pp. 455–469. Springer, Heidelberg (2013)

[Con12] Conchon, S.: SMT Techniques and their Applications: from Alt-Ergo to Cubicle. Habilitation à Diriger des Recherches, Université Paris-Sud (2012)

[Coo72] Cooper, D.: Theorem proving in arithmetic without multiplication. Machine Learning 7, 91–99 (1972)

[DD07] Demri, S., D'Souza, D.: An automata-theoretic approach to constraint LTL. I & C 205(3), 380–415 (2007)

[DDS12] Demri, S., Dhar, A.K., Sangnier, A.: Taming Past LTL and Flat Counter Systems. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS (LNAI), vol. 7364, pp. 179–193. Springer, Heidelberg (2012)

[DDS13] Demri, S., Dhar, A.K., Sangnier, A.: On the complexity of verifying regular properties on flat counter systems, In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part II. LNCS, vol. 7966, pp. 162–173. Springer, Heidelberg (2013)

[DFGvD10] Demri, S., Finkel, A., Goranko, V., van Drimmelen, G.: Model-checking CTL* over flat Presburger counter systems. JANCL 20(4), 313–344 (2010)

[DG08] Demri, S., Gascon, R.: Verification of qualitative Z constraints. TCS 409(1), 24–40 (2008)

[DG09] Demri, S., Gascon, R.: The effects of bounding syntactic resources on Presburger LTL. JLC 19(6), 1541–1575 (2009)

[DLN07] Demri, S., Lazić, R., Nowak, D.: On the freeze quantifier in constraint LTL: decidability and complexity. I & C 205(1), 2–24 (2007)

[DLS10] Demri, S., Lazić, R., Sangnier, A.: Model checking memoryful linear-time logics over one-counter automata. TCS 411(22-24), 2298–2316 (2010)

[dMB08] de Moura, L., Bjørner, N.S.: Z3: An Efficient SMT Solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008)

[Esp94] Esparza, J.: On the decidability of model checking for several μ-calculi and Petri nets. In: Tison, S. (ed.) CAAP 1994. LNCS, vol. 787, pp. 115–129. Springer, Heidelberg (1994)

[Fit02] Fitting, M.: Modal logic between propositional and first-order. JLC 12(6), 1017–1026 (2002)

[FL02] Finkel, A., Leroux, J.: How to compose Presburger-accelerations: Applications to broadcast protocols. In: Agrawal, M., Seth, A. (eds.) FSTTCS 2002. LNCS, vol. 2556, pp. 145–156. Springer, Heidelberg (2002)

[FO97] Fribourg, L., Olsén, H.: Proving safety properties of infinite state systems by compilation into Presburger arithmetic. In: Mazurkiewicz, A., Winkowski, J. (eds.) CONCUR 1997. LNCS, vol. 1243, pp. 213–227. Springer, Heidelberg (1997)

[FR74] Fischer, M., Rabin, M.: Super-exponential complexity of Presburger arithmetic. In: Complexity of Computation. SIAM-AMS proceedings, vol. 7, pp. 27–42. AMS (1974)

[FR79] Ferrante, J., Rackoff, C.: The Computational Complexity of Logical Theories. Lecture Notes in Mathematics, vol. 718. Springer (1979)

[Fri00] Fribourg, L.: Petri nets, flat languages and linear arithmetic. In: 9th Workshop on Functional and Logic Programming (WFLP), pp. 344–365 (2000)

[FS01] Finkel, A., Schnoebelen, P.: Well-structured transitions systems everywhere! TCS 256(1-2), 63–92 (2001)

[FS08] Finkel, A., Sangnier, A.: Reversal-bounded counter machines revisited. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 323–334. Springer, Heidelberg (2008)

[FS09] Figueira, D., Segoufin, L.: Future-looking logics on data words and trees. In: Královič, R., Niwiński, D. (eds.) MFCS 2009. LNCS, vol. 5734, pp. 331–343. Springer, Heidelberg (2009)

[GI81] Gurari, E., Ibarra, O.: The complexity of decision problems for finite-turn multicounter machines. In: Even, S., Kariv, O. (eds.) ICALP 1981. LNCS, vol. 115, pp. 495–505. Springer, Heidelberg (1981)

[GNRZ07] Ghilardi, S., Nicolini, E., Ranise, S., Zucchelli, D.: Combination methods for satisfiability and model-checking of infinite-state systems. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 362–378. Springer, Heidelberg (2007)

[Gor94] Goranko, V.: Temporal logic with reference pointers. In: Gabbay, D.M., Ohlbach, H.J. (eds.) ICTL 1994. LNCS (LNAI), vol. 827, pp. 133–148. Springer, Heidelberg (1994)

[Grä88] Grädel, E.: Subclasses of Presburger arithmetic and the polynomial-time hierarchy. TCS 56, 289–301 (1988)

[GS92] German, S., Sistla, P.: Reasoning about systems with many processes. JACM 39(3), 675–735 (1992)

[Hab97] Habermehl, P.: On the complexity of the linear-time mu-calculus for Petri nets. In: Azéma, P., Balbo, G. (eds.) ICATPN 1997. LNCS, vol. 1248, pp. 102–116. Springer, Heidelberg (1997)

[Hen90] Henzinger, T.: Half-order modal logic: how to prove real-time properties. In: PODC 1990, pp. 281–296. ACM Press (1990)

[HL11] Hague, M., Lin, A.W.: Model checking recursive programs with numeric data types. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 743–759. Springer, Heidelberg (2011)

[HP79] Hopcroft, J., Pansiot, J.J.: On the reachability problem for 5-dimensional vector addition systems. TCS 8, 135–159 (1979)

[HR87] Howell, R.R., Rosier, L.E.: An analysis of the nonemptiness problem for classes of reversal-bounded multicounter machines. JCSS 34(1), 55–74 (1987)

[HR89] Howell, R.R., Rosier, L.E.: Problems concerning fairness and temporal logic for conflict-free petri nets. TCS 64, 305–329 (1989)

[HRS76] Hunt, H., Rosenkrantz, D., Szymanski, T.: On the equivalence, containment, and covering problems for the regular and context-free languages. JCSS 12, 222–268 (1976)

[Iba78] Ibarra, O.: Reversal-bounded multicounter machines and their decision problems. JACM 25(1), 116–133 (1978)

[Jan90] Jančar, P.: Decidability of a temporal logic problem for Petri nets. TCS 74(1), 71–93 (1990)

[KM69] Karp, R.M., Miller, R.E.: Parallel program schemata. JCSS 3(2), 147–195 (1969)

[Kos82] Kosaraju, R.: Decidability of reachability in vector addition systems. In: STOC 1982, pp. 267–281 (1982)

[Ler03] 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)

[Ler09] Leroux, J.: The general vector addition system reachability problem by Presburger inductive invariants. In: LICS 2009, pp. 4–13. IEEE (2009)

[Ler12] Leroux, J.: Presburger counter machines. Habilitation à Diriger des Recherches, Université Bordeaux (2012)

[Ler13] Leroux, J.: Presburger Vector Addition Systems. In: LICS 2013, pp. 23–32. IEEE (2013)

[Lip76] Lipton, R.J.: The reachability problem requires exponential space. Technical Report 62, Department of Computer Science, Yale University (1976)

[LP05] Lisitsa, A., Potapov, I.: Temporal logic with predicate λ-abstraction. In: TIME 2005, pp. 147–155. IEEE (2005)

[LP09] Leroux, J., Point, G.: TaPAS: The Talence Presburger Arithmetic Suite. In: Kowalewski, S., Philippou, A. (eds.) TACAS 2009. LNCS, vol. 5505, pp. 182–185. Springer, Heidelberg (2009)

[LS05] Leroux, J., Sutre, G.: Flat counter automata almost everywhere! In: Peled, D.A., Tsay, Y.-K. (eds.) ATVA 2005. LNCS, vol. 3707, pp. 489–503. Springer, Heidelberg (2005)

[Lut04] Lutz, C.: NEXPTIME-complete description logics with concrete domains. ACM ToCL 5(4), 669–705 (2004)

[May84] Mayr, E.: An algorithm for the general Petri net reachability problem. SIAM Journal of Computing 13(3), 441–460 (1984)

[May03] Mayr, R.: Undecidable problems in unreliable computations. TCS 297(1-3), 337–354 (2003)

[Min61] Minsky, M.: Recursive unsolvability of Post's problems of 'tag' and other topics in theory of Turing machines. Annals of Mathematics 74(3), 437–455 (1961)

[Min67] Minsky, M.: Computation, Finite and Infinite Machines. Prentice Hall (1967)

[Mon10] Monniaux, D.: Quantifier elimination by lazy model enumeration. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 585–599. Springer, Heidelberg (2010)

[MP92] Manna, Z., Pnueli, A.: The Temporal Logic of Reactive and Concurrent Systems: Specification. Springer (1992)

[MP95] Manna, Z., Pnueli, A.: Temporal verification of reative systems: safety. Springer (1995)

[MS77] Mandel, A., Simon, I.: On finite semigroups of matrices. TCS 5(2), 101–111 (1977)

[Opp78] Oppen, D.: A $2^{2^{2pn}}$ upper bound on the complexity of Presburger arithmetic. JCSS 16(3), 323–332 (1978)

[OW05] Ouaknine, J., Worrell, J.: On the Decidability of Metric Temporal Logic. In: LICS 2005, pp. 188–197. IEEE (2005)

[Pnu77] Pnueli, A.: The temporal logic of programs. In: FOCS 1977, pp. 46–57. IEEE (1977)

[Pre29] Presburger, M.: Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In: Comptes Rendus du Premier Congrès de Mathématiciens des Pays Slaves, Warszawa, pp. 92–101 (1929)

[Pug92] Pugh, W.: A practical algorithm for exact array dependence analysis. Communications of the ACM 35(8), 102–114 (1992)

[QR05] Qadeer, S., Rehof, J.: Context-bounded model checking of concurrent software. In: Halbwachs, N., Zuck, L.D. (eds.) TACAS 2005. LNCS, vol. 3440, pp. 93–107. Springer, Heidelberg (2005)

[Reu90] Reutenauer, C.: The mathematics of Petri nets. Masson and Prentice (1990)

[RL78] Reddy, C., Loveland, W.: Presburger arithmetic with bounded quantifier alternation. In: STOC 1978, pp. 320–325. ACM Press (1978)

[Sch02] Schnoebelen, P.: Verifying lossy channel systems has nonprimitive recursive complexity. IPL 83, 251–261 (2002)

[Sch10] Schnoebelen, P.: Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 616–628. Springer, Heidelberg (2010)

[Sho79] Shostak, R.: A practical decision procedure for arithmetic with function symbols. JACM 26(2), 351–360 (1979)

[SJ80] Suzuki, N., Jefferson, D.: Verification Decidability of Presburger Array Programs. JACM 27(1), 191–205 (1980)

[ST11] Segoufin, L., Torunczyk, S.: Automata based verification over linearly ordered data domains. In: STACS 2011, pp. 81–92 (2011)