On the Complexity of Verifying Regular Properties on Flat Counter Systems*,**

Stéphane Demri[2,3], Amit Kumar Dhar[1], and Arnaud Sangnier[1]

[1] LIAFA, Univ Paris Diderot, Sorbonne Paris Cité, CNRS, France
[2] New York University, USA
[3] LSV, CNRS, France

Abstract. Among the approximation methods for the verification of counter systems, one of them consists in model-checking their flat unfoldings. Unfortunately, the complexity characterization of model-checking problems for such operational models is not always well studied except for reachability queries or for Past LTL. In this paper, we characterize the complexity of model-checking problems on flat counter systems for the specification languages including first-order logic, linear mu-calculus, infinite automata, and related formalisms. Our results span different complexity classes (mainly from PTime to PSpace) and they apply to languages in which arithmetical constraints on counter values are systematically allowed. As far as the proof techniques are concerned, we provide a uniform approach that focuses on the main issues.

1 Introduction

Flat Counter Systems. Counter systems, finite-state automata equipped with program variables (counters) interpreted over non-negative integers, are known to be ubiquitous in formal verification. Since counter systems can actually simulate Turing machines [17], it is undecidable to check the existence of a run satisfying a given (reachability, temporal, etc.) property. However it is possible to approximate the behavior of counter systems by looking at a subclass of witness runs for which an analysis is feasible. A standard method consists in considering a finite union of path schemas for abstracting the whole bunch of runs, as done in [14]. More precisely, given a finite set of transitions Δ, a *path schema* is an ω-regular expression over Δ of the form $\mathrm{L} = p_1(l_1)^* \cdots p_{k-1}(l_{k-1})^* p_k(l_k)^\omega$ where both p_i's and l_i's are paths in the control graph and moreover, the l_i's are loops. A path schema defines a set of infinite runs that respect a sequence of transitions that belongs to L. We write $\texttt{Runs}(c_0, \mathrm{L})$ to denote such a set of runs starting at the initial configuration c_0 whereas $\texttt{Reach}(c_0, \mathrm{L})$ denotes the set of configurations occurring in the runs of $\texttt{Runs}(c_0, \mathrm{L})$. A counter system is *flattable* whenever the set of configurations reachable from c_0 is equal to $\texttt{Reach}(c_0, \mathrm{L})$ for

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

** A version with proofs is available as [5].

F.V. Fomin et al. (Eds.): ICALP 2013, Part II, LNCS 7966, pp. 162–173, 2013.

some finite union of path schemas L. Similarly, a *flat counter system*, a system in which each control state belongs to at most one simple loop, verifies that the set of runs from c_0 is equal to $\texttt{Runs}(c_0, \mathrm{L})$ for some finite union of path schemas L. Obviously, flat counter systems are flattable. Moreover, reachability sets of flattable counter systems are known to be Presburger-definable, see e.g. [1,3,7]. That is why, verification of flat counter systems belongs to the core of methods for model-checking arbitrary counter systems and it is desirable to characterize the computational complexity of model checking problems on this kind of systems (see e.g. results about loops in [2]). Decidability results for verifying safety and reachability properties on flat counter systems have been obtained in [3,7,2]. For the verification of temporal properties, it is much more difficult to get sharp complexity characterization. For instance, it is known that verifying flat counter systems with CTL* enriched with arithmetical constraints is decidable [6] whereas it is only NP-complete with Past LTL [4] (NP-completeness already holds with flat Kripke structures [10]).

Our Motivations. Our objectives are to provide a thorough classification of model-checking problems on flat counter systems when linear-time properties are considered. So far complexity is known with Past LTL [4] but even the decidability status with linear μ-calculus is unknown. Herein, we wish to consider several formalisms specifying linear-time properties (FO, linear μ-calculus, infinite automata) and to determine the complexity of model-checking problems on flat counter systems. Note that FO is as expressive as Past LTL but much more concise whereas linear μ-calculus is strictly more expressive than Past LTL, which motivates the choice for these formalisms dealing with linear properties.

Our Contributions. We characterize the computational complexity of model-checking problems on flat counter systems for several prominent linear-time specification languages whose alphabets are related to atomic propositions but also to linear constraints on counter values. We obtain the following results:

- The problem of **model-checking first-order formulae on flat counter systems is PSpace-complete** (Theorem 9). Note that model-checking classical first-order formulae over arbitrary Kripke structures is already known to be non-elementary. However the flatness assumption allows to drop the complexity to PSpace even though linear constraints on counter values are used in the specification language.
- **Model-checking linear μ-calculus formulae on flat counter systems is PSpace-complete** (Theorem 14). Not only linear μ-calculus is known to be more expressive than first-order logic (or than Past LTL) but also the decidability status of the problem on flat counter systems was open [6]. So, we establish decidability and we provide a complexity characterization.
- **Model-checking Büchi automata over flat counter systems is NP-complete** (Theorem 12).
- **Global model-checking is possible** for all the above mentioned formalisms (Corollary 16).

2 Preliminaries

2.1 Counter Systems

Counter constraints are defined below as a subclass of Presburger formulae whose free variables are understood as counters. Such constraints are used to define guards in counter systems but also to define arithmetical constraints in temporal formulae. Let $\mathtt{C} = \{\mathtt{x}_1, \mathtt{x}_2, \ldots\}$ be a countably infinite set of *counters* (variables interpreted over non-negative integers) and $\mathrm{AT} = \{\mathtt{p}_1, \mathtt{p}_2, \ldots\}$ be a countable infinite set of propositional variables (abstract properties about program points). We write $\mathtt{C}_n$ to denote the restriction of $\mathtt{C}$ to $\{\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n\}$. The set of *guards* $\mathtt{g}$ using the counters from $\mathtt{C}_n$, written $\mathtt{G}(\mathtt{C}_n)$, is made of Boolean combinations of *atomic guards* of the form $\sum_{i=0}^{n} a_i \cdot \mathtt{x}_i \sim b$ where the a_i's are in $\mathbb{Z}$, $b \in \mathbb{N}$ and $\sim \in \{=, \leq, \geq, <, >\}$. For $\mathtt{g} \in \mathtt{G}(\mathtt{C}_n)$ and a vector $\mathbf{v} \in \mathbb{N}^n$, we say that $\mathbf{v}$ satisfies $\mathtt{g}$, written $\mathbf{v} \models \mathtt{g}$, if the formula obtained by replacing each $\mathtt{x}_i$ by $\boldsymbol{v}[i]$ holds. For $n \geq 1$, a *counter system* of dimension n (shortly a counter system) S is a tuple $\langle Q, \mathtt{C}_n, \Delta, \mathbf{l}\rangle$ where: Q is a finite set of *control states*, $\mathbf{l} : Q \to 2^{\mathrm{AT}}$ is a *labeling function*, $\Delta \subseteq Q \times \mathtt{G}(\mathtt{C}_n) \times \mathbb{Z}^n \times Q$ is a finite set of transitions labeled by guards and updates. As usual, to a counter system $S = \langle Q, \mathtt{C}_n, \Delta, \mathbf{l}\rangle$, we associate a labeled transition system $TS(S) = \langle C, \to\rangle$ where $C = Q \times \mathbb{N}^n$ is the set of *configurations* and $\to \subseteq C \times \Delta \times C$ 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 $\delta = \langle q, \mathtt{g}, \mathbf{u}, q'\rangle \in \Delta$, $\mathbf{v} \models \mathtt{g}$ and $\mathbf{v}' = \mathbf{v} + \mathbf{u}$. Note that in such a transition system, the counter values are non-negative since $C = Q \times \mathbb{N}^n$.

Given an initial configuration $c_0 \in Q \times \mathbb{N}^n$, a *run* ρ starting from c_0 in S is an infinite path in the associated transition system $TS(S)$ 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 $i \in \mathbb{N}$. We say that 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) [3,14,4]. We denote by $\mathcal{CFS}$ the class of flat counter systems. A *Kripke structure* S can be seen as a counter system without counter and is denoted by $\langle Q, \Delta, \mathbf{l}\rangle$ where $\Delta \subseteq Q \times Q$ and $\mathbf{l} : Q \to 2^{\mathrm{AT}}$. Standard notions on counter systems, as configuration, run or flatness, naturally apply to Kripke structures.

2.2 Model-Checking Problem

We define now our main model-checking problem on flat counter systems parameterized by a specification language $\mathcal{L}$. First, we need to introduce the notion of constrained alphabet whose letters should be understood as Boolean combinations of atomic formulae (details follow). A *constrained alphabet* is a triple of the form $\langle at, ag_n, \Sigma\rangle$ where at is a finite subset of AT, ag_n is a finite subset of atomic guards from $\mathtt{G}(\mathtt{C}_n)$ and Σ is a subset of $2^{at \cup ag_n}$. The size of a constrained alphabet is given by $\mathrm{size}(\langle at, ag_n, \Sigma\rangle) = \mathrm{card}(at) + \mathrm{card}(ag_n) + \mathrm{card}(\Sigma)$ where $\mathrm{card}(X)$ denotes the cardinality of the set X. Of course, any standard alphabet (finite set of letters) can be easily viewed as a constrained alphabet (by ignoring the structure of letters). Given an infinite run $\rho := \langle q_0, \mathbf{v}_0\rangle \to \langle q_1, \mathbf{v}_1\rangle \cdots$ from

a counter system with n counters and an ω-word over a constrained alphabet $w = a_0, a_1, \ldots \in \Sigma^\omega$, we say that ρ *satisfies* w, written $\rho \models w$, whenever for $i \geq 0$, we have $\mathtt{p} \in \mathbf{l}(q_i)$ [resp. $\mathtt{p} \notin \mathbf{l}(q_i)$] for every $\mathtt{p} \in (a_i \cap at)$ [resp. $\mathtt{p} \in (at \setminus a_i)$] and $\mathbf{v}_i \models \mathtt{g}$ [resp. $\mathbf{v}_i \not\models \mathtt{g}$] for every $\mathtt{g} \in (a_i \cap ag_n)$ [resp. $\mathtt{g} \in (ag_n \setminus a_i)$].

A *specification language* $\mathcal{L}$ over a constrained alphabet $\langle at, ag_n, \Sigma\rangle$ is a set of *specifications* A, each of it defining a set $\mathrm{L}(A)$ of ω-words over Σ. We will also sometimes consider specification languages over (unconstrained) standard finite alphabets (as usually defined). We now define the *model-checking problem* over flat counter systems with specification language $\mathcal{L}$ (written $\mathrm{MC}(\mathcal{L}, \mathcal{CFS})$): it takes as input a flat counter system S, a configuration c and a specification A from $\mathcal{L}$ and asks whether there is a run ρ starting at c and $w \in \Sigma^\omega$ in $\mathrm{L}(A)$ such that $\rho \models w$. We write $\rho \models A$ whenever there is $w \in \mathrm{L}(A)$ such that $\rho \models w$.

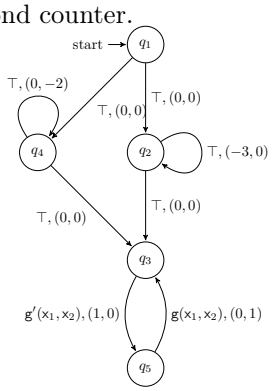
2.3 A Bunch of Specification Languages

Infinite Automata. Now let us define the specification languages BA and ABA, respectively with nondeterministic Büchi automata and with alternating Büchi automata. We consider here transitions labeled by Boolean combinations of atoms from $at \cup ag_n$. A specification A in ABA is a structure of the form $\langle Q, E, q_0, F\rangle$ where E is a finite subset of $Q \times \mathbb{B}(at \cup ag_n) \times \mathbb{B}^+(Q)$ and $\mathbb{B}^+(Q)$ denotes the set of positive Boolean combinations built over Q. Specification A is a concise representation for the alternating Büchi automaton $\mathcal{B}_A = \langle Q, \delta, q_0, F\rangle$ where $\delta : Q \times 2^{at \cup ag_n} \to \mathbb{B}^+(Q)$ and $\delta(q, a) \stackrel{\text{def}}{=} \bigvee_{\langle q, \psi, \psi'\rangle \in E,\ a \models \psi} \psi'$. We say that A is over the constrained alphabet $\langle at, ag_n, \Sigma\rangle$, whenever, for all edges $\langle q, \psi, \psi'\rangle \in E$, ψ holds at most for letters from Σ (i.e. the transition relation of $\mathcal{B}_A$ belongs to $Q \times \Sigma \to \mathbb{B}^+(Q)$). We have then $\mathrm{L}(A) = \mathrm{L}(\mathcal{B}_A)$ with the usual acceptance criterion for alternating Büchi automata. The specification language BA is defined in a similar way using Büchi automata. Hence the transition relation E of $A = \langle Q, E, q_0, F\rangle$ in BA is included in $Q \times \mathbb{B}(at \cup ag_n) \times Q$ and the transition relation of the Büchi automaton $\mathcal{B}_A$ is then included in $Q \times 2^{at \cup ag_n} \times Q$.

Linear-Time Temporal Logics. Below, we present briefly three logical languages that are tailored to specify runs of counter systems, namely ETL (see e.g.[25,19]), Past LTL (see e.g. [21]) and linear μ-calculus (or μTL), see e.g. [23]. A specification in one of these logical specification languages is just a formula. The differences with their standard versions in which models are ω-sequences of propositional valuations are listed below: models are infinite runs of counters systems; atomic formulae are either propositional variables in AT or atomic guards; given an infinite run $\rho := \langle q_0, \mathbf{v}_0\rangle \to \langle q_1, \mathbf{v}_1\rangle \cdots$, we will have $\rho, i \models \mathtt{p} \stackrel{\text{def}}{\Leftrightarrow} \mathtt{p} \in \mathbf{l}(q_i)$ and $\rho, i \models \mathtt{g} \stackrel{\text{def}}{\Leftrightarrow} \mathbf{v}_i \models \mathtt{g}$. The temporal operators, fixed point operators and automata-based operators are interpreted then as usual. A formula ϕ built over the propositional variables in at and the atomic guards in ag_n defines a language $\mathrm{L}(\phi)$ over $\langle at, ag_n, \Sigma\rangle$ with $\Sigma = 2^{at \cup ag_n}$. There is no need to recall here the syntax and semantics of ETL, Past LTL and linear μ-calculus since with their standard definitions and with the above-mentioned differences, their variants for counter systems are defined unambiguously (see a lengthy presentation

of Past LTL for counter systems in [4]). However, we may recall a few definitions on-the-fly if needed. Herein the size of formulae is understood as the number of subformulae.

Example. In adjoining figure, we present a flat counter system with two counters and with labeling function $\mathbf{l}$ such that $\mathbf{l}(q_3) = \{\mathtt{p}, \mathtt{q}\}$ and $\mathbf{l}(q_5) = \{\mathtt{p}\}$. We would like to characterize the set of configurations c with control state q_1 such that there is some infinite run from c for which after some position i, all future even positions j (i.e. $i \equiv_2 j$) satisfy that $\mathtt{p}$ holds and the first counter is equal to the second counter.



This can be specified in linear μ-calculus using as atomic formulae either propositional variables or atomic guards. The corresponding formula in linear μ-calculus is: $\mu \mathtt{z}_1.(\mathtt{X}(\nu \mathtt{z}_2.(\mathtt{p} \wedge (\mathtt{x}_1 - \mathtt{x}_2 = 0) \wedge \mathtt{X}\mathtt{X}\mathtt{z}_2) \vee \mathtt{X}\mathtt{z}_1)$. Clearly, such a position i occurs in any run after reaching the control state q_3 with the same value for both counters. Hence, the configurations $\langle q_1, \mathbf{v} \rangle$ satisfying these properties have counter values $\mathbf{v} \in \mathbb{N}^2$ verifying the Presburger formula below:

$$\exists\, \mathtt{y}\ (((\mathtt{x}_1 = 3\mathtt{y} + \mathtt{x}_2) \wedge (\forall\, \mathtt{y}'\ \mathtt{g}(\mathtt{x}_2 + \mathtt{y}', \mathtt{x}_2 + \mathtt{y}') \wedge \mathtt{g}'(\mathtt{x}_2 + \mathtt{y}', \mathtt{x}_2 + \mathtt{y}' + 1))) \vee$$

$$((\mathtt{x}_2 = 2\mathtt{y} + \mathtt{x}_1) \wedge (\forall\, \mathtt{y}'\ \mathtt{g}(\mathtt{x}_1 + \mathtt{y}', \mathtt{x}_1 + \mathtt{y}') \wedge \mathtt{g}'(\mathtt{x}_1 + \mathtt{y}', \mathtt{x}_1 + \mathtt{y}' + 1))))$$

In the paper, we shall establish how to compute systematically such formulae (even without universal quantifications) for different specification languages.

3 Constrained Path Schemas

In [4] we introduced minimal path schemas for flat counter systems. Now, we introduce *constrained path schemas* that are more abstract than path schemas. A *constrained path schema* $\mathtt{cps}$ is a pair $\langle p_1(l_1)^* \cdots p_{k-1}(l_{k-1})^* p_k(l_k)^\omega, \phi(\mathtt{x}_1, \ldots, \mathtt{x}_{k-1}) \rangle$ where the first component is an ω-regular expression over a constrained alphabet $\langle at, ag_n, \Sigma \rangle$ with p_i, l_i's in Σ^*, and $\phi(\mathtt{x}_1, \ldots, \mathtt{x}_{k-1}) \in \mathtt{G}(\mathtt{C}_{k-1})$. Each constrained path schema defines a language $\mathrm{L}(\mathtt{cps}) \subseteq \Sigma^\omega$ given by $\mathrm{L}(\mathtt{cps}) \stackrel{\text{def}}{=} \{p_1(l_1)^{n_1} \cdots p_{k-1}(l_{k-1})^{n_{k-1}} p_k(l_k)^\omega \ :\ \phi(n_1, \ldots, n_{k-1})$ holds true$\}$. The size of $\mathtt{cps}$, written $\mathrm{size}(\mathtt{cps})$, is equal to $2k + \mathrm{len}(p_1 l_1 \cdots p_{k-1} l_{k-1} p_k l_k) + \mathrm{size}(\phi(\mathtt{x}_1, \ldots, \mathtt{x}_{k-1}))$. Observe that in general constrained path schemas are defined under constrained alphabet and so will the associated specifications unless stated otherwise.

Let us consider below the three decision problems on constrained path schemas that are useful in the rest of the paper. *Consistency problem* checks whether

$\mathrm{L}(\mathtt{cps})$ is non-empty. It amounts to verify the satisfiability status of the second component. Let us recall the result below.

Theorem 1. *[20] There are polynomials* $\mathtt{pol}_1(\cdot)$, $\mathtt{pol}_2(\cdot)$ *and* $\mathtt{pol}_3(\cdot)$ *such that for every guard* $\mathtt{g}$*, say in* $\mathtt{G}(\mathtt{C}_n)$*, of size* N*, we have (I) there exist* $B \subseteq [0, 2^{\mathtt{pol}_1(N)}]^n$ *and* $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_\alpha \in [0, 2^{\mathtt{pol}_1(N)}]^n$ *with* $\alpha \leq 2^{\mathtt{pol}_2(N)}$ *such that for every* $\boldsymbol{y} \in \mathbb{N}^n$, $\boldsymbol{y} \models \mathtt{g}$ *iff there are* $\boldsymbol{b} \in B$ *and* $\boldsymbol{a} \in \mathbb{N}^\alpha$ *such that* $\boldsymbol{y} = \boldsymbol{b} + \boldsymbol{a}[1]\boldsymbol{P}_1 + \cdots + \boldsymbol{a}[\alpha]\boldsymbol{P}_\alpha$*; (II) if* $\mathtt{g}$ *is satisfiable, then there is* $\boldsymbol{y} \in [0, 2^{\mathtt{pol}_3(N)}]^n$ *s.t.* $\boldsymbol{y} \models \mathtt{g}$.

Consequently, the consistency problem is NP-complete (the hardness being obtained by reducing SAT). The *intersection non-emptiness problem*, clearly related to model-checking problem, takes as input a constrained path schema $\mathtt{cps}$ and a specification $A \in \mathcal{L}$ and asks whether $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(A) \neq \emptyset$. Typically, for several specification languages $\mathcal{L}$, we establish the existence of a computable map $f_{\mathcal{L}}$ (at most exponential) such that whenever $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(A) \neq \emptyset$ there is $p_1(l_1)^{n_1} \cdots p_{k-1}(l_{k-1})^{n_{k-1}} p_k(l_k)^\omega$ belonging to the intersection and for which each n_i is bounded by $f_{\mathcal{L}}(A, \mathtt{cps})$. This motivates the introduction of the *membership problem* for $\mathcal{L}$ that takes as input a constrained path schema $\mathtt{cps}$, a specification $A \in \mathcal{L}$ and $n_1, \ldots, n_{k-1} \in \mathbb{N}$ and checks whether $p_1(l_1)^{n_1} \cdots p_{k-1}(l_{k-1})^{n_{k-1}} p_k(l_k)^\omega \in \mathrm{L}(A)$. Here the n_i's are understood to be encoded in binary and we do not require them to satisfy the constraint of the path schema.

Since constrained path schemas are abstractions of path schemas used in [4], from this work we can show that runs from flat counter systems can be represented by a finite set of constrained path schemas as stated below.

Theorem 2. *Let at be a finite set of atomic propositions,* ag_n *be a finite set of atomic guards from* $\mathtt{G}(\mathtt{C}_n)$*, S be a flat counter system whose atomic propositions and atomic guards are from* $at \cup ag_n$ *and* $c_0 = \langle q_0, \boldsymbol{v}_0 \rangle$ *be an initial configuration. One can construct in exponential time a set X of constrained path schemas s.t.: (I) Each constrained path schema* $\mathtt{cps}$ *in X has an alphabet of the form* $\langle at, ag_n, \Sigma \rangle$ *(*Σ *may vary) and* $\mathtt{cps}$ *is of polynomial size. (II) Checking whether a constrained path schema belongs to X can be done in polynomial time. (III) For every run* ρ *from* c_0*, there is a constrained path schema* $\mathtt{cps}$ *in X and* $w \in \mathrm{L}(\mathtt{cps})$ *such that* $\rho \models w$*. (IV) For every constrained path schema* $\mathtt{cps}$ *in X and for every* $w \in \mathrm{L}(\mathtt{cps})$*, there is a run* ρ *from* c_0 *such that* $\rho \models w$.

In order to take advantage of Theorem 2 for the verification of flat counter systems, we need to introduce an additional property: $\mathcal{L}$ has the *nice subalphabet property* iff for all specifications $A \in \mathcal{L}$ over $\langle at, ag_n, \Sigma \rangle$ and for all constrained alphabets $\langle at, ag_n, \Sigma' \rangle$, one can build a specification A' over $\langle at, ag_n, \Sigma' \rangle$ in polynomial time in the sizes of A and $\langle at, ag_n, \Sigma' \rangle$ such that $\mathrm{L}(A) \cap (\Sigma')^\omega = \mathrm{L}(A')$. We need this property to build from A and a constraint path schema over $\langle at, ag_n, \Sigma' \rangle$, the specification A'. This property will also be used to transform a specification over $\langle at, ag_n, \Sigma \rangle$ into a specification over the finite alphabet Σ'.

Lemma 3. *BA, ABA,* μTL*, ETL, Past LTL have the nice subalphabet property.*

The abstract Algorithm 1 which performs the following steps (1) to (3) takes as input S, a configuration c_0 and $A \in \mathcal{L}$ and solves $\mathrm{MC}(\mathcal{L}, \mathcal{CFS})$: (1) Guess $\mathtt{cps}$

over $\langle at, ag_n, \Sigma' \rangle$ in X; (2) Build A' such that $\mathrm{L}(A) \cap (\Sigma')^\omega = \mathrm{L}(A')$; (3) Return $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(A') \neq \emptyset$. Thanks to Theorem 2, the first guess can be performed in polynomial time and with the nice subalphabet property, we can build A' in polynomial time too. This allows us to conclude the following lemma which is a consequence of the correctness of the above algorithm (see [5]).

Lemma 4. *If $\mathcal{L}$ has the nice subalphabet property and its intersection non-emptiness problem is in* NP*[resp.* PSPACE*], then* $\mathrm{MC}(\mathcal{L}, \mathcal{CFS})$ *is in* NP*[resp.* PSPACE*]*

We know that the membership problem for Past LTL is in PTIME and the intersection non-emptiness problem is in NP (as a consequence of [4, Theorem 3]). By Lemma 4, we are able to conclude the main result from [4]: $\mathrm{MC}(\mathrm{PastLTL}, \mathcal{CFS})$ is in NP. This is not surprising at all since in this paper we present a general method for different specification languages that rests on Theorem 2 (a consequence of technical developments from [4]).

4 Taming First-Order Logic and Flat Counter Systems

In this section, we consider first-order logic as a specification language. By Kamp's Theorem, first-order logic has the same expressive power as Past LTL and hence model-checking first-order logic over flat counter systems is decidable too [4]. However this does not provide us an optimal upper bound for the model-checking problem. In fact, it is known that the satisfiability problem for first-order logic formulae is non-elementary and consequently the translation into Past LTL leads to a significant blow-up in the size of the formula.

4.1 First-Order Logic in a Nutshell

For defining first-order logic formulae, we consider a countably infinite set of variables $\mathtt{Z}$ and a finite (unconstrained) alphabet Σ. The syntax of *first-order logic* over atomic propositions FO_Σ is then given by the following grammar: $\phi ::= a(\mathtt{z}) \mid S(\mathtt{z}, \mathtt{z}') \mid \mathtt{z} < \mathtt{z}' \mid \mathtt{z} = \mathtt{z}' \mid \neg\phi \mid \phi \wedge \phi' \mid \exists \mathtt{z}\ \phi(\mathtt{z})$ where $a \in \Sigma$ and $\mathtt{z}, \mathtt{z}' \in \mathtt{Z}$. For a formula ϕ, we will denote by $free(\phi)$ its set of free variables defined as usual. A formula with no free variable is called a *sentence*. As usual, we define the *quantifier height* $qh(\phi)$ of a formula ϕ as the maximum nesting depth of the operators $\exists$ in ϕ. Models for FO_Σ are ω-words over the alphabet Σ and variables are interpreted by positions in the word. A *position assignment* is a partial function $f : \mathtt{Z} \to \mathbb{N}$. Given a model $w \in \Sigma^\omega$, a FO_Σ formula ϕ and a position assignment f such that $f(\mathtt{z}) \in \mathbb{N}$ for every variable $\mathtt{z} \in free(\phi)$, the satisfaction relation $\models_f$ is defined as usual. Given a FO_Σ sentence ϕ, we write $w \models \phi$ when $w \models_f \phi$ for an arbitrary position assignment f. The language of ω-words w over Σ associated to a sentence ϕ is then $\mathcal{L}(\phi) = \{w \in \Sigma^\omega \mid w \models \phi\}$. For $n \in \mathbb{N}$, we define the equivalence relation $\approx_n$ between ω-words over Σ as: $w \approx_n w'$ when for every sentence ϕ with $qh(\phi) \leq n$, $w \models \phi$ iff $w' \models \phi$.

FO on CS. FO formulae interpreted over infinite runs of counter systems are defined as FO formulae over a finite alphabet except that atomic formulae of the form $a(\mathtt{z})$ are replaced by atomic formulae of the form $\mathtt{p}(\mathtt{z})$ or $\mathtt{g}(\mathtt{z})$ where $\mathtt{p}$ is an atomic formula or $\mathtt{g}$ is an atomic guard from $\mathtt{G}(\mathtt{C}_n)$. Hence, a formula ϕ built over atomic formulae from a finite set at of atomic propositions and from a finite set ag_n of atomic guards from $\mathtt{G}(\mathtt{C}_n)$ defines a specification for the constrained alphabet $\langle at, at_n, 2^{at \cup ag_n} \rangle$. Note that the alphabet can be of exponential size in the size of ϕ and $\mathtt{p}(\mathtt{z})$ actually corresponds to a disjunction $\bigvee_{\mathtt{p} \in a} a(\mathtt{z})$.

Lemma 5. *FO has the nice subalphabet property.*

We have taken time to properly define first-order logic for counter systems (whose models are runs of counter systems, see also Section 2.2) but below, we will mainly operate with FO_Σ over a standard (unconstrained) alphabet. Let us state our first result about FO_Σ which allows us to bound the number of times each loop is taken in a constrained path schema in order to satisfy a formula. We provide a stuttering theorem equivalent for FO_Σ formulas as is done in [4] for PLTL and in [12] for LTL. The lengthy proof of Theorem 6 uses Ehrenfeuch-Fraïssé game (see [5]).

Theorem 6 (Stuttering Theorem). *Let* $w = w_1 \mathbf{s}^M w_2, w' = w_1 \mathbf{s}^{M+1} w_2 \in \Sigma^\omega$ *such that* $N \geq 1$, $M > 2^{N+1}$ *and* $\mathbf{s} \in \Sigma^+$. *Then* $w \approx_N w'$.

4.2 Model-Checking Flat Counter Systems with FO

Let us characterize the complexity of $\mathrm{MC}(\mathrm{FO}, \mathcal{CFS})$. First, we will state the complexity of the intersection non-emptiness problem. Given a constrained path schema $\mathtt{cps}$ and a FO sentence ψ, Theorem 1 provides two polynomials $\mathtt{pol}_1$ and $\mathtt{pol}_2$ to represent succinctly the solutions of the guard in $\mathtt{cps}$. Theorem 6 allows us to bound the number of times loops are visited. Consequently, we can compute a value $f_{\mathrm{FO}}(\psi, \mathtt{cps})$ exponential in the size of ψ and $\mathtt{cps}$, as explained earlier, which allows us to find a witness for the intersection non-emptiness problem where each loop is taken a number of times smaller than $f_{\mathrm{FO}}(\psi, \mathtt{cps})$.

Lemma 7. *Let* $\mathtt{cps}$ *be a constrained path schema and* ψ *be a* FO_Σ *sentence. Then* $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(\psi)$ *is non-empty iff there is an* ω*-word in* $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(\psi)$ *in which each loop is taken at most* $2^{(qh(\psi)+2)+\mathtt{pol}_1(\mathrm{size}(\mathtt{cps}))+\mathtt{pol}_2(\mathrm{size}(\mathtt{cps}))}$ *times.*

Hence $f_{\mathrm{FO}}(\psi, \mathtt{cps})$ has the value $2^{(qh(\psi)+2)+(\mathtt{pol}_1+\mathtt{pol}_2)(\mathrm{size}(\mathtt{cps}))}$. Furthermore checking whether $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(\psi)$ is non-empty amounts to guess some $\mathbf{n} \in [0, 2^{(qh(\psi)+2)+\mathtt{pol}_1(\mathrm{size}(\mathtt{cps}))+\mathtt{pol}_2(\mathrm{size}(\mathtt{cps}))}]^{k-1}$ and verify whether $w = p_1(l_1)^{\mathbf{n}[1]} \cdots p_{k-1}(l_{k-1})^{\mathbf{n}[k-1]} p_k(l_k)^\omega \in \mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(\psi)$. Checking if $w \in \mathrm{L}(\mathtt{cps})$ can be done in polynomial time in $(qh(\psi)+2) + \mathtt{pol}_1(\mathrm{size}(\mathtt{cps})) + \mathtt{pol}_2(\mathrm{size}(\mathtt{cps}))$ (and therefore in polynomial time in $\mathrm{size}(\psi) + \mathrm{size}(\mathtt{cps})$) since this amounts to verify whether $\mathbf{n} \models \phi$. Checking whether $w \in \mathrm{L}(\psi)$ can be done in exponential space in $\mathrm{size}(\psi) + \mathrm{size}(\mathtt{cps})$ by using [15, Proposition 4.2]. Hence, this leads to a nondeterministic exponential space decision procedure for the intersection non-emptiness problem but it is possible to get down to nondeterministic polynomial

space using the succinct representation of constrained path schema as stated by Lemma 8 below for which the lower bound is deduced by the fact that model-checking ultimately periodic words with first-order logic is PSPACE-hard [15].

Lemma 8. *Membership problem with* FO_Σ *is* PSPACE*-complete.*

Note that the membership problem for FO is for unconstrained alphabet, but due to the nice subalphabet property of FO, the same holds for constrained alphabet since given a FO formula over $\langle at, ag_n, \Sigma\rangle$, we can build in polynomial time a FO formula over $\langle at, ag_n, \Sigma'\rangle$ from which we can build also in polynomial time a formula of $\mathrm{FO}_{\Sigma'}$ (where Σ' is for instance the alphabet labeling a constrained path schema). We can now state the main results concerning FO.

Theorem 9. *(I) The intersection non-emptiness problem with FO is* PSPACE*-complete. (II)* $\mathrm{MC}(\mathrm{FO}, \mathcal{CFS})$ *is* PSPACE*-complete. (III) Model-checking flat Kripke structures with FO is* PSPACE*-complete.*

Proof. (I) is a consequence of Lemma 7 and Lemma 8. We obtain (II) from (I) by applying Lemma 4 and Lemma 5. (III) is obtained by observing that flat Kripke structures form a subclass of flat counter systems. To obtain the lower bound, we use that model-checking ultimately periodic words with first-order logic is PSPACE-hard [15]. □

5 Taming Linear μ-calculus and Other Languages

We now consider several specification languages defining ω-regular properties on atomic propositions and arithmetical constraints. First, we deal with BA by establishing Theorem 10 and then deduce results for ABA, ETL and μTL.

Theorem 10. *Let* $\mathcal{B} = \langle Q, \Sigma, q_0, \Delta, F\rangle$ *be a Büchi automaton (with standard definition) and* $\mathtt{cps} = \langle p_1(l_1)^* \cdots p_{k-1}(l_{k-1})^* p_k(l_k)^\omega, \phi(\mathtt{x}_1, \ldots, \mathtt{x}_{k-1})\rangle$ *be a constrained path schema over* Σ*. We have* $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(\mathcal{B}) \neq \emptyset$ *iff there exists* $\boldsymbol{y} \in [0, 2^{\mathtt{pol}_1(\mathrm{size}(\mathtt{cps}))} + 2.\mathrm{card}(Q)^k \times 2^{\mathtt{pol}_1(\mathrm{size}(\mathtt{cps})) + \mathtt{pol}_2(\mathrm{size}(\mathtt{cps}))}]^{k-1}$ *such that* $p_1(l_1)^{\boldsymbol{y}[1]} \ldots p_{k-1}(l_{k-1})^{\boldsymbol{y}[k-1]} p_k l_k^\omega \in \mathrm{L}(\mathcal{B}) \cap \mathrm{L}(\mathtt{cps})$ *(*$\mathtt{pol}_1$ *and* $\mathtt{pol}_2$ *are from Theorem 1).*

Theorem 10 can be viewed as a pumping lemma involving an automaton and semilinear sets. Thanks to it we obtain an exponential bound for the map f_{BA} so that $f_{\mathrm{BA}}(\mathcal{B}, \mathtt{cps}) = 2^{\mathtt{pol}_1(\mathrm{size}(\mathtt{cps}))} + 2.\mathrm{card}(Q)^{\mathrm{size}(\mathtt{cps})} \times 2^{\mathtt{pol}_1(\mathrm{size}(\mathtt{cps})) + pol_2(\mathrm{size}(\mathtt{cps}))}$. So checking $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(\mathcal{B}) \neq \emptyset$ amounts to guess some $\mathbf{n} \in [0, 2^{\mathtt{pol}_1(\mathrm{size}(\mathtt{cps}))} + 2.\mathrm{card}(Q)^{\mathrm{size}(\mathtt{cps})} \times 2^{\mathtt{pol}_1(\mathrm{size}(\mathtt{cps})) + \mathtt{pol}_2(\mathrm{size}(\mathtt{cps}))}]^{k-1}$ and to verify whether the word $w = p_1(l_1)^{\mathbf{n}[1]} \cdots p_{k-1}(l_{k-1})^{\mathbf{n}[k-1]} p_k(l_k)^\omega \in \mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(\mathcal{B})$. Checking whether $w \in \mathrm{L}(\mathtt{cps})$ can be done in polynomial time in $\mathrm{size}(\mathcal{B}) + \mathrm{size}(\mathtt{cps})$ since this amounts to check $\mathbf{n} \models \phi$. Checking whether $w \in \mathrm{L}(\mathcal{B})$ can be also done in polynomial time by using the results from [15]. Indeed, w can be encoded in polynomial time as a pair of straight-line programs and by [15, Corollary 5.4] this can be done in polynomial time. So, the membership problem for Büchi automata is in PTIME. By using that BA has the nice subalphabet property and that we can create a polynomial size Büchi automata from a given BA specification and $\mathtt{cps}$, we get the following result.

Lemma 11. *The intersection non-emptiness problem with BA is* NP*-complete.*

Now, by Lemma 3, Lemma 4 and Lemma 11, we get the result below for which the lower bound is obtained from an easy reduction of SAT.

Theorem 12. $\mathrm{MC}(\mathrm{BA}, \mathcal{CFS})$ *is* NP*-complete.*

We are now ready to deal with ABA, ETL and linear μ-calculus. A language $\mathcal{L}$ has the *nice BA property* iff for every specification A from $\mathcal{L}$, we can build a Büchi automaton $\mathcal{B}_A$ such that $\mathrm{L}(A) = \mathrm{L}(\mathcal{B}_A)$, each state of $\mathcal{B}_A$ is of polynomial size, it can be checked if a state is initial [resp. accepting] in polynomial space and the transition relation can be decided in polynomial space too. So, given a language $\mathcal{L}$ having the nice BA property, a constrained path schema $\mathtt{cps}$ and a specification in $A \in \mathcal{L}$, if $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(A)$ is non-empty, then there is an ω-word in $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(A)$ such that each loop is taken at most a number of times bounded by $f_{\mathrm{BA}}(\mathcal{B}_A, \mathtt{cps})$. So $f_{\mathcal{L}}(A, \mathtt{cps})$ is obviously bounded by $f_{\mathrm{BA}}(\mathcal{B}_A, \mathtt{cps})$. Hence, checking whether $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(A)$ is non-empty amounts to guess some $\mathbf{n} \in [0, f_{\mathcal{L}}(A, \mathtt{cps})]^{k-1}$ and check whether $w = p_1(l_1)^{\mathbf{n}[1]} \cdots p_{k-1}(l_{k-1})^{\mathbf{n}[k-1]} p_k(l_k)^{\omega} \in \mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(A)$. Checking whether $w \in \mathrm{L}(\mathtt{cps})$ can be done in polynomial time in $\mathrm{size}(A) + \mathrm{size}(\mathtt{cps})$ since this amounts to check $\mathbf{n} \models \phi$. Checking whether $w \in \mathrm{L}(A)$ can be done in nondeterministic polynomial space by reading w while guessing an accepting run for $\mathcal{B}_A$. Actually, one guesses a state q from $\mathcal{B}_A$ and check whether the prefix $p_1(l_1)^{\mathbf{n}[1]} \cdots p_{k-1}(l_{k-1})^{\mathbf{n}[k-1]} p_k$ can reach it and then nonemptiness between $(l_k)^{\omega}$ and the Büchi automaton $\mathcal{B}_A^q$ in which q is an initial state is checked. Again, this can be done in nondeterministic polynomial space thanks to the nice BA property. We obtain the lemma below.

Lemma 13. *Membership problem and intersection non-emptiness problem for $\mathcal{L}$ having the nice BA property are in* PSpace*.*

Let us recall consequences of results from the literature. ETL has the nice BA property by [24], linear μ-calculus has the nice BA property by [23] and ABA has the nice BA property by [18]. Note that the results for ETL and ABA can be also obtained thanks to translations into linear μ-calculus. By Lemma 13, Lemma 4 and the above-mentioned results, we obtain the following results.

Theorem 14. $\mathrm{MC}(\mathrm{ABA}, \mathcal{CFS})$*,* $\mathrm{MC}(\mathrm{ETL}, \mathcal{CFS})$ *and* $\mathrm{MC}(\mu\mathrm{TL}, \mathcal{CFS})$ *are in* PSpace*.*

Note that for obtaining the PSpace upper bound, we use the same procedure for all the logics. Using that the emptiness problem for finite alternating automata over a single letter alphabet is PSpace-hard [8], we are also able to get lower bounds.

Theorem 15. *(I) The intersection non-emptiness problem for* ABA *[resp. μTL] is* PSpace*-hard. (II)* $\mathrm{MC}(\mathrm{ABA}, \mathcal{CFS})$ *and* $\mathrm{MC}(\mu\mathrm{TL}, \mathcal{CFS})$ *are* PSpace*-hard.*

According to the proof of Theorem 15 (see [5]), PSpace-hardness already holds for a fixed Kripke structure, that is actually a simple path schema. Hence, for linear μ-caluclus, there is a complexity gap between model-checking unconstrained

path schemas with two loops (in UP∩co-UP [9]) and model-checking unconstrained path schemas (Kripke structures) made of a single loop, which is in contrast to Past LTL for which model-checking unconstrained path schemas with a bounded number of loops is in PTIME [4, Theorem 9].

As an additional corollary, we can solve the global model-checking problem with existential Presburger formulae. The global model-checking consists in characterizing the set of initial configurations from which there exists a run satisfying a given specification. We knew that Presburger formulae exist for global model-checking [6] for Past LTL (and therefore for FO) but we can conclude that they are structurally simple and we provide an alternative proof. Moreover, the question has been open for μTL since the decidability status of $\mathrm{MC}(\mu\mathrm{TL}, \mathcal{CFS})$ has been only resolved in the present work.

Corollary 16. *Let $\mathcal{L}$ be a specification language among FO, BA, ABA, ETL or μTL. Given a flat counter system S, a control state q and a specification A in $\mathcal{L}$, one can effectively build an existential Presburger formula $\phi(\mathsf{z}_1, \ldots, \mathsf{z}_n)$ such that for all $\boldsymbol{v} \in \mathbb{N}^n$. $\boldsymbol{v} \models \phi$ iff there is a run ρ starting at $\langle q, \boldsymbol{v}\rangle$ verifying $\rho \models A$.*

6 Conclusion

We characterized the complexity of $\mathrm{MC}(\mathcal{L}, \mathcal{CFS})$ for prominent linear-time specification languages $\mathcal{L}$ whose letters are made of atomic propositions and linear constraints. We proved the PSPACE-completeness of the problem with linear μ-calculus (decidability was open), for alternating Büchi automata and also for FO. When specifications are expressed with Büchi automata, the problem is shown NP-complete. Global model-checking is also possible on flat counter systems with such specification languages. Even though the core of our work relies on small solutions of quantifier-free Presburger formulae, stuttering properties, automata-based approach and on-the-fly algorithms, our approach is designed to be generic. Not only this witnesses the robustness of our method but our complexity characterization justifies further why verification of flat counter systems can be at the core of methods for model-checking counter systems. Our main results are in the table below with useful comparisons ('Ult. periodic KS' stands for ultimately periodic Kripke structures namely a path followed by a loop).

	Flat counter systems	Kripke struct.	Flat Kripke struct.	Ult. periodic KS
μTL	PSPACE-C (Thm. 14)	PSPACE-C [23]	PSPACE-C (Thm. 14)	in UP∩co-UP [16]
ABA	PSPACE-C (Thm. 14)	PSPACE-C	PSPACE-C (Thm. 14)	in PTIME (see e.g. [11, p. 3])
ETL	in PSPACE (Thm. 14)	PSPACE-C [21]	in PSPACE [21]	in PTIME (see e.g. [19,11])
BA	NP-C (Thm.12)	in PTIME	in PTIME	in PTIME
FO	PSPACE-C (Thm. 9)	Non-el. [22]	PSPACE-C (Thm. 9)	PSPACE-C [15]
Past LTL	NP-C [4]	PSPACE-C [21]	NP-C [10,4]	PTIME [13]

References

1. Boigelot, B.: Symbolic methods for exploring infinite state spaces. PhD thesis, Université de Liège (1998)
2. 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)
3. Comon, H., Jurski, Y.: Multiple counter automata, safety analysis and PA. In: Vardi, M.Y. (ed.) CAV 1998. LNCS, vol. 1427, pp. 268–279. Springer, Heidelberg (1998)
4. 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)
5. Demri, S., Dhar, A.K., Sangnier, A.: On the complexity of verifying regular properties on flat counter systems (2013), `http://arxiv.org/abs/1304.6301`
6. Demri, S., Finkel, A., Goranko, V., van Drimmelen, G.: Model-checking CTL^* over flat Presburger counter systems. JANCL 20(4), 313–344 (2010)
7. Finkel, A., Leroux, J.: How to compose presburger-accelerations: Applications to broadcast protocols. In: Agrawal, M., Seth, A.K. (eds.) FSTTCS 2002. LNCS, vol. 2556, pp. 145–156. Springer, Heidelberg (2002)
8. Jančar, P., Sawa, Z.: A note on emptiness for alternating finite automata with a one-letter alphabet. IPL 104(5), 164–167 (2007)
9. Jurdziński, M.: Deciding the winner in parity games is in UP ∩ co-UP. IPL 68(3), 119–124 (1998)
10. Kuhtz, L., Finkbeiner, B.: Weak kripke structures and LTL. In: Katoen, J.-P., König, B. (eds.) CONCUR 2011. LNCS, vol. 6901, pp. 419–433. Springer, Heidelberg (2011)
11. Kupferman, O., Vardi, M.: Weak alternating automata are not that weak. ACM Transactions on Computational Logic 2(3), 408–429 (2001)
12. Kučera, A., Strejček, J.: The stuttering principle revisited. Acta Informatica 41(7-8), 415–434 (2005)
13. Laroussinie, F., Markey, N., Schnoebelen, P.: Temporal logic with forgettable past. In: LICS 2002, pp. 383–392. IEEE (2002)
14. Leroux, J., Sutre, G.: Flat counter systems are everywhere! In: Peled, D.A., Tsay, Y.-K. (eds.) ATVA 2005. LNCS, vol. 3707, pp. 489–503. Springer, Heidelberg (2005)
15. Markey, N., Schnoebelen, P.: Model checking a path. In: Amadio, R.M., Lugiez, D. (eds.) CONCUR 2003. LNCS, vol. 2761, pp. 251–265. Springer, Heidelberg (2003)
16. Markey, N., Schnoebelen, P.: Mu-calculus path checking. IPL 97(6) (2006)
17. Minsky, M.: Computation, Finite and Infinite Machines. Prentice Hall (1967)
18. Miyano, S., Hayashi, T.: Alternating finite automata on ω-words. Theor. Comput. Sci. 32, 321–330 (1984)
19. Piterman, N.: Extending temporal logic with ω-automata. Master's thesis, The Weizmann Institute of Science (2000)
20. Pottier, L.: Minimal Solutions of Linear Diophantine Systems: Bounds and Algorithms. In: Book, R.V. (ed.) RTA 1991. LNCS, vol. 488, pp. 162–173. Springer, Heidelberg (1991)
21. Sistla, A., Clarke, E.: The complexity of propositional linear temporal logic. JACM 32(3), 733–749 (1985)
22. Stockmeyer, L.J.: The complexity of decision problems in automata and logic. PhD thesis, MIT (1974)
23. Vardi, M.: A temporal fixpoint calculus. In: POPL 1988, pp. 250–259. ACM (1988)
24. Vardi, M., Wolper, P.: Reasoning about infinite computations. I&C 115 (1994)
25. Wolper, P.: Temporal logic can be more expressive. I&C 56, 72–99 (1983)