Equivalence Between Model-Checking Flat Counter Systems and Presburger Arithmetic

Stéphane Demri, Amit K Kumar Dhar, Arnaud Sangnier

Equivalence Between Model-Checking Flat Counter Systems and Presburger Arithmetic*

Stéphane Demri[2], Amit Kumar Dhar[1], and Arnaud Sangnier[1]

[1] LIAFA, Univ Paris Diderot, Sorbonne Paris Cité, CNRS, France
[2] NYU – CNRS

Abstract. We show that model-checking flat counter systems over CTL* (with arithmetical constraints on counter values) has the same complexity as the satisfiability problem for Presburger arithmetic. The lower bound already holds with the temporal operator EF only, no arithmetical constraints in the logical language and with guards on transitions made of simple linear constraints. This complements our understanding of model-checking flat counter systems with linear-time temporal logics, such as LTL for which the problem is already known to be (only) NP-complete with guards restricted to the linear fragment.

1 Introduction

Branching-time temporal logics for counter systems. At first glance, model-checking counter systems with temporal logics seems a hopeless enterprise since control-state reachability problem for Minsky machines is already known to be undecidable [23]. Fortunately, many subclasses of counter systems admit a decidable reachability problem and more importantly, sometime the reachability sets are even definable in Presburger arithmetic (PA) [24]. That is why, model-checking problems with temporal logics for one-counter automata [15,16], Petri nets [17], reversal-bounded counter systems [2], flat counter systems [14] have been considered. The previous list is certainly not exhaustive and a general question is how to take advantage of the decidability of the reachability problem to conclude the decidability of model-checking problems with temporal logics. This can lead to endless studies, since the variety of subclasses of counter systems and temporal logics is extremely rich. By way of example, reachability sets of flat counter systems are known to be definable in (PA), see e.g. [3,6,14,4], but it is unclear how this can be extended to model-checking problems with temporal logics, which indeed, is done in [10] for flat counter systems. A complexity characterization for model-checking linear-time properties is provided in [9]. In the present paper, we study flat counter systems and branching-time temporal logics, more specifically with a variant of CTL* [12], already known to be difficult to mechanize in the propositional case with labelled transition systems.

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

Our motivations We have seen that reachability problems and the verification of linear-time properties for flat counter systems are nowadays well-studied (see e.g. [9,4]) and in this paper we wish to understand the computational complexity for branching-time temporal logics such as CTL or CTL* (see e.g. [12]). Branching-time extensions often lead to undecidability, see e.g. the case with Petri nets for which CTL is undecidable (with propositional variables only) whereas the reachability problem and model-checking for several LTL variants are known to be decidable [17]. For flat counter systems, we are on the safe side since decidability of model-checking CTL* formulae is established in [10] but no lower bound is provided in [10] and the translation into (PA) gives an exponential size formula, which is rather unsatisfactory. Our main motivation is therefore to understand the complexity of model-checking flat counter systems with branching-time logics so that optimal algorithms for model-checking can be eventually designed.

Our contribution. We show that the model-checking problem for flat counter systems over a version of CTL* with arithmetical constraints on counter values is equivalent to satisfiability for (PA), modulo logarithmic-space reductions.

- For the complexity lower bound, we show that the satisfiability problem for (PA) can be reduced to the model-checking problem but there is no need for arithmetical constraints and for temporal operators other than `EF`.
- For the complexity upper bound, we reduce the model-checking problem to satisfiability problem in (PA) by using the fact that runs in flat counter systems can be encoded by tuples of natural numbers and then the semantics for CTL* can be internalized in (PA). This very fact has been already observed in [10] but herein, we provide a logarithmic-space reduction which makes a substantial difference with [10]. Indeed, we are also able to quantify over path schemas (symbolic representation of potential infinite sets of runs), but concisely. This witnesses once more, that verification problems can be encoded efficiently to (PA), see e.g. [5].
- As a consequence, we are able to get the equivalence with (PA) to known branching-time temporal logics stronger than CTL$_{\texttt{EF}}$ (such as CTL) and our proof technique can be applied to extensions with past-time operators with a minimal amount of change.

As far as proofs are concerned, for the lower bound, we take advantage of the observation that a quantification in (PA) over a variable `z` can be simulated by a loop that increments a corresponding counter and there is a correspondence between first-order quantifier $\exists$ [resp. $\forall$] and temporal connective `EF` [resp. `AG`]. For the upper bound, quantification over path schemas is done, directly followed by a quantification over the number of times loops are visited. However, we provide a new way to encode runs in flat counter systems, which is rewarding complexity-wise. Not only we provide a much better complexity characterization than [10] but also our reduction into (PA) is much simpler, and therefore this leaves some hope to use then some solvers for (PA), see e.g. [7,20].
Due to lack of space, omitted proofs can be found in [11].

2 Branching-time temporal logics on flat counter systems

Presburger arithmetic. (PA), i.e. the first-order theory of natural numbers with addition, is introduced by M. Presburger who has shown decidability by quantifier elimination. Let VAR $= \{\mathtt{z}_1, \mathtt{z}_2, \mathtt{z}_3, \ldots\}$ be a countably infinite set of *variables*. *Terms* are expressions of the form $a_1\mathtt{z}_1 + \cdots + a_n\mathtt{z}_n + k$ where a_1, $\ldots$, a_n, k are in $\mathbb{N}$. A *valuation* f is a map VAR $\to \mathbb{N}$ and it can be extended to the set of all terms as follows: $f(k) = k$, $f(a\mathtt{z}) = a \times f(\mathtt{z})$ and $f(t + t') = f(t) + f(t')$ for all terms t and t'. *Formulae* are defined by the grammar $\phi ::= \ t \leq t' \mid \neg\phi \mid \phi \wedge \phi \mid \exists\ \mathtt{z}\ \phi$ where t and t' are terms and $\mathtt{z} \in$ VAR. A formula ϕ is in the *linear fragment* $\stackrel{\text{def}}{\Leftrightarrow}$ ϕ is a Boolean combination of atomic formulae of the form $t \leq t'$. The semantics for formulae in (PA) is defined with the satisfaction relation $\models$: for instance $f \models t \leq t' \stackrel{\text{def}}{\Leftrightarrow} f(t) \leq f(t')$ and $f \models \exists\ \mathtt{z}\ \phi \stackrel{\text{def}}{\Leftrightarrow}$ there is $n \in \mathbb{N}$ such that $f[\mathtt{z} \mapsto n] \models \phi$. Any formula $\phi(\mathtt{z}_1, \ldots, \mathtt{z}_n)$ whose free variables are among $\mathtt{z}_1$, $\ldots$, $\mathtt{z}_n$, with $n \geq 1$, defines a set of n-tuples $[\![\phi(\mathtt{z}_1, \ldots, \mathtt{z}_n)]\!] \stackrel{\text{def}}{=} \{\langle f(\mathtt{z}_1), \ldots, f(\mathtt{z}_n)\rangle \in \mathbb{N}^n : f \models \phi\}$, in that case for a vector $\mathbf{v} \in \mathbb{N}^n$, we will also write $\mathbf{v} \models \phi$ for $\mathbf{v} \in [\![\phi(\mathtt{z}_1, \ldots, \mathtt{z}_n)]\!]$. For a given PA formula ϕ, the set of free variables of ϕ is denoted by $\mathtt{free}(\phi)$. The *satisfiability problem* for (PA) is a decision problem that takes as input a formula ϕ and asks whether there is a valuation f such that $f \models \phi$. If such a valuation exists, we say that ϕ is *satisfiable*. Decidability of Presburger arithmetic has been shown in [24]. An exact complexity characterization is provided in [1].

Counter systems. Let $\mathtt{C} = \{\mathtt{x}_1, \mathtt{x}_2, \ldots\}$ be a countably infinite set of *counters* with the finite subset $\{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$ denoted as $\mathtt{C}_n$ and AT $= \{\mathtt{p}_1, \mathtt{p}_2, \ldots\}$ be a countably infinite set of atomic propositional variables. A *counter system* is a tuple $\langle Q, \mathtt{C}_n, \Delta, \ell\rangle$ where Q is a finite set of *control states*, $\ell : Q \to 2^{\text{AT}}$ is a *labelling function*, $\Delta \subseteq Q \times G_n \times \mathbb{Z}^n \times Q$ is a finite set of edges labelled by *guards* and *updates* of the counter values *(transitions)* where G_n is a finite set of Presburger formulae ϕ with $\mathtt{free}(\phi) \subseteq \{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$. Guards are quite general and we basically only need them in the linear fragment. However, since we provide a translation into (PA), we can be a bit more general, as in Presburger counter machines [10,19].

For each transition $\delta = \langle q, \mathtt{g}, \mathbf{u}, q'\rangle$ in Δ, we use the following notations: $\mathit{source}(\delta) = q$, $\mathit{target}(\delta) = q'$, $\mathit{guard}(\delta) = \mathtt{g}$ and $\mathit{update}(\delta) = \mathbf{u}$. As usual, to a counter system $S = \langle Q, \mathtt{C}_n, \Delta, \ell\rangle$, we associate a labelled transition system $\mathfrak{T}(S) = \langle \mathcal{C}, \to\rangle$ where $\mathcal{C} = Q \times \mathbb{N}^n$ is the set of *configurations* and $\to \subseteq \mathcal{C} \times \Delta \times \mathcal{C}$ is the *transition relation* defined by: $\langle\langle q, \mathbf{v}\rangle, \delta, \langle q', \mathbf{v}'\rangle\rangle \in \to$ (also written $\langle q, \mathbf{v}\rangle \xrightarrow{\delta} \langle q', \mathbf{v}'\rangle$) iff $q = \mathit{source}(\delta)$, $q' = \mathit{target}(\delta)$, $\mathbf{v} \models \mathit{guard}(\delta)$ and $\mathbf{v}' = \mathbf{v} + \mathit{update}(\delta)$. We write $c \to c'$ iff there exists some edge δ, such that $c \xrightarrow{\delta} c'$.

Given $c_0 \in Q \times \mathbb{N}^n$, a *run* ρ starting from c_0 in S is a (possibly infinite) path in the associated transition system $\mathfrak{T}(S)$ denoted as $\rho := c_0 \xrightarrow{\delta_0} \cdots \xrightarrow{\delta_{m-1}} c_m \xrightarrow{\delta_m} \cdots$, where $c_i \in Q \times \mathbb{N}^n$ and $\delta_i \in \Delta$, for all $i \in \mathbb{N}$.

Let $trans(\rho)$ be the ω-word $\delta_0\delta_1\ldots$ of the sequence of transitions appearing in the run ρ. For every $i \geq 0$, we define $\rho(i) = c_i$ and $\rho_{\leq i} = c_0 \xrightarrow{\delta_0} c_1 \cdots \xrightarrow{\delta_{i-1}} c_i$. Also, we say $c \rightarrow^* c'$ iff there exist a run ρ and $i \in \mathbb{N}$ such that $\rho(0) = c$ and $\rho(i) = c'$. Note that a run in a counter system S is either finite or infinite. A run ρ is *maximal* iff either it is infinite, or it is finite and the last configuration is a deadlock (i.e. with no successor configurations). For a finite maximal run $\rho := c_0 \xrightarrow{\delta_0} \cdots \xrightarrow{\delta_{m-1}} c_m \xrightarrow{\delta_m} c_{m+1}$, we write $|\rho| = m$, otherwise for an infinite maximal run ρ, $|\rho| = \omega$.

A counter system is *flat* if every node in the underlying graph belongs to at most one simple cycle (a cycle being simple if no edge is repeated twice in it) [6,22]. In a flat counter system, simple cycles can be organized as a DAG where two simple cycles are in the relation whenever there is path between a node of the first cycle and a node of the second cycle. We denote by $\texttt{FlatCS}$ the class of flat counter systems.

Logical specifications. The formulae for CTL* are defined as follows: $\phi ::= \mathtt{p} \mid \psi(\mathtt{x}_1, \ldots, \mathtt{x}_n) \mid \phi \wedge \phi \mid \neg\phi \mid \mathtt{X}\phi \mid \phi\mathtt{U}\phi \mid \mathtt{E}\phi$ where $\mathtt{p} \in$ AT and $\psi(\mathtt{x}_1, \ldots, \mathtt{x}_n)$ is a Presburger formula. We write CTL$_{\mathtt{EF}}$ to denote the fragment of CTL* in which the only (unary) temporal operator is $\mathtt{EF}$ ($\mathtt{EF}\phi \stackrel{\text{def}}{=} \mathtt{E}\ (\top\ \mathtt{U}\ \phi)$ and $\top \stackrel{\text{def}}{=} (\mathtt{x}_1 = \mathtt{x}_1)$). Our version of CTL* is defined as the standard version, see e.g. [12], except that the Kripke structures are replaced by transition systems from counter systems and at the atomic level, arithmetical constraints are allowed. Let $S = \langle Q, \mathtt{C}_n, \Delta, \ell\rangle$ be a counter system with transition system $\mathfrak{T}(S) = \langle \mathcal{C}, \rightarrow\rangle$. The satisfaction relation $\models$ is defined as follows (CTL* formula ϕ, maximal run ρ in $\mathfrak{T}(S)$, position $i < |\rho|$):

$$
\begin{array}{lll}
\rho, i \models \mathtt{p} & \stackrel{\text{def}}{\Leftrightarrow} & \mathtt{p} \in \ell(q), \text{ where } \rho(i) = \langle q, \mathbf{v}\rangle \\
\rho, i \models \psi(\mathtt{x}_1, \ldots, \mathtt{x}_n) & \stackrel{\text{def}}{\Leftrightarrow} & \mathbf{v} \models \psi(\mathtt{x}_1, \ldots, \mathtt{x}_n), \text{ where } \rho(i) = \langle q, \mathbf{v}\rangle \\
\rho, i \models \mathtt{X}\psi & \stackrel{\text{def}}{\Leftrightarrow} & \rho, i+1 \models \psi \text{ and } i+1 < |\rho| \\
\rho, i \models \psi_1 \mathtt{U} \psi_2 & \stackrel{\text{def}}{\Leftrightarrow} & \rho, j \models \psi_2 \text{ for some } i \leq j \\
 & & \quad \text{such that } j < |\rho| \text{ and } \rho, k \models \psi_1 \text{ for all } i \leq k < j \\
\rho, i \models \mathtt{E}\phi & \stackrel{\text{def}}{\Leftrightarrow} & \text{there is a maximal run } \rho' \text{ s.t. } \rho'(0) = \rho(i) \text{ and } \rho', 0 \models \phi
\end{array}
$$

Given a CTL* formula ϕ, a counter system S and a configuration c from S, we say that $S, c \models \phi$ iff there exists a maximal run ρ in the configuration graph $\mathfrak{T}(S)$ with $\rho(0) = c$ such that $\rho, 0 \models \phi$ (note that there is an overload for $\models$ in $S, c \models \phi$). A flat counter system S is called *non-blocking* if every maximal run ρ in S is infinite. Otherwise it is called a *blocking* flat counter system.

Lemma 1. *Let* L *be either* CTL* *or* CTL$_{\mathtt{EF}}$. *Given a flat counter system S, a configuration c and a formula ϕ in* L, *there exist a non-blocking flat counter system S', a configuration c' and a formula ϕ' in* L *such that $S, c \models \phi$ iff $S', c' \models \phi'$. Such a reduction can be performed in logarithmic space.*

It is easy to see that we can add some dummy state to a blocking flat counter system to obtain a non-blocking one preserving the satisfiability of formulae.

The formula is also transformed by using a standard relativization method (a new propositional variable is introduced that holds true only on configurations that were not reachable in the original counter system). Due to Lemma 1, henceforth we consider only non-blocking flat counter systems. Since the reachability relation is definable in (PA) for flat counter systems [10], it is even possible to decide whether all maximal runs from a given configuration are infinite.

The model-checking problem for flat counter systems over CTL* is defined as follows (let us call it MC(CTL*, `FlatCS`)): given a flat counter system S, a configuration c and a formula ϕ in CTL*, determine whether $S, c \models \phi$. We know that MC(CTL*, `FlatCS`) is decidable [10] but its exact complexity is not fully characterized (actually, this is the purpose of the present paper). The restriction to LTL formulae is known to be NP-complete [8] when guards are restricted to the linear fragment. In Section 3, we show that the satisfiability problem for (PA) can be reduced to MC(CTL*, `FlatCS`) restricted to CTL$_{\texttt{EF}}$ without arithmetical constraints and to flat counter systems such that the guards are restricted to simple linear constraints. By contrast, model-checking flat finite Kripke structures over CTL* is Δ_2^P-complete [13,18].

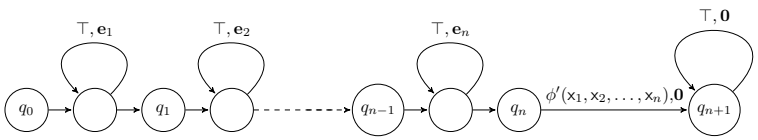
3 Reducing (PA) to a subproblem of MC(CTL*, `FlatCS`)

In a flat counter system with n counters, the guards on transitions are Presburger formulae with free variables in $\{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$. That is why, it is easy to show that the satisfiability problem for (PA) can be reduced (in logarithmic space) to MC(CTL*, `FlatCS`). Clearly, this is not interesting and the generality of the guards in the paper is only considered because, when establishing the complexity upper bound, we can be quite liberal with the set of guards. However, a more reasonable set of guards is the linear fragment (i.e., without any first-order quantification). Below, we show that a very restricted fragment of MC(CTL*, `FlatCS`), simply called MC$^-$(CTL*, `FlatCS`), is already as hard as the satisfiability problem for (PA) and our reduction is based on a simple and nice correspondence between quantifiers in (PA) and the temporal operators `EF` and `AG` in CTL*. First, let us define MC$^-$(CTL*, `FlatCS`) as the subproblem of MC(CTL*, `FlatCS`) with the following restrictions: (a) atomic formulae are restricted to propositional variables and the only temporal connective is `EF` (and its dual `AG`, by closure under negation) and (b) the guards on the transitions are linear constraints $t \leq t'$ or their negations.

Theorem 2. *There is a logarithmic-space reduction from the satisfiability problem for (PA) to MC$^-$(*CTL**,* `FlatCS`*).*

Proof. (sketch) Let ϕ be a formula in (PA). Without any loss of generality, we can assume that ϕ has the form $\mathcal{Q}_1\ \mathtt{z}_1\ \mathcal{Q}_2\ \mathtt{z}_2\ \cdots\ \mathcal{Q}_n\ \mathtt{z}_n\ \phi'(\mathtt{z}_1, \mathtt{z}_2, \ldots, \mathtt{z}_n)$ with $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_n \in \{\exists, \forall\}$ and ϕ' is a quantifier-free formula. Note that given any formula in (PA), we can reduce it to an equisatisfiable formula of that form in logarithmic space (which is then fine for our main result since logarithmic-space reductions are closed under composition). This is essentially based on the construction of formulae in prenex normal form in first-order logic.

Let us consider S_ϕ defined below where $\mathbf{e}_i \in \mathbb{N}^n$ is the ith unit vector.



Observe that $\phi'(\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n)$ may contain Boolean connectives but we explain below how to get rid of them in S_ϕ. Below, we define ψ in $\mathrm{CTL}_{\mathtt{EF}}$ whose atomic formulae are among $q_1, \ldots, q_{n+1}$ (also abusively understood as control states) such that (†) $S_\phi, \langle q_0, \mathbf{0}\rangle \models \psi$ iff ϕ is satisfiable in (PA). Intuitively, the possible value associated to each variable $\mathtt{z}_i$ from ϕ is taken care of by the ith loop (that can only increment the ith counter). This is not enough, and additionally, the quantifications from ϕ are simulated in the formula ψ by using $\mathtt{EF}$ or $\mathtt{AG}$, depending whether the first-order quantification is either existential or universal. Let us define below the formulae ψ_i with $i \in [1, n+1]$ so that $\psi \stackrel{\text{def}}{=} \psi_1$, $\psi_{n+1} \stackrel{\text{def}}{=} \mathtt{EF}\ q_{n+1}$, and for every $i \in [1, n]$, if $\mathcal{Q}_i = \exists$ then $\psi_i \stackrel{\text{def}}{=} \mathtt{EF}(q_i \wedge \psi_{i+1})$, otherwise $\psi_i \stackrel{\text{def}}{=} \mathtt{AG}(q_i \Rightarrow \psi_{i+1})$. In order to establish (†) it is sufficient to show the property (††) below. Given a valuation $f : \mathrm{VAR} \to \mathbb{N}$, we write $\mathbf{v}_f \in \mathbb{N}^n$ to denote the vector such that $\mathbf{v}_f[i] \stackrel{\text{def}}{=} f(\mathtt{z}_i)$ for every $i \in [1, n]$. One can show that (††) for all f, we have $f \models \phi'(\mathtt{z}_1, \mathtt{z}_2, \ldots, \mathtt{z}_n)$ iff $\langle q_n, \mathbf{v}_f\rangle \models \psi_{n+1}$ and for all $i \in [1, n]$ and for all valuations f such that $f(\mathtt{z}_i) = \cdots = f(\mathtt{z}_n) = 0$, we have $f \models \mathcal{Q}_i\ \mathtt{z}_i\ \cdots\ \mathcal{Q}_n\ \mathtt{z}_n\ \phi'(\mathtt{z}_1, \mathtt{z}_2, \ldots, \mathtt{z}_n)$ iff $\langle q_{i-1}, \mathbf{v}_f\rangle \models \psi_i$. We get the property (†) by applying (††) with $i = 1$.

To eliminate the Boolean connectives in the guard of the transition between q_n and q_{n+1}, we follow two simple rules, while preserving flatness (easy to check since that transition does not belong to a loop). W.l.o.g., we can assume that negations are only in front of linear constraints. A transition $q \xrightarrow{\psi_1 \wedge \psi_2, \mathbf{0}} q'$ is replaced by $q \xrightarrow{\psi_1, \mathbf{0}} q'' \xrightarrow{\psi_2, \mathbf{0}} q'$ where q'' is new. Similarly, a transition $q \xrightarrow{\psi_1 \vee \psi_2, \mathbf{0}} q'$ is replaced by $q \xrightarrow{\psi_1, \mathbf{0}} q'$ and $q \xrightarrow{\psi_2, \mathbf{0}} q'$, assuming that q does not belong to a loop. It is easy to show that S_ϕ can be transformed into a flat counter system S'_ϕ by applying the two rules above as much as possible so that eventually, S'_ϕ is a proper counter system for $\mathrm{MC}^-(\mathrm{CTL}^*, \mathtt{FlatCS})$. □

4 From MC(CTL*, FlatCS) to (PA)

In this section, we present a logarithmic-space reduction from $\mathrm{MC}(\mathrm{CTL}^*, \mathtt{FlatCS})$ to the satisfiability problem for (PA). In [10], a reduction is already presented to get decidability of $\mathrm{MC}(\mathrm{CTL}^*, \mathtt{FlatCS})$. Unfortunately, it requires exponential space and is quite difficult to parse. Following a similar idea, we propose here a simpler reduction that has the great advantage to be optimal complexity-wise. The idea of this reduction is based on the two following points:

1. encoding the runs in flat counter systems by tuples of natural numbers thanks to a symbolic representation for potential infinite sets of runs, see path schemas in [8],

2. internalizing CTL* semantics into (PA) by using the encoding of runs.

Below, we consider a fixed flat counter system $S = \langle Q, \mathtt{C}_n, \Delta, \ell \rangle$ and w.l.o.g, $Q = \{1, \ldots, \alpha\}$ for some $\alpha \geq 1$ and $\Delta = \{\delta_1, \ldots, \delta_\beta\}$. Since $Q \subseteq \mathbb{N}$, configurations of S are vectors in $\mathbb{N}^{n+1}$ where the first component represents the control state.

4.1 Minimal path schemas

In [8], following an idea from [21], *minimal path schemas* are introduced as a mean to symbolically represent all runs in flat counter systems. Path schemas can be defined as finite sequences made of transitions or simple loops (conditions apply). Formal definition is recalled below. A *simple loop* l of S is a non-empty finite sequence of transitions $\delta_1, \ldots, \delta_m$ such that $\mathit{source}(\delta_1) = \mathit{target}(\delta_m)$, $\mathit{source}(\delta_j) = \mathit{target}(\delta_{j+1})$ for all $j \in [1, m-1]$, and, for all $j, k \in [1, m]$, if $j \neq k$ then $\delta_j \neq \delta_k$. The length of l, written $\mathit{length}(l)$, is the value m and we denote by $\mathit{source}(l) = \mathit{target}(l)$ the control state $\mathit{source}(\delta_1)$. The number of simple loops is necessarily finite and we assume that the set of loops of S is $L = \{l_1, l_2, \ldots, l_\gamma\}$. Since S is flat, we have $\gamma \leq \alpha$. A *minimal path schema* P is a non-empty sequence $u_1, \ldots, u_N$ s.t. each $u_i \in \Delta \cup L$ and the following conditions are verified.

1. u_N is a loop,
2. $i \neq j$ implies $u_i \neq u_j$,
3. for all $i \in [1, N-1]$, we have $\mathit{target}(u_i) = \mathit{source}(u_{i+1})$.

The second condition guarantees minimality whereas the third condition ensures that P respects the control graph of S. The *size* of P, denoted by $\mathit{size}(P)$, is equal to N. For all $j \in [1, N]$, we write $P[j]$ to denote u_j. Here is an obvious result.

Lemma 3. *The size of a minimal path schema is bounded by* $\beta + \gamma \leq \beta + \alpha$.

In order to obtain concrete paths from a path schema P, we augment P with a vector specifying how many times each internal loop is visited. By definition, a loop in P is *internal* if it is not the last one. An *iterated path schema* is a pair $\langle P, \mathbf{m} \rangle$ where P is a minimal path schema and $\mathbf{m} \in \mathbb{N}^{\mathit{size}(P)}$ such that $\mathbf{m}[1] = \mathit{size}(P)$ and for all $i \in [1, \mathit{size}(P) - 1]$, $\mathbf{m}[i+1] > 0$ and if $P[i] \in \Delta$, then $\mathbf{m}[i+1] = 1$. From $\langle P, \mathbf{m} \rangle$, we define the ω-word

$$\mathit{trans}(P, \mathbf{m}) \stackrel{\text{def}}{=} P[1]^{\mathbf{m}[2]} \ldots P[j]^{\mathbf{m}[j+1]} \ldots P[\mathit{size}(P) - 1]^{\mathbf{m}[\mathit{size}(P)]} P[\mathit{size}(P)]^\omega$$

Lemma 4 below states that iterated path schemas encode all runs from flat counter systems by noting that infinite runs necessarily end by a simple loop (repeated infinitely) and the visit of loops is strictly ordered.

Lemma 4. *[8] Given an infinite run ρ in a flat counter system S , there exists an iterated path schema $\langle P, \boldsymbol{m} \rangle$ such that $\mathit{trans}(\rho) = \mathit{trans}(P, \boldsymbol{m})$.*

Encoding iterated path schemas. Thanks to Lemma 3, we show that it is possible to encode path schemas by vectors in $\mathbb{N}^K$ with $K = 1 + \beta + \gamma$. Intuitively, we encode a path schema P by two vectors $\mathbf{v}_p$ and $\mathbf{v}_t$ in $\mathbb{N}^K$ where the first element of each vector is equal to $size(P)$ and for all $i \in [2, size(P)+1]$, we have $\mathbf{v}_t[i] = 1$ if $P[i]$ is a loop and $\mathbf{v}_t[i] = 0$ otherwise. So, $\mathbf{v}_t$ encodes the *type* of each element (transition vs. loop) in the sequence defining P. Similarly, $\mathbf{v}_p[i]$ represents the number of the associated transition or loop; for instance, $\mathbf{v}_p[i] = 2$ and $\mathbf{v}_t[i] = 1$ encodes that $P[i]$ is the second loop, say l_2. Furthermore, we encode the vector $\mathbf{m}$ by a vector $\mathbf{v}_{it} \in \mathbb{N}^K$. Let us formalize this. First, we define the function $\tau : (\{0\} \times [1, \beta]) \cup (\{1\} \times [1, \gamma])) \to \Delta \cup L$ such that $\tau(0, i) \stackrel{\text{def}}{=} \delta_i$ and $\tau(1, i) \stackrel{\text{def}}{=} l_i$. Now, we provide a set of conditions $\mathfrak{C}$ on the vectors $\mathbf{v}_t, \mathbf{v}_p, \mathbf{v}_{it} \in \mathbb{N}^K$ which have to be respected so that, we can build from them an iterated path schema.

- $\mathfrak{C}.1$ $\mathbf{v}_p[1] = \mathbf{v}_t[1] = \mathbf{v}_{it}[1]$ with $\mathbf{v}_p[1] \in [1, K-1]$; for all $i \in [\mathbf{v}_{it}[1] + 2, K]$, $\mathbf{v}_p[i] = \mathbf{v}_t[i] = \mathbf{v}_{it}[i] = 0$,
- $\mathfrak{C}.2$ $\mathbf{v}_t[i] \in \{0, 1\}$ for all $i \in [2, K]$,
- $\mathfrak{C}.3$ if $\mathbf{v}_t[i] = 0$ then $\mathbf{v}_p[i] \in [1, \beta]$, for all $i \in [2, \mathbf{v}_p[1] + 1]$,
- $\mathfrak{C}.4$ if $\mathbf{v}_t[i] = 1$ then $\mathbf{v}_p[i] \in [1, \gamma]$, for all $i \in [2, \mathbf{v}_p[1] + 1]$,
- $\mathfrak{C}.5$ $\mathbf{v}_t[\mathbf{v}_p[1] + 1] = 1$,
- $\mathfrak{C}.6$ there are no $i, j \in [2, \mathbf{v}_p[1]+1]$ such that $i \neq j$, $\mathbf{v}_t[i] = \mathbf{v}_t[j]$ and $\mathbf{v}_p[i] = \mathbf{v}_p[j]$,
- $\mathfrak{C}.7$ $target(\tau(\mathbf{v}_t[i], \mathbf{v}_p[i])) = source(\tau(\mathbf{v}_t[i+1], \mathbf{v}_p[i+1]))$ for all $i \in [2, \mathbf{v}_p[1]]$,
- $\mathfrak{C}.8$ for all $i \in [2, \mathbf{v}_p[1]]$, $\mathbf{v}_{it}[i] > 0$ and if $\mathbf{v}_t[i] = 0$ then $\mathbf{v}_{it}[i] = 1$.

The first four conditions ensure that the vectorial representation is coherent. The three next conditions guarantee that the encoding respects the structure of a minimal path schema, i.e. that the last element is a loop ($\mathfrak{C}.5$), that there are no two identical transitions or loops in the schema ($\mathfrak{C}.6$) and that the succession of elements effectively represents a path in the counter system ($\mathfrak{C}.7$). The last condition ensures that $\mathbf{v}_{it}$ matches the definition of the vector in an iterated path schema. It follows that given vectors $\mathbf{v}_p$, $\mathbf{v}_t$ and $\mathbf{v}_{it}$ in $\mathbb{N}^K$ that satisfy all the conditions ($\mathfrak{C}.i$), we can build a minimal path schema $P_{\mathbf{v}_t, \mathbf{v}_p}$ equal to

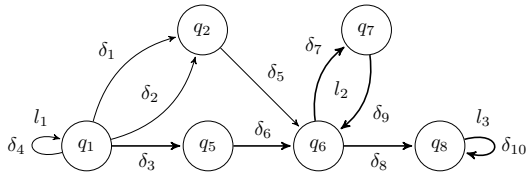
$$\tau(\mathbf{v}_t[2], \mathbf{v}_p[2]) \cdots \tau(\mathbf{v}_t[\mathbf{v}_p[1] + 1], \mathbf{v}_p[\mathbf{v}_p[1] + 1])$$

From the vector $\mathbf{v}_{it}$, we can define the vector $\mathbf{m}_{\mathbf{v}_{it}} \in \mathbb{N}^{\mathbf{v}_{it}[1]}$ such that for all $i \in [1, \mathbf{v}_{it}[1]]$, $\mathbf{m}_{\mathbf{v}_{it}}[i] \stackrel{\text{def}}{=} \mathbf{v}_{it}[i]$. We will see that there exists a Presburger formula $Schema(\mathsf{Z}_t, \mathsf{Z}_p, \mathsf{Z}_{it})$ over the sets of variables $\mathsf{Z}_p = \{\mathsf{z}_p^1, \ldots, \mathsf{z}_p^K\}$, $\mathsf{Z}_t = \{\mathsf{z}_t^1, \ldots, \mathsf{z}_t^K\}$ and $\mathsf{Z}_{it} = \{\mathsf{z}_{it}^1, \ldots, \mathsf{z}_{it}^K\}$ to express the conditions $(\mathfrak{C}.i)_{i \in [1,8]}$.

Lemma 5.

1. *Let P be a finite non-empty sequence of length $N \leq \beta + \gamma$ over the alphabet $\Delta \cup L$ and $\boldsymbol{m} \in \mathbb{N}^N$. Then, $\langle P, \boldsymbol{m} \rangle$ is an iterated path schema iff there are $\boldsymbol{v}_t$, $\boldsymbol{v}_p$ and $\boldsymbol{v}_{it}$ in $\mathbb{N}^K$ respecting $\mathfrak{C}$ and such that $P = P_{\boldsymbol{v}_t, \boldsymbol{v}_p}$ and $\boldsymbol{m} = \boldsymbol{m}_{\boldsymbol{v}_{it}}$.*
2. *One can build a (PA) formula $Schema(\mathsf{Z}_t, \mathsf{Z}_p, \mathsf{Z}_{it})$ of polynomial size in the size of the counter system S such that for all $\boldsymbol{v}_t, \boldsymbol{v}_p, \boldsymbol{v}_{it} \in \mathbb{N}^K$, we have $\boldsymbol{v}_t, \boldsymbol{v}_p, \boldsymbol{v}_{it} \models Schema(\mathsf{Z}_t, \mathsf{Z}_p, \mathsf{Z}_{it})$ iff $\boldsymbol{v}_t$, $\boldsymbol{v}_p$ and $\boldsymbol{v}_{it}$ satisfy $\mathfrak{C}$.*

Let us consider the following flat counter system.



There are 10 transitions and 3 simple loops. The enumeration of edges and loops is done as shown above. Let $\langle P, \mathbf{m}\rangle$ be such that $P = \delta_3 \cdot \delta_6 \cdot l_2 \cdot \delta_8 \cdot l_3$ and $\mathbf{m} = (5, 1, 1, 146, 1)$. So, we get the resulting ω-word as $\delta_3 \cdot \delta_6 \cdot (l_2)^{146} \cdot \delta_8 \cdot (l_3)^\omega$. From the previous encoding, the ω-word is encoded by vectors $\mathbf{v}_p = (5, 3, 6, 2, 8, 3, 0, 0, 0, 0, 0, 0, 0, 0)$, $\mathbf{v}_t = (5, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)$ and $\mathbf{v}_{it} = (5, 1, 1, 146, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)$.

4.2 Encoding runs using vectors

Lemma 4 states that any infinite run can be encoded by an iterated path schema. However, not every iterated path schema corresponds to a run in due form. We will see now how to check that an iterated path schema indeed represents a run starting from a given configuration c. First, we need to introduce a notion of *pseudo-run* in which only the updates are precise. Given $c \in Q \times \mathbb{Z}^n$, a *pseudo-run* ρ starting from c in S is an infinite sequence $\rho := c_0 \overset{\delta_1}{\rightsquigarrow} \cdots \overset{\delta_{m-1}}{\rightsquigarrow} c_m \overset{\delta_m}{\rightsquigarrow} \cdots$ where $c_0 = c$, $c_i = \langle q_i, \mathbf{v}_i\rangle \in Q \times \mathbb{Z}^n$ for all $i \geq 0$ and for all transitions $\delta \in \Delta$ we have $\langle q, \mathbf{v}\rangle \overset{\delta}{\rightsquigarrow} \langle q', \mathbf{v}'\rangle \overset{\text{def}}{\Leftrightarrow} q = \mathit{source}(\delta)$, $q' = \mathit{target}(\delta)$ and $\mathbf{v}' = \mathbf{v} + \mathit{update}(\delta)$. So, a pseudo-run $\rho = \langle q_0, \mathbf{v}_0\rangle \overset{\delta_1}{\rightsquigarrow} \cdots \overset{\delta_{m-1}}{\rightsquigarrow} \langle q_m, \mathbf{v}_m\rangle \cdots$ is a run iff for all $i \in \mathbb{N}$, $\mathbf{v}_i \models \mathit{guard}(\delta_i)$ and $\mathbf{v}_i \in \mathbb{N}^n$. Note also that for all configurations $\langle q, \mathbf{v}\rangle$, if $\langle P, \mathbf{m}\rangle$ is an iterated path schema such that $\mathit{source}(P[1]) = q$ then there exists a pseudo-run starting from c such that $\mathit{trans}(\rho) = \mathit{trans}(\langle P, \mathbf{m}\rangle)$.

From these observations, we conclude that an iterated path schema $\langle P, \mathbf{m}\rangle$ augmented with $c_0 = \langle q_0, \mathbf{v}_0\rangle$ verifying $\mathit{source}(P[1]) = q_0$, defines a unique pseudo-run that is denoted by $\rho(P, \mathbf{m}, c_0)$. Given a configuration $c_0 = \langle q_0, \mathbf{v}_0\rangle$, we say that $\rho(P, \mathbf{m}, c_0)$ is *well-defined* if $\mathit{source}(P[1]) = q_0$. For every $i \in [0, \text{size}(P) - 1]$ let $\mathfrak{p}_i \overset{\text{def}}{=} \sum_{j=1}^{j=i} \mathbf{m}[j] * \mathit{length}(P[j])$ with limit case $\mathfrak{p}_0 = 0$. The value $\mathfrak{p}_i$ is called the *position* of the *ith witness configuration* $\rho(P, \mathbf{m}, c_0)(\mathfrak{p}_i)$. Intuitively, we reach the ith witness configuration after going through the first i elements of the path schema P in $\rho(P, \mathbf{m}, c_0)$. We say that $\rho(P, \mathbf{m}, c_0)$ is *positive* if for all the witness configurations $\langle q, \mathbf{v}\rangle$, we have $\mathbf{v} \in \mathbb{N}^n$. Note that since for all path schemas P, we have $\text{size}(P) \leq \beta + \gamma$, the number of witness configurations for such a pseudo-run is bounded by $\beta + \gamma$.

Now, we show how to build a (PA) formula whose set of solutions corresponds to the witness configurations of a pseudo-run associated to an iterated path schema equipped with an initial configuration. Before defining the formula, we explain some further notions. In the sequel, we use the sets of variables $\mathsf{X}_0 = \{\mathsf{x}_0^1, \ldots, \mathsf{x}_0^{n+1}\}$ and $\mathsf{X} = \{\mathsf{x}^1, \ldots, \mathsf{x}^{n+1}\}$ to represent configurations and $\mathsf{W}_i =$

$\{\mathsf{w}_i^1, \ldots, \mathsf{w}_i^{n+1}\}$ for every $i \in [0, \beta+\gamma-1]$ to represent pseudo-configurations and the variables $\mathsf{p}_0, \ldots, \mathsf{p}_{\beta+\gamma-1}$ and y to represent positions in (pseudo)-runs. Furthermore, given sets of variables X, W representing (pseudo)-configurations, a variable x and a vector $\mathbf{u} \in \mathbb{N}^n$, we use the shortcut $\mathsf{X} = \mathsf{W} + \mathsf{x}.\mathbf{u}$ to denote the formula $\bigwedge_{i=2}^{n+1} \mathsf{x}^i = \mathsf{w}^i + \mathsf{x}.\mathbf{u}[i-1]$. Let us define the formula *Witness* that states whether a given set of configurations and natural numbers represent the witness configurations and their respective positions in a pseudo-run associated to an iterated path schema. The main idea of the formula is to check at each step whether the control states of the witness configurations match with the states of the taken transitions or loops in the path schema and then to compute the effect of the corresponding element of the iterated path schema taking into account the number of iterations.

$$\begin{array}{c} \mathit{Witness}(\mathsf{W}_0, \ldots, \mathsf{W}_{\beta+\gamma-1}, \mathsf{p}_0, \ldots, \mathsf{p}_{\beta+\gamma-1}, \mathsf{Z}_t, \mathsf{Z}_p, \mathsf{Z}_{it}, \mathsf{X}_0) \stackrel{\text{def}}{=} \\ (\mathsf{p}_0 = 0 \wedge \mathsf{X}_0 = \mathsf{W}_0 \wedge \bigvee_{t=0}^{1} \bigvee_{j=1}^{\max(\beta,\gamma)} \mathsf{z}_p^2 = j \wedge \mathsf{z}_t^2 = t \wedge \mathsf{x}_0^1 = \mathit{source}(\tau(t,j))) \wedge \end{array}$$

$$\begin{array}{c} \bigwedge_{i=1}^{\beta+\gamma-1} (i < \mathsf{z}_t^1 \Rightarrow \bigvee_{t=0}^{1} \bigvee_{j=1}^{\max(\beta,\gamma)} (\mathsf{z}_p^{i+1} = j \wedge \mathsf{z}_t^{i+1} = t \wedge \mathsf{p}_i = \mathsf{p}_{i-1} + \mathsf{z}_{it}^{i+1} * \mathit{length}(\tau(t,j)) \wedge \\ \mathsf{w}_i^1 = \mathit{target}(\tau(t,j)) \wedge \mathsf{W}_i = \mathsf{W}_{i-1} + \mathsf{z}_{it}^{i+1} * \mathit{update}(\tau(t,i))) \end{array}$$

Lemma 6 below characterizes the formula *Witness*.

Lemma 6. *Let* $\boldsymbol{w}_0, \ldots, \boldsymbol{w}_{\beta+\gamma-1}, \boldsymbol{c}_0 \in \mathbb{N}^{n+1}$ *and* $\mathfrak{p}_0, \ldots, \mathfrak{p}_{\beta+\gamma-1} \in \mathbb{N}$ *and* $\boldsymbol{v}_t$, $\boldsymbol{v}_p$, $\boldsymbol{v}_{it}$ *in* $\mathbb{N}^K$ *such that* $\boldsymbol{v}_t, \boldsymbol{v}_p, \boldsymbol{v}_{it} \models \mathit{Schema}(\mathsf{Z}_t, \mathsf{Z}_p, \mathsf{Z}_{it})$. *We have* $\boldsymbol{w}_0, \ldots, \boldsymbol{w}_{\beta+\gamma-1}, \mathfrak{p}_0, \ldots, \mathfrak{p}_{\beta+\gamma-1}, \boldsymbol{v}_t, \boldsymbol{v}_p, \boldsymbol{v}_{it}, \boldsymbol{c}_0 \models \mathit{Witness}$ *iff* $\rho(P_{\boldsymbol{v}_t,\boldsymbol{v}_p}, \boldsymbol{m}_{\boldsymbol{v}_{it}}, \boldsymbol{c}_0)$ *is well-defined and positive and for all* $j \in [0, \beta+\gamma-1]$, *if* $j < \mathrm{size}(P_{\boldsymbol{v}_t,\boldsymbol{v}_p})$, *then* $\boldsymbol{w}_j$ *represents the* j*th witness configuration of* $\rho(P_{\boldsymbol{v}_t,\boldsymbol{v}_p}, \boldsymbol{m}_{\boldsymbol{v}_{it}}, c_0)$ *and* $\mathfrak{p}_j$ *its position.*

Using *Witness*, one can build in logarithmic space a formula to check whether a vector $\mathbf{c}$ is the ith configuration $\langle q_i, \mathbf{v}_i \rangle$ of a pseudo-run $\rho(P_{\mathbf{v}_t,\mathbf{v}_p}, \mathbf{m}_{\mathbf{v}_{it}}, c_0)$ with the property that $\mathbf{v}_i \models \mathit{guard}(\mathit{trans}(\langle P_{\mathbf{v}_t,\mathbf{v}_p}, \mathbf{m}_{\mathbf{v}_{it}} \rangle)[i+1])$ and $\mathbf{v}_i \in \mathbb{N}^n$ (here i is the number of transitions to reach that configuration) and then to construct a formula to check whether a pseudo-run is a run. In fact, as observed earlier, it is enough to check whether at each step the ith configuration satisfies the guard of the $(i+1)$th transition.

Lemma 7. *One can build in logarithmic-space in the size of flat counter system* S *two PA formulae* $\mathit{Conf}(\mathsf{Z}_t, \mathsf{Z}_p, \mathsf{Z}_{it}, \mathsf{X}_0, \mathsf{y}, \mathsf{X})$ *and* $\mathit{Run}(\mathsf{Z}_t, \mathsf{Z}_p, \mathsf{Z}_{it}, \mathsf{X}_0)$ *such that for all* $\boldsymbol{c}_0, \boldsymbol{c} \in \mathbb{N}^{n+1}$, *for all* $i \in \mathbb{N}$ *and for all* $\boldsymbol{v}_t, \boldsymbol{v}_p, \boldsymbol{v}_{it} \in \mathbb{N}^{\beta+\gamma+1}$, *we have the two following properties:*

1. $\boldsymbol{v}_t, \boldsymbol{v}_p, \boldsymbol{v}_{it}, \boldsymbol{c}_0, i, \boldsymbol{c} \models \mathit{Conf}$ *iff* $\boldsymbol{v}_t, \boldsymbol{v}_p, \boldsymbol{v}_{it} \models \mathit{Schema}$ *and* $\rho(P_{\boldsymbol{v}_t,\boldsymbol{v}_p}, \boldsymbol{m}_{\boldsymbol{v}_{it}}, \boldsymbol{c}_0)$ *is well defined and* $\boldsymbol{c} = \rho(P_{\boldsymbol{v}_t,\boldsymbol{v}_p}, \boldsymbol{m}_{\boldsymbol{v}_{it}}, c_0)(i)$ *and* $\boldsymbol{c}[2], \cdots, \boldsymbol{c}[n+1] \models \mathit{guard}(\mathit{trans}(\langle P_{\boldsymbol{v}_t,\boldsymbol{v}_p}, \boldsymbol{m}_{\boldsymbol{v}_{it}} \rangle)[i+1])$
2. $\boldsymbol{v}_t, \boldsymbol{v}_p, \boldsymbol{v}_{it}, \boldsymbol{c}_0 \models \mathit{Run}$ *iff* $\boldsymbol{v}_t, \boldsymbol{v}_p, \boldsymbol{v}_{it} \models \mathit{Schema}$ *and* $\rho(P_{\boldsymbol{v}_t,\boldsymbol{v}_p}, \boldsymbol{m}_{\boldsymbol{v}_{it}}, c_0)$ *is well-defined and is a run.*

4.3 Encoding CTL* formulae using (PA)

We can encode path schemas and runs using vectors and check their validity using Presburger arithmetic formula, our next aim is to encode a given CTL* formula using a formula in (PA). The forthcoming encoding internalizes CTL* semantics and a similar idea has been already used in [10]. For each CTL* formula ϕ, we build a (PA) formula $\mathit{Check}_\phi(\mathsf{Z}_t, \mathsf{Z}_p, \mathsf{Z}_{it}, \mathsf{X}_0, \mathsf{y})$ where the variables $\mathsf{Z}_t, \mathsf{Z}_p, \mathsf{Z}_{it}$ and X_0 represent a run as in the formula *Run* and, y represents a position such that the formula checks whether the CTL* formula is satisfied at the current position. Formula Check_ϕ is defined recursively (Boolean clauses are omitted):

$$
\begin{array}{ll}
\mathit{Check}_{\mathtt{p}} & \stackrel{\text{def}}{=} \exists\, \mathsf{X}\ (\mathit{Conf}(\mathsf{Z}_t, \mathsf{Z}_p, \mathsf{Z}_{it}, \mathsf{X}_0, \mathsf{y}, \mathsf{X}) \wedge \bigvee_{\{j \mid \mathtt{p} \in \ell(j)\}} \mathsf{x}^1 = j) \\
\mathit{Check}_{\psi(\mathsf{x}_1,\ldots,\mathsf{x}_n)} & \stackrel{\text{def}}{=} \exists\, \mathsf{X}\ (\mathit{Conf}(\mathsf{Z}_t, \mathsf{Z}_p, \mathsf{Z}_{it}, \mathsf{X}_0, \mathsf{y}, \mathsf{X}) \wedge \psi(\mathsf{X})) \\
\mathit{Check}_{\mathtt{X}\phi} & \stackrel{\text{def}}{=} \exists\, \mathsf{y}'\ (\mathsf{y}' = \mathsf{y} + 1 \wedge \mathit{Check}_\phi(\mathsf{Z}_t, \mathsf{Z}_p, \mathsf{Z}_{it}, \mathsf{X}_0, \mathsf{y}')) \\
\mathit{Check}_{\phi \mathtt{U} \phi'} & \stackrel{\text{def}}{=} \exists\, \mathsf{y}''\ (\mathsf{y} \leq \mathsf{y}'' \wedge \mathit{Check}_{\phi'}(\mathsf{Z}_t, \mathsf{Z}_p, \mathsf{Z}_{it}, \mathsf{X}_0, \mathsf{y}'') \wedge \\
 & \quad \forall\, \mathsf{y}'\ (\mathsf{y} \leq \mathsf{y}' < \mathsf{y}'' \Rightarrow \mathit{Check}_\phi(\mathsf{Z}_t, \mathsf{Z}_p, \mathsf{Z}_{it}, \mathsf{X}_0, \mathsf{y}'))) \\
\mathit{Check}_{\mathtt{E}\phi} & \stackrel{\text{def}}{=} \exists\, \mathsf{Z}'_t\ \exists\, \mathsf{Z}'_p\ \exists\, \mathsf{Z}'_{it}\ \exists\, \mathsf{X}\ (\mathit{Conf}(\mathsf{Z}_t, \mathsf{Z}_p, \mathsf{Z}_{it}, \mathsf{X}_0, \mathsf{y}, \mathsf{X}) \wedge \\
 & \quad \mathit{Run}(\mathsf{Z}'_t, \mathsf{Z}'_p, \mathsf{Z}'_{it}, \mathsf{X}) \wedge\ \exists\, \mathsf{y}'\ (\mathsf{y}' = 0 \wedge \mathit{Check}_\phi(\mathsf{Z}'_t, \mathsf{Z}'_p, \mathsf{Z}'_{it}, \mathsf{X}, \mathsf{y}')))
\end{array}
$$

Now, we can state the main property concerning the formulae Check_ϕ based on Lemmas 4, 5 and 7.

Lemma 8. *Let $\boldsymbol{c}_0 \in \mathbb{N}^{n+1}$, $i \in \mathbb{N}$ and $\boldsymbol{v}_t, \boldsymbol{v}_p, \boldsymbol{v}_{it} \in \mathbb{N}^K$ be such that $\boldsymbol{v}_t, \boldsymbol{v}_p, \boldsymbol{v}_{it}, \boldsymbol{c}_0 \models \mathit{Run}$. We have $\rho(P_{\boldsymbol{v}_t, \boldsymbol{v}_p}, \boldsymbol{m}_{\boldsymbol{v}_{it}}, \boldsymbol{c}_0), i \models \phi$ iff $\boldsymbol{v}_t, \boldsymbol{v}_p, \boldsymbol{v}_{it}, \boldsymbol{c}_0, i \models \mathit{Check}_\phi(\mathsf{Z}_t, \mathsf{Z}_p, \mathsf{Z}_{it}, \mathsf{X}_0, \mathsf{y})$.*

This allows us to conclude the main result of this section.

Theorem 9. *There is a logarithmic-space reduction from MC(*CTL*, `FlatCS`) to the satisfiability problem for (PA).*

It is possible to extend the reduction by admitting linear past-time operators to the temporal language since we have seen that we can easily quantify over runs. However, in that case, finite prefixes in runs should not be reset.

5 Conclusion

We have been able to characterize the computational complexity for MC(CTL*, `FlatCS`) by showing that the problem is equivalent to the satisfiability problem for Presburger arithmetic (modulo logarithmic-space reductions). The lower bound is obtained by considering a quite strong restriction (no arithmetical constraints in formulae, the only temporal operator is `EF`, guards on transitions are simple linear constraints). By contrast, the restriction of the problem to LTL formulae is known to be NP-complete [8] when guards are in the linear fragment and the restriction of the problem to formulae in $\mathrm{CTL}_{\mathtt{EF}}$ is also equivalent to

(PA). We have proposed a new way for encoding runs in flat counter systems using Presburger arithmetic formulae, but without any exponential blow up, which allows us to get a precise complexity characterization. It remains open to determine which extensions of CTL* on flat counter systems preserve decidability, if not an efficient translation into (PA).

References

1. L. Berman. The complexity of logical theories. *TCS*, 11:71–78, 1980.
2. M. Bersani and S. Demri. The complexity of reversal-bounded model-checking. In *FROCOS'11*, volume 6989 of *LNAI*, pages 71–86. Springer, 2011.
3. B. Boigelot. *Symbolic methods for exploring infinite state spaces*. PhD thesis, Université de Liège, 1998.
4. M. Bozga, R. Iosif, and F. Konecny. Safety problems are NP-complete for flat integer programs with octagonal loops. In *VMCAI'14*, volume 8318 of *LNCS*, pages 242–261. Springer, 2014.
5. V. Bruyère, E. Dall'Olio, and J. Raskin. Durations, parametric model-checking in timed automata with Presburger arithmetic. In *STACS'03*, volume 2607 of *LNCS*, pages 687–698. Springer, 2003.
6. H. Comon and Y. Jurski. Multiple counter automata, safety analysis and Presburger Arithmetic. In *CAV'98*, volume 1427 of *LNCS*, pages 268–279. Springer, 1998.
7. L. de Moura and N. Björner. Z3: An Efficient SMT Solver. In *TACAS'08*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.
8. S. Demri, A. Dhar, and A. Sangnier. Taming Past LTL and Flat Counter Systems. In *IJCAR'12*, volume 7364 of *LNAI*, pages 179–193. Springer, 2012.
9. S. Demri, A. Dhar, and A. Sangnier. On the complexity of verifying regular properties on flat counter systems. In *ICALP'13*, volume 7966 of *LNCS*, pages 162–173. Springer, 2013.
10. S. Demri, A. Finkel, V. Goranko, and G. van Drimmelen. Model-checking CTL* over flat Presburger counter systems. *JANCL*, 20(4):313–344, 2010.
11. A. K. Dhar. PhD thesis, Université Paris VII-Denis Diderot, 2014.
12. A. Emerson and J. Halpern. 'sometimes' and 'not never' revisited: on branching versus linear time temporal logic. *JACM*, 33:151–178, 1986.
13. E. A. Emerson and C.-L. Lei. Modalities for model checking: Branching time logic strikes back. *Sci. Comput. Program.*, 8(3):275–306, 1987.
14. A. Finkel and J. Leroux. How to compose Presburger accelerations: Applications to broadcast protocols. In *FST&TCS'02*, volume 2256 of *LNCS*, pages 145–156. Springer, 2002.
15. S. Göller, C. Haase, J. Ouaknine, and J. Worrell. Branching-time model checking of parametric one-counter automata. In *FoSSaCS'12*, volume 7213 of *LNCS*, pages 406–420. Springer, 2012.
16. S. Göller and M. Lohrey. Branching-time model checking of one-counter processes and timed automata. *SIAM J. Comput.*, 42(3):884–923, 2013.
17. P. Habermehl. On the complexity of the linear-time mu-calculus for Petri nets. In *ICATPN'97*, volume 1248 of *LNCS*, pages 102–116. Springer, 1997.
18. F. Laroussinie, N. Markey, and P. Schnoebelen. Model checking CTL$^+$ and FCTL is hard. In *FOSSACS'01*, volume 2030 of *LNCS*, pages 318–331. Springer, 2001.
19. J. Leroux. Presburger counter machines. *Habilitation thesis, U. of Bordeaux*, 2012.

20. J. Leroux and G. Point. TaPAS: The Talence Presburger Arithmetic Suite. In *TACAS'09*, volume 5505 of *LNCS*, pages 182–185. Springer, 2009.
21. J. Leroux and G. Sutre. On flatness for 2-dimensional vector addition systems with states. In *CONCUR*, volume 3170 of *LNCS*, pages 402–416. Springer, 2004.
22. J. Leroux and G. Sutre. Flat counter automata almost everywhere! In *ATVA'05*, volume 3707 of *LNCS*, pages 489–503. Springer, 2005.
23. M. Minsky. *Computation, Finite and Infinite Machines*. Prentice Hall, 1967.
24. M. Presburger. 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*, pages 92–101, 1929.