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 [19], 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 [15]. 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 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

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

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. [2,4,8]. 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 [3]). Decidability results for verifying safety and reachability properties on flat counter systems have been obtained in [4,8,3]. 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 [5] (NP-completeness already holds with flat Kripke structures [11]).

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 [5] 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 formalism (Corollary 16).

Due to lack of space, omitted proofs can be found in the technical appendix.

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 $\Sigma_{i=0}^{n} a_i \cdot \mathtt{x}_i \sim b$ 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*, $\mathbf{l} : Q \rightarrow 2^{\mathrm{AT}}$ 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 $TS(S) = \langle C, \rightarrow \rangle$ where $C = Q \times \mathbb{N}^n$ is the set of *configurations* and $\rightarrow \subseteq C \times \Delta \times C$ is the *transition relation* defined by: $\langle \langle q, \mathbf{v} \rangle, \delta, \langle q', \mathbf{v}' \rangle \rangle \in \rightarrow$ (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) [4,15,5]. 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 $\mathbf{l} : Q \rightarrow 2^{\mathrm{AT}}$. 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 $\rho := \langle q_0, \mathbf{v}_0\rangle \rightarrow \langle q_1, \mathbf{v}_1\rangle \cdots$ 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. $\mathtt{g} \in (ag_n \setminus a_i)$].

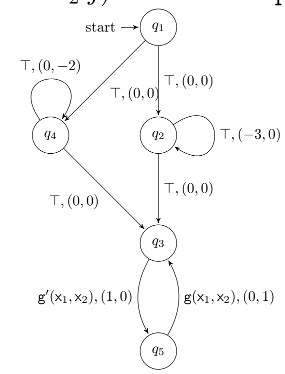
A *specification language* $\mathcal{L}$ over a constrained alphabet $\langle at, ag_n, \Sigma\rangle$ is a set of *specifications* A, each of it defining a set $\mathrm{L}(A)$ of ω-words over Σ. We will also sometimes consider specification languages over (unconstrained) standard finite alphabets (as 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 $\delta : Q \times 2^{at \cup ag_n} \rightarrow \mathbb{B}^+(Q)$ 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 $Q \times \Sigma \rightarrow \mathbb{B}^+(Q)$). 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.[28,21]), Past LTL (see e.g. [23]) and linear μ-calculus (or μTL), see e.g. [25]. 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 $\rho := \langle q_0, \mathbf{v}_0\rangle \rightarrow \langle q_1, \mathbf{v}_1\rangle \cdots$, 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 [5]). 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 $\mathbf{l}$ s.t. $\mathbf{l}(q_3) = \{\mathtt{p}, \mathtt{q}\}$ 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. $i \equiv_2 j$) satisfies that $\mathtt{p}$ holds and first counter is equal to 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: $\mu \mathsf{z}_1.(\mathtt{X}(\nu \mathsf{z}_2.(\mathsf{p} \wedge (\mathsf{x}_1 - \mathsf{x}_2 = 0) \wedge \mathtt{XX}\mathsf{z}_2) \vee \mathtt{X}\mathsf{z}_1)$. 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:

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

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

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 [5] 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 $\langle p_1(l_1)^* \cdots p_{k-1}(l_{k-1})^* p_k(l_k)^\omega, \phi(\mathsf{x}_1, \ldots, \mathsf{x}_{k-1})\rangle$ 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 $\phi(\mathsf{x}_1, \ldots, \mathsf{x}_{k-1}) \in \mathtt{G}(\mathtt{C}_{k-1})$. Each constrained path schema defines a language $\mathrm{L}(\mathtt{cps}) \subseteq \Sigma^\omega$ given by $\mathrm{L}(\mathtt{cps}) \stackrel{\text{def}}{=} \{p_1(l_1)^{n_1} \cdots p_{k-1}(l_{k-1})^{n_{k-1}} p_k(l_k)^\omega \ : \ \phi(n_1, \ldots, n_{k-1}) \text{ holds true}\}$. The size of $\mathtt{cps}$, written $\mathrm{size}(\mathtt{cps})$, is equal to $2k + \mathrm{len}(p_1 l_1 \cdots p_{k-1} l_{k-1} p_k l_k) + \mathrm{size}(\phi(\mathsf{x}_1, \ldots, \mathsf{x}_{k-1}))$. 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. *[22] 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 [5], 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 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

$L(\texttt{cps}) \cap L(A') \neq \emptyset$. 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 Appendix C).

Lemma 4. *If $\mathcal{L}$ has the nice subalphabet property and its intersection non-emptiness problem is in* NP *[resp.* PSPACE*], then* 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 [5, Theorem 3]). By Lemma 4, we are able to conclude the main result from [5]: MC(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 [5]).

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 [5]. 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 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: $\phi ::= a(\mathsf{z}) \mid S(\mathsf{z},\mathsf{z}') \mid \mathsf{z} < \mathsf{z}' \mid \mathsf{z} = \mathsf{z}' \mid \neg\phi \mid \phi \wedge \phi' \mid \exists \mathsf{z}\ \phi(\mathsf{z})$ where $a \in \Sigma$ and $\mathsf{z},\mathsf{z}' \in \mathsf{Z}$. 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 $f : \mathsf{Z} \to \mathbb{N}$. Given a model $w \in \Sigma^\omega$, a FO_Σ formula ϕ and a position assignment f such that $f(\mathsf{z}) \in \mathbb{N}$ for every variable $\mathsf{z} \in free(\phi)$, 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 [5] for PLTL and in [13] for LTL. The lengthy proof of Theorem 6 uses Ehrenfeuch-Fraïssé game (see Appendix E).

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 MC(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 [17, 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 [17].

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) applying Lemma 4 and Lemma 5. And (III) is obtained by observing that flat Kripke structures form a subclass of flat counter systems. Here again to obtain all the lower bound we use that model-checking ultimately periodic words with first-order logic is PSpace-hard [17]. □

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 [17]. Indeed, w can be encoded in polynomial time as a pair of straight-line programs and by [17, 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 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. MC(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 `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. Consequently we deduce the next lemma.

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 [26], linear μ-calculus has the nice BA property by [25] and ABA has the nice BA property by [20]. 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. MC(ABA,$\mathcal{CFS}$), MC(ETL,$\mathcal{CFS}$) *and* MC(μTL,$\mathcal{CFS}$) *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 [9], we are also able to get lower bounds.

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

According to the proof of Theorem 15 (Appendix K), 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 [10]) 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 [5, Theorem 9].

As an additional corollary, we can solve the global model-checking problem with existential Presburger formulae. The global model-checking consists in computing 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 MC(μTL,$\mathcal{CFS}$) 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 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 [25]	PSpace-C (Thm. 14)	in UP∩co-UP [18]
ABA	PSpace-C (Thm. 14)	PSpace-C	PSpace-C (Thm. 14)	in PTime (see e.g. [12, p. 3])
ETL	in PSpace (Thm. 14)	PSpace-C [23]	in PSpace [23]	in PTime (see e.g. [21,12])
BA	NP-C (Thm.12)	in PTime	in PTime	in PTime
FO	PSpace-C (Thm. 9)	Non-el. [24]	PSpace-C (Thm. 9)	PSpace-C [17]
Past LTL	NP-C [5]	PSpace-C [23]	NP-C [11,5]	PTime [14]

References

1. A. Arnold and D. Niwinski. *Rudiments of μ-calculus.* Elsevier, 2001.

2. B. Boigelot. *Symbolic methods for exploring infinite state spaces.* PhD thesis, Université de Liège, 1998.
3. M. Bozga, R. Iosif, and F. Konecný. Fast acceleration of ultimately periodic relations. In *CAV'10*, volume 6174 of *LNCS*, pages 227–242. Springer, 2009.
4. H. Comon and Y. Jurski. Multiple counter automata, safety analysis and PA. In *CAV'98*, volume 1427 of *LNCS*, pages 268–279. Springer, 1998.
5. 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. See also `http://arxiv.org/abs/1205.6584`.
6. S. Demri, A. Finkel, V. Goranko, and G. van Drimmelen. Model-checking CTL^* over flat Presburger counter systems. *JANCL*, 20(4):313–344, 2010.
7. K. Etessami and T. Wilke. An until hierarchy and other applications of an Ehrenfeucht-Fraïssé game for temporal logic. *I&C*, 160(1–2):88–108, 2000.
8. 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.
9. P. Jančar and Z. Sawa. A note on emptiness for alternating finite automata with a one-letter alphabet. *IPL*, 104(5):164–167, 2007.
10. M. Jurdziński. Deciding the winner in parity games is in UP $\cap$ co-UP. *IPL*, 68(3):119–124, 1998.
11. L. Kuhtz and B. Finkbeiner. Weak Kripke structures and LTL. In *CONCUR'11*, volume 6901 of *LNCS*, pages 419–433. Springer, 2011.
12. O. Kupferman and M. Vardi. Weak alternating automata are not that weak. *ACM Transactions on Computational Logic*, 2(3):408–429, 2001.
13. A. Kučera and J. Strejček. The stuttering principle revisited. *Acta Informatica*, 41(7–8):415–434, 2005.
14. F. Laroussinie, N. Markey, and P. Schnoebelen. Temporal logic with forgettable past. In *LICS'02*, pages 383–392. IEEE, 2002.
15. J. Leroux and G. Sutre. Flat counter systems are everywhere! In *ATVA'05*, volume 3707 of *LNCS*, pages 489–503. Springer, 2005.
16. L. Libkin. *Elements of Finite Model Theory.* Springer, 2004.
17. N. Markey and P. Schnoebelen. Model checking a path. In *CONCUR'03, Marseille, France*, volume 2761 of *LNCS*, pages 251–261. Springer, 2003.
18. N. Markey and P. Schnoebelen. Mu-calculus path checking. *IPL*, 97(6):225–230, 2006.
19. M. Minsky. *Computation, Finite and Infinite Machines.* Prentice Hall, 1967.
20. S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theor. Comput. Sci.*, 32:321–330, 1984.
21. N. Piterman. Extending temporal logic with ω-automata. Master's thesis, The Weizmann Institute of Science, 2000.
22. L. Pottier. Minimal Solutions of Linear Diophantine Systems: Bounds and Algorithms. In *RTA'91*, pages 162–173. Springer, 1991.
23. A. Sistla and E. Clarke. The complexity of propositional linear temporal logic. *JACM*, 32(3):733–749, 1985.
24. L. J. Stockmeyer. *The complexity of decision problems in automata and logic.* PhD thesis, MIT, 1974.
25. M. Vardi. A temporal fixpoint calculus. In *POPL'88*, pages 250–259. ACM, 1988.
26. M. Vardi and P. Wolper. Reasoning about infinite computations. *I&C*, 115:1–37, 1994.
27. I. Walukiewicz. Automata and logic, 2001. Lecture notes.
28. P. Wolper. Temporal logic can be more expressive. *I&C*, 56:72–99, 1983.

A Proof of Theorem 2

Below, we provide the main steps of the proof, details can be found in [5].

Proof. (sketch) Let us explain how to build the set X.

1. Given a flat counter system S and a state q from c, there is at most an exponential number of minimal path schemas starting at q in the sense of [5, Lemma 4]. Let Y_1 be this set of minimal path schemas.
2. For each path schema P in Y_1, there is a set of path schemas Y_P such that the path schemas in Y_P have no disjunctions in guards and satisfaction of guards can be concluded from the states, see [5, Theorem 14]. Let Y_2 be this set of unfolded path schemas and it is of cardinality at most exponential.
3. Following [5, Lemma 12], every path schema from Y_2 is equivalent to a constrained path schema. The set X is precisely the set of constrained path schemas obtained from all the unfolded path schemas from Y_2.

Completeness of the set X is a consequence of [5, Lemma 12] and [5, Theorem 14(4–6)]. Satisfaction of the size constraints is a consequence of [5, Lemma 12] and [5, Theorem 14(2–3)]. □

B Proof of Lemma 3

Proof. Let $A = \langle Q, E, q_0, F\rangle$ be a specification in BA. over the alphabet $\langle at, ag_n, \Sigma\rangle$ and $\Sigma' \subseteq \Sigma$. The specification $A' = \langle Q, E', q_0, F\rangle$ such that $\mathrm{L}(A') = \mathrm{L}(A) \cap (\Sigma')^\omega$ is defined as follows: for every $q \xrightarrow{\psi} q' \in E$, we include in E' the edge $q \xrightarrow{(\bigvee_{a \in \Sigma'} \ \psi_a) \wedge \psi} q'$ where φ_a is defined as a conjunction made of positive literals from a and negative literals from $(at \cup ag_n) \setminus a$. A similar transformation can be performed with specifications in ABA.

Let ϕ be a formula for $\mathcal{L}$ among linear μ-calculus, ETL or Past LTL built over atomic formulae in $at \cup ag_n$ and $\langle at, ag_n, \Sigma'\rangle$ be a constrained alphabet. The formulae ϕ' such that $\mathrm{L}(\phi') = \mathrm{L}(\phi) \cap (\Sigma')^\omega$ is obtained from ϕ by replacing every atomic formula ψ by $\bigvee_{\{a \in \Sigma' \mid \psi \in a\}} \varphi_a$. □

C Correctness of Algorithm 1

Proof. First assume there exists a run ρ of S starting at c_0 such that $\rho \models A$. By Theorem 2, there is a constrained path schema $\mathtt{cps}$ with an alphabet of the form $\langle at, ag_n, \Sigma'\rangle$ in X and $w \in \mathrm{L}(\mathtt{cps})$ such that $\rho \models w$. Consequently we deduce that $w \in \mathrm{L}(A)$ and that $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(A) \neq \emptyset$. Since $\mathrm{L}(\mathtt{cps}) \subseteq (\Sigma')^\omega$ and since $\mathrm{L}(A) \cap (\Sigma')^\omega = \mathrm{L}(A')$, we deduce that $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(A') \neq \emptyset$. Hence the Algorithm has an accepting run.

Now if the Algorithm 1 has an accepting run, we deduce that there exists a constrained path schema $\mathtt{cps}$ with an alphabet of the form $\langle at, ag_n, \Sigma'\rangle$ in X such that there exists a word w in $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(A')$. Using the nice subalphabet property we deduce that $w \in \mathrm{L}(A)$ and by the last point of Theorem 2, we know that there exists a run ρ from S starting at c_0 such that $\rho \models w$. This allows us to conclude that $\rho \models A$. □

D Proof of Lemma 5

Proof. Consider a FO formula ϕ that defines a specification over the constrained alphabet $\langle at, ag_n, \Sigma\rangle$ with $\Sigma = 2^{at \cup ag_n}$. Consider a subalphabet $\Sigma' \subseteq \Sigma$. Let ϕ'' be the formula obtained from ϕ by replacing every occurrence of $\mathtt{p}(\mathtt{z})$ by $\bigvee_{\{a \in \Sigma' | \mathtt{p} \in a\}} a(\mathtt{z})$ and every occurrence of $\mathtt{g}(\mathtt{z})$ is replaced by $\bigvee_{\{a \in \Sigma' | \mathtt{g} \in a\}} a(\mathtt{z})$. It is easy to see that, by construction, $\mathrm{L}(\phi'') = \mathrm{L}(\phi') \cap (\Sigma)^\omega$ □

E EF Games and Proof of Theorem 6

E.1 Ehrenfeucht-Fraïssé Games

Ehrenfeucht-Fraïssé (EF) game is a well known technique to determine whether two structures are equivalent with respect to a set of formulae. We recall here the definition of a EF game adapted to our context. Given $N \in \mathbb{N}$ and two ω-words w, w' over Σ, the main idea of the corresponding EF game is that two players, the Spoiler and the Duplicator, plays in a turn based manner. The Spoiler begins by choosing a word between w and w' and a position in this word, then the Duplicator aims at finding a position in the other word which is *similar* and this during N rounds. At the end, the Duplicator wins if the set of chosen positions respects some isomorphism. We now move to the formal definition of such a game.

Let w and w' be two ω-words over Σ. We define a play as a finite sequence of triples $(p_1, a_1, b_1)(p_2, a_2, b_2)\cdots(p_i, a_i, b_i)$ in $(\{0,1\} \times \mathbb{N}^2)^*$ where for each triple the first element describes which word has been chosen by the Spoiler (0 for the word w), then the second element corresponds to the position chosen in w and the third element the position chosen in w' by the Spoiler or the Duplicator according to the word chosen by the Spoiler. For instance if $p_1 = 1$, this means that at the first turn Spoiler has chosen the position b_1 in w' and Duplicator the position a_1 in w. A play of size $i \in \mathbb{N}$ is called an i-round play (a 0-round play being an empty sequence). A strategy for the Spoiler is a mapping $\sigma_S : (\{0,1\} \times \mathbb{N}^2)^* \to \{0,1\} \times \mathbb{N}$ which takes as input a play and outputs 0 or 1 for words w or w' respectively and a position in the word. Similarly, a strategy for the Duplicator is a mapping $\sigma_D : (\{0,1\} \times \mathbb{N}^2)^* \times (\{0,1\} \times \mathbb{N}) \to \mathbb{N}$ with the difference being that Duplicator takes into account the position played by the Spoiler in the current round. For all $i \in \mathbb{N}$, a strategy σ_S for the Spoiler and a strategy σ_D for the Duplicator, the i-round play over w and w' following σ_S and σ_D is defined inductively as follows: $\Pi_i^{\sigma_S,\sigma_D}(w, w') = \Pi_{i-1}^{\sigma_S,\sigma_D}(w, w')(p, a, b)$ where if $p = 0$, $(0, a) = \sigma_D(\Pi_{i-1}^{\sigma_S,\sigma_D}(w, w'))$ and $b = \sigma_S(\Pi_{i-1}^{\sigma_S,\sigma_D}(w, w'), (0, a))$ and if $p = 1$, $(1, b) = \sigma_D(\Pi_{i-1}^{\sigma_S,\sigma_D}(w, w'))$ and $a = \sigma_S(\Pi_{i-1}^{\sigma_S,\sigma_D}(w, w'), (1, b))$.

For $N \in \mathbb{N}$, a N-round play $(p_1, a_1, b_1)(p_2, a_2, b_2) \cdots (p_N, a_N, b_N)$ over w and w' is winning for Duplicator iff the following conditions are satisfied for all $i, j \in [1, N]$:

- $a_i = a_j$ iff $b_i = b_j$,
- $a_i + 1 = a_j$ iff $b_i + 1 = b_j$,
- $a_i < a_j$ iff $b_i < b_j$,
- $w(a_i) = w'(b_i)$.

A N-round EF game over the ω-words w, w', denoted as $EF_N(w, w')$, is said to be winning if there exists a strategy σ_D for Duplicator such that for all strategies σ_S of spoiler, the play $\Pi_N^{\sigma_S, \sigma_D}(w, w')$ is winning for Duplicator. We write $w \equiv_N w'$ iff the game $EF_N(w, w')$ is winning. Theorem 17 below states that two ω-words, the N-round game is winning iff these two ω-words satisfy the same set of first-order formulae of quantifier height smaller than N.

Theorem 17 (EF Theorem, see e.g. [16]). *For any two ω-words w, w' over Σ, $w \equiv_N w'$ iff $w \approx_N w'$.*

We will use EF games for FO_Σ formulae to prove a *stuttering theorem* which will allow us to bound the number of times each loop needs to be taken in a path schema in order to satisfy a FO_Σ formula. Note that in [7], EF games have been introduced for the specific case of LTL specifications here also to show some small model properties.

E.2 Stuttering Theorem for FO_Σ

In this section, we prove that if in an ω-sequence w, a subword $\mathbf{s}$ is repeated consecutively a large number of times, then this ω-word and other ω-words obtained by removing some of the repetitions of $\mathbf{s}$ satisfy the same set of FO_Σ sentences, this is what we call the stuttering theorem for FO_Σ. Such a result will allow us to bound the repetition of iteration of loops in path schema and thus to obtain a model-checking algorithm for the logic FO_Σ optimal in complexity. In order to prove the stuttering theorem, we will use EF games.

In the sequel we consider a natural $N \geq 1$ and two ω-words over Σ of the following form $w = w_1 \mathbf{s}^M w_2, w' = w_1 \mathbf{s}^{M+1} w_2 \in \Sigma^\omega$ with $M > 2^{N+1}$, $w_1 \in \Sigma^*, \mathbf{s} \in \Sigma^+$ and $w_2 \in \Sigma^\omega$. We will now show that the game $EF_N(w, w')$ is winning. The strategy for Duplicator will work as follows: at the i-th round (for $i \leq N$), if the point chosen by the Spoiler is close to another previously chosen position then the Duplicator will choose a point in the other word at the exact same distance from the corresponding position and if the point is far from any other position then in the other word the Duplicator will chose a position also far away from any other position.

Before providing a winning strategy for the Duplicator we define some invariants on any i-round play (with $i \leq N$) that will be maintained by the Duplicator's strategy. In order to define this invariant and the Duplicator's strategy, let introduce a few notations:

- $a_{-3} = b_{-3} = 0$; $a_{-2} = b_{-2} = \text{len}(w_1)$;
- $a_{-1} = \text{len}(w_1\mathbf{s}^M)$ and $b_{-1} = \text{len}(w_1\mathbf{s}^{M+1})$;
- $a_0 = b_0 = \omega$.

We extend the substraction and addition operations in order to deal with $\mathbb{N} \cup \{-\omega, \omega\}$ such that: $\alpha - \omega = -\omega$, $\omega - \alpha = \omega$ and $\omega + \alpha = \omega$ if $\alpha \in \mathbb{N}$ (no need to define the other cases for what follows). The relation $<$ on $\mathbb{N} \cup \{-\omega, \omega\}$ is extended in the obvious way. Given a i-round play $\Pi_i = (p_1, a_1, b_1)(p_2, a_2, b_2) \cdots (p_i, a_i, b_i)$, we say that Π_i respects the invariant $\mathfrak{I}$ iff the following conditions are satisfied for all $j, k \in [-3, i]$:

1. $a_j \leq a_k$ iff $b_j \leq b_k$,
2. $\mid a_j - a_k \mid < 2^{N+1-i}\text{len}(\mathbf{s})$ iff $\mid b_j - b_k \mid < 2^{N+1-i}\text{len}(\mathbf{s})$,
3. $\mid a_j - a_k \mid < 2^{N+1-i}\text{len}(\mathbf{s})$ implies $a_j - a_k = b_j - b_k$,
4. $a_j \leq a_{-2}$ or $b_j \leq b_{-2}$ implies $b_j = a_j$,
5. $a_j \geq a_{-1}$ or $b_j \geq b_{-1}$ implies $b_j = a_j + \text{len}(\mathbf{s})$,
6. $a_{-2} < a_j < a_{-1}$ or $b_{-2} < b_j < b_{-1}$ implies $\mid a_j - b_j \mid = 0 \bmod \text{len}(\mathbf{s})$.

First we remark the invariant $\mathfrak{I}$ is a sufficient condition for a play to be winning as stated by the following lemma.

Lemma 18. *If a N-round play over w and w' respects $\mathfrak{I}$, then it is a winning play for the Duplicator.*

Proof. Let $(p_1, a_1, b_1)(p_2, a_2, b_2) \cdots (p_N, a_N, b_N)$ be a N-round play over w and w' respecting $\mathfrak{I}$. Let $i, j \in [1, N]$. It is easy to see that satisfaction of $\mathfrak{I}$ implies that $a_i = a_j$ iff $b_i = b_j$, $a_i < a_j$ iff $b_i < b_j$, and $a_i + 1 = a_j$ iff $b_i + 1 = b_j$. Moreover, Condition $\mathfrak{I}$(4–6) obviously guarantees that $w(a_j) = w'(b_j)$. □

Given an $(i-1)$-round play $\Pi_{i-1} = (p_1, a_1, b_1)(p_2, a_2, b_2) \cdots (p_{i-1}, a_{i-1}, b_{i-1})$ and $a_i \in \mathbb{N}$ such that $a_i \notin \{a_{-3}, a_{-2}, \ldots, a_{i-2}, a_{i-1}\}$, we define $\mathit{left}(a_i) = \max(a_k \mid k \in [-3, i-1] \text{ and } a_k < a_i)$ and $\mathit{right}(a_i) = \min(a_k \mid k \in [-3, i-1] \text{ and } a_i < a_k)$ (i.e. $\mathit{left}(a_i)$ and $\mathit{right}(a_i)$ are the closest neighbor of a_i). We define similarly $\mathit{left}(b_i)$ and $\mathit{right}(b_i)$.

We define now a strategy $\hat{\sigma}_D$ for the Duplicator that respects at each round the invariant $\mathfrak{I}$ and this no matter what the Spoiler plays. By Lemma 18, we can conclude that this strategy is winning for the Duplicator. Let $i \in [1, N]$ and $\Pi_{i-1} = (p_1, a_1, b_1)(p_2, a_2, b_2) \cdots (p_{i-1}, a_{i-1}, b_{i-1})$ be a $(i-1)$-round play. First, we define $b_i = \sigma_D(\Pi_{i-1}, \langle 0, a_i \rangle)$ that is what Duplicator answers if the Spoiler chooses position a_i in the ω-word w. We have $b_i = \hat{\sigma}_D(\Pi_{i-1}, \langle 0, a_i \rangle)$ defined as follows:

- If $a_i = a_j$ for some $j \in [-3, i-1]$, then $b_i \stackrel{\text{def}}{=} b_j$;
- Otherwise, let $a_l = \mathit{left}(a_i)$ and $a_r = \mathit{right}(a_i)$:
  - If $a_i - a_l \leq a_r - a_i$, we have $b_i \stackrel{\text{def}}{=} b_l + (a_i - a_l)$
  - If $a_r - a_i < a_i - a_l$, we have $b_i \stackrel{\text{def}}{=} b_r - (a_r - a_i)$

Similarly we have $a_i = \hat{\sigma}_D(\Pi_{i-1}, \langle 1, b_i \rangle)$ defined as follows:

– If $b_i = b_j$ for some $j \in [-3, i-1]$, then $a_i \stackrel{\text{def}}{=} a_j$;
– Otherwise, let $b_l = \mathit{left}(b_i)$ and $b_r = \mathit{right}(b_i)$:
  - If $b_i - b_l \leq b_r - b_i$, we have $a_i \stackrel{\text{def}}{=} a_l + (b_i - b_l)$
  - If $b_r - b_i < b_i - b_l$, we have $a_i \stackrel{\text{def}}{=} a_r - (b_r - b_i)$

Lemma 19. *For any Spoiler's strategy σ_S and for all $i \in [0, N]$, we have that $\Pi_i^{\sigma_S, \hat{\sigma}_D}(w, w')$ respects $\mathfrak{I}$.*

Proof. The proof proceeds by induction on i. The base case for $i = 0$ is obvious since the empty play respects $\mathfrak{I}$. However, we need to use the fact that $M > 2^{N+1}$ (otherwise condition $\mathfrak{I}$.2 might not hold).

Let σ_S be a Spoiler's strategy and for every $i \in [1, N-1]$, we assume that $\Pi_{i-1}^{\sigma_S, \hat{\sigma}_D}(w, w')$ respects $\mathfrak{I}$. Suppose that $\sigma_S(\Pi_{i-1}^{\sigma_S, \hat{\sigma}_D}(w, w')) = \langle 0, a_i \rangle$ and let $b_i = \hat{\sigma}_D(\Pi_{i-1}^{\sigma_S, \hat{\sigma}_D}(w, w'), \langle 0, a_i \rangle)$.
(1.) Let $j, k \in [-3, i]$. If $j, k \in [-3, i-1]$, then by the induction hypothesis, $a_j \leq a_k$ iff $b_j \leq b_k$. Otherwise, let suppose $j = i$ and $k \neq i$ (only remaining interesting case). If $a_i = a_{j'}$ for some $j' \in [-3, i-1]$, then $b_i = b_{j'}$ and therefore $a_i \leq a_k$ iff $b_i \leq b_k$ by the induction hypothesis. Otherwise, $a_l < a_i < a_r$ and $b_l < b_i < b_r$ which entails that $a_i \leq a_k$ iff $b_i \leq b_k$.
(4.) The case $j \in [-3, i-1]$ is immediate from the induction hypothesis. Now, suppose that $a_{-3} \leq a_i \leq a_{-2}$. If $a_i = a_j$ for some $j \in [-3, i-1]$, then $b_i = b_j$ and $a_{-3} \leq a_j \leq a_{-2}$. By induction hypothesis, $b_i = b_j = a_j = a_i$. Otherwise, $a_l < a_i < a_r$ and $a_l = b_l$ and $a_r = b_r$ by induction hypothesis. Either $a_i - a_l \leq a_r - a_i$ or $a_r - a_i < a_i - a_l$ implies that $b_i = a_i$.
(5.) The case $j \in [-3, i-1]$ is immediate from the induction hypothesis. Now, suppose that $a_{-1} \leq a_i$. If $a_i = a_j$ for some $j \in [-3, i-1]$, then $b_i = b_j$ and $a_{-1} \leq a_j$. By induction hypothesis, $b_i = b_j = a_j + \mathrm{len}(\mathbf{s}) = a_i + \mathrm{len}(\mathbf{s})$. Otherwise, $a_l < a_i$ and $b_l = a_l + \mathrm{len}(\mathbf{s})$ by induction hypothesis. Since $a_r = \omega$, we have $b_i = b_l + (a_i - a_l) = a_i + \mathrm{len}(\mathbf{s})$.
(6.) The case $j \in [-3, i-1]$ is immediate from the induction hypothesis. Now, let us deal with $j = i$. Satisfaction of (4.) and (5.) implies that $a_{-2} < a_i < a_{-1}$ iff $b_{-2} < b_i < b_{-1}$. Suppose that $a_{-2} < a_i < a_{-1}$. So, $a_l < a_i < a_r$ and by induction hypothesis $\mid a_l - b_l \mid = 0 \bmod \mathrm{len}(\mathbf{s})$ and $\mid a_r - b_r \mid = 0 \bmod \mathrm{len}(\mathbf{s})$. If $a_i - a_l \leq a_r - a_i$, then $b_i = b_l + (a_i - a_l)$ and $\mid a_i - b_i \mid = \mid a_l - b_l \mid$, whence $\mid a_i - b_i \mid = 0 \bmod \mathrm{len}(\mathbf{s})$. Similarly, if $a_r - a_i < a_i - a_l$, then $b_i = b_r - (a_r - a_i)$ and $\mid a_i - b_i \mid = \mid a_r - b_r \mid$, whence $\mid a_i - b_i \mid = 0 \bmod \mathrm{len}(\mathbf{s})$.
(2.–3.) Let $j, k \in [-3, i]$. If $j, k \in [-3, i-1]$, then by the induction hypothesis, it is easy to verify that

– $\mid a_j - a_k \mid < 2^{N+1-i}\mathrm{len}(\mathbf{s})$ iff $\mid b_j - b_k \mid < 2^{N+1-i}\mathrm{len}(\mathbf{s})$,
– $\mid a_j - a_k \mid < 2^{N+1-i}\mathrm{len}(\mathbf{s})$ implies $a_j - a_k = b_j - b_k$.

Indeed, it is a consequence of the stronger properties below satisfied by induction hypothesis:

– $\mid a_j - a_k \mid < 2^{N+2-i}\mathrm{len}(\mathbf{s})$ iff $\mid b_j - b_k \mid < 2^{N+2-i}\mathrm{len}(\mathbf{s})$,
– $\mid a_j - a_k \mid < 2^{N+2-i}\mathrm{len}(\mathbf{s})$ implies $a_j - a_k = b_j - b_k$.

Otherwise, let suppose $j = i$ and $k \neq i$ (only remaining interesting case). If $a_i = a_{j'}$ for some $j' \in [-3, i-1]$, then by the induction hypothesis, we have

- $\mid a_{j'} - a_k \mid < 2^{N+2-i}\text{len}(\mathbf{s})$ iff $\mid b_{j'} - b_k \mid < 2^{N+2-i}\text{len}(\mathbf{s})$,
- $\mid a_{j'} - a_k \mid < 2^{N+2-i}\text{len}(\mathbf{s})$ implies $a_{j'} - a_k = b_{j'} - b_k$.

Again, this implies that

(I) $\mid a_i - a_k \mid < 2^{N+1-i}\text{len}(\mathbf{s})$ iff $\mid b_i - b_k \mid < 2^{N+1-i}\text{len}(\mathbf{s})$,
(II) $\mid a_i - a_k \mid < 2^{N+1-i}\text{len}(\mathbf{s})$ implies $a_i - a_k = b_i - b_k$.

Now, suppose that there is no $j' \in [-3, i-1]$ such that $a_i = a_{j'}$.
Case 1: $a_i - a_l \leq a_r - a_i$ and $b_i = b_l + (a_i - a_l)$.
Case 1.1: $a_k \leq a_l$.

- $a_k = a_l$: $\mid a_i - a_k \mid = \mid b_i - b_k \mid$ and therefore (I)-(II) holds.
- $a_k < a_l$: If $\mid a_l - a_k \mid \geq 2^{N+1-i}\text{len}(\mathbf{s})$, by induction hypothesis $\mid b_l - b_k \mid \geq 2^{N+1-i}\text{len}(\mathbf{s})$ and $b_k < b_l$. So, $\mid a_i - a_k \mid \geq 2^{N+1-i}\text{len}(\mathbf{s})$ and $\mid b_i - b_k \mid \geq 2^{N+1-i}$.
  If $\mid a_i - a_l \mid \geq 2^{N+1-i}\text{len}(\mathbf{s})$, by definition of b_l, $\mid a_i - a_l \mid = \mid b_i - b_l \mid \geq 2^{N+1-i}\text{len}(\mathbf{s})$ and $b_k < b_l$. So, $\mid a_i - a_k \mid \geq 2^{N+1-i}\text{len}(\mathbf{s})$ and $\mid b_i - b_k \mid \geq 2^{N+1-i}$.
  If $\mid a_l - a_k \mid \leq 2^{N+1-i}\text{len}(\mathbf{s})$ and $\mid a_i - a_l \mid \leq 2^{N+1-i}\text{len}(\mathbf{s})$, then by induction hypothesis $\mid b_l - b_k \mid = \mid a_l - a_k \mid$, $\mid a_i - a_l \mid = \mid b_i - b_l \mid$, $a_k < a_l < a_i$ and $b_k < b_l < b_i$. So $\mid a_i - a_k \mid = \mid b_i - b_k \mid$, whence (I)–(II) holds.

Case 1.2: $a_r \leq a_k$.

- $\mid a_k - a_r \mid \geq 2^{N+1-i}\text{len}(\mathbf{s})$: By induction hypothesis, $\mid b_k - b_r \mid \geq 2^{N+1-i}\text{len}(\mathbf{s})$. So, $\mid a_k - a_i \mid \geq 2^{N+1-i}\text{len}(\mathbf{s})$ and $\mid b_k - b_i \mid \geq 2^{N+1-i}\text{len}(\mathbf{s})$ since $a_i < a_r \leq a_k$ and $b_i < b_r \leq b_k$.
- $\mid a_k - a_r \mid \leq 2^{N+1-i}\text{len}(\mathbf{s})$: By induction hypothesis, $\mid b_k - b_r \mid = \mid a_k - a_r \mid$.
  Case 1.2.1. $\mid a_r - a_l \mid \leq 2^{N+2-i}\text{len}(\mathbf{s})$. By induction hypothesis, $\mid a_r - a_l \mid = \mid b_r - b_l \mid$ and therefore $\mid b_r - b_i \mid = \mid a_r - a_i \mid$. Whence, $\mid b_k - b_i \mid = \mid a_k - a_i \mid$ so (I)-(II) holds.
  Case 1.2.2 $\mid a_r - a_l \mid \geq 2^{N+2-i}\text{len}(\mathbf{s})$. By induction hypothesis, $\mid b_r - b_l \mid \geq 2^{N+2-i}\text{len}(\mathbf{s})$. Moreover, since $a_i - a_l \leq a_r - a_i$, $a_r - a_i \geq 2^{N+1-i}\text{len}(\mathbf{s})$. Since $b_i - b_l = a_i - b_l$, we have $b_r - b_i \geq 2^{N+1-i}\text{len}(\mathbf{s})$ too. So, $a_k - a_i \geq 2^{N+1-i}\text{len}(\mathbf{s})$ and $b_k - b_i \geq 2^{N+1-i}\text{len}(\mathbf{s})$, which guarantees (I)–(II).

Case 2: $a_r - a_i < a_i - a_l$ and $b_i = b_r - (a_r - a_i)$.
Case 2.1: $a_k \geq a_r$. Similar to Case 1.1 by replacing a_l by a_r, b_l by b_r and, by permuting '<' by '>' and '≤' by '≥' about positions.
Case 2.2: $a_k \leq a_l$. Similar to Case 1.2 by replacing a_r by a_l, b_r by b_l and, by permuting '<' by '>' and '≤' by '≥' about positions. □

Using Lemma 18 and 19, we deduce that Duplicator has a winning strategy against any strategy of the Spoiler in $EF_N(w, w')$, so by Theorem 17, we can conclude **Theorem 6 [Stuttering Theorem]**.

F Proof of Lemma 7

Proof. Let $\texttt{cps} = \langle p_1(l_1)^* \cdots p_{k-1}(l_{k-1})^* p_k(l_k)^\omega, \phi(\mathsf{x}_1, \ldots, \mathsf{x}_{k-1})\rangle$ be a constrained path schema and ψ be a first-order sentence. Suppose that

$$p_1(l_1)^{\mathbf{n}[1]} \cdots p_{k-1}(l_{k-1})^{\mathbf{n}[k-1]} p_k(l_k)^\omega \in \mathrm{L}(\texttt{cps}) \cap \mathrm{L}(\psi).$$

Let $B \subseteq [0, 2^{p_1(\mathrm{size}(\texttt{cps}))}]^{k-1}$ and $\mathbf{P}_1, \ldots, \mathbf{P}_\alpha \in [0, 2^{p_1(\mathrm{size}(\texttt{cps}))}]^{k-1}$ defined for the guard ϕ following Theorem 1. Since $\mathbf{n} \models \phi$, there are $\mathbf{b} \in B$ and $\mathbf{a} \in \mathbb{N}^\alpha$ such that $\mathbf{n} = \mathbf{b} + \mathbf{a}[1]\mathbf{P}_1 + \cdots + \mathbf{a}[\alpha]\mathbf{P}_\alpha$. Let $\mathbf{a}' \in \mathbb{N}^\alpha$ defined from $\mathbf{a}$ such that $\mathbf{a}'[i] = \mathbf{a}[i]$ if $\mathbf{a}[i] \leq 2^{qh(\psi)+1} + 1$ otherwise $\mathbf{a}'[i] = 2^{qh(\psi)+1} + 1$. Note that $\mathbf{n}' = \mathbf{b} + \mathbf{a}'[1]\mathbf{P}_1 + \cdots + \mathbf{a}'[\alpha]\mathbf{P}_\alpha$ still satisfies ϕ and for every loop $i \in [1, k-1]$, $\mathbf{n}[i] > 2^{qh(\psi)+1}$ iff $\mathbf{n}'[i] > 2^{qh(\psi)+1}$. By Theorem 6, $p_1(l_1)^{\mathbf{n}'[1]} \cdots p_{k-1}(l_{k-1})^{\mathbf{n}'[k-1]} p_k(l_k)^\omega \in \mathrm{L}(\psi)$.

Now, let us bound the values in $\mathbf{n}'$.

- There are at most $2^{p_2(\mathrm{size}(\texttt{cps}))}$ periods.
- Each basis or period has values in $[0, 2^{p_1(\mathrm{size}(\texttt{cps}))}]$.
- Each period in $\mathbf{n}'$ is taken at most $2^{qh(\psi)+1} + 1$ times.

Consequently, each $\mathbf{n}'[i]$ is bounded by

$$2^{p_1(\mathrm{size}(\texttt{cps}))} + (2^{qh(\psi)+1} + 1)2^{p_2(\mathrm{size}(\texttt{cps}))} \times 2^{p_2(\mathrm{size}(\texttt{cps}))}$$

which is itself bounded by $2^{(qh(\psi)+2)+p_1(\mathrm{size}(\texttt{cps}))+p_2(\mathrm{size}(\texttt{cps}))}$. $\square$

G Proof of Lemma 8

Proof. We want to show that the membership problem with first-order logic (with unconstrained alphabets) can be solved in polynomial space in $\mathrm{size}(\texttt{cps}) + \mathrm{size}(\psi)$. Let $\texttt{cps}$, ψ and $\mathbf{n} \in \mathbb{N}^{k-1}$ be an instance of the problem. For $i \in [1, k-1]$, let $\mathbf{n}'[i] \stackrel{\text{def}}{=} \texttt{min}(\mathbf{n}[i], 2^{qh(\psi)+1} + 1)$. By Theorem 6, the propositions below are equivalent:

- $p_1(l_1)^{\mathbf{n}[1]} \cdots p_{k-1}(l_{k-1})^{\mathbf{n}[k-1]} p_k(l_k)^\omega \in \mathrm{L}(\psi)$,
- $p_1(l_1)^{\mathbf{n}'[1]} \cdots p_{k-1}(l_{k-1})^{\mathbf{n}'[k-1]} p_k(l_k)^\omega \in \mathrm{L}(\psi)$.

Without any loss of generality, let us assume then that $\mathbf{n} \in [0, 2^{qh(\psi)+1} + 1]^{k-1}$.

Let us decompose $w = p_1(l_1)^{\mathbf{n}[1]} \cdots p_{k-1}(l_{k-1})^{\mathbf{n}[k-1]} p_k(l_k)^\omega$ as $u \cdot (v)^\omega$ where $u = p_1(l_1)^{\mathbf{n}[1]} \cdots p_{k-1}(l_{k-1})^{\mathbf{n}[k-1]} p_k$ and $v = l_k$. Note that the length of u is exponential in the size of the instance. We write $\hat{\psi}$ to denote the formula ψ in which every existential quantification is relativized to positions less than $\mathrm{len}(u) + \mathrm{len}(v) \times 2^{qh(\psi)}$. This means that every quantification '$\exists\ \mathsf{x}\ \cdots$' is replaced by '$\exists\, \mathsf{x} < (\mathrm{len}(u) + \mathrm{len}(v) \times 2^{qh(\psi)})\ \cdots$'. By [17], we know that $w \models \psi$ iff $w \models \hat{\psi}$. Now, checking $w \models \hat{\psi}$ can be done in polynomial space by using a standard first-order model-checking algorithm by restricting ourselves to positions in $[0, \mathrm{len}(u) + \mathrm{len}(v) \times 2^{qh(\psi)}]$ for existential quantifications. Such positions can be obviously encoded in polynomial space. Moreover, note that given $i \in [0, \mathrm{len}(u) + \mathrm{len}(v) \times$

Algorithm 1 FOSAT($\mathtt{cps}, \mathbf{n}, \psi, f$)

if $\phi = a(\mathsf{z})$ **then**
  Calculate the $f(\mathsf{z})$th letter b of w and return $a = b$.
else if ψ is of the form $\neg\psi'$ **then**
  return not FOSAT($\mathtt{cps}, \mathbf{n}, \psi', f$)
else if $\psi = \psi_1 \wedge \psi_2$ **then**
  return FOSAT($\mathtt{cps}, \mathbf{n}, \psi_1, f$) and FOSAT($\mathtt{cps}, \mathbf{n}, \psi_2, f$).
else if ψ is of the form $\exists \mathsf{z} < m\ \psi'$ **then**
  guess a position $k \in [0, m-1]$.
  return FOSAT($\mathtt{cps}, \mathbf{n}, \psi', f[\mathsf{z} \mapsto k]$).
else if ϕ is of the form $R(\mathsf{z}, \mathsf{z}')$ for some $R \in \{=, <, S\}$ **then**
  return $R(f(\mathsf{z}), f(\mathsf{z}'))$.
end if

$2^{qh(\psi)}]$, one can check in polynomial time what is the ith letter of w. Details are standard and omitted here. By way of example, the ith letter of w is the first letter of l_k iff $i \geq \alpha$ and $(i - \alpha) = 0 \bmod \mathrm{len}(l_k))$ with $\alpha = (\Sigma_{j\in[1,k-1]}(\mathrm{len}(p_j) + \mathrm{len}(l_j) \times \mathbf{n}[j])) + \mathrm{len}(p_j)$.

Polynomial space algorithm for membership problem is obtained by computing FOSAT($\mathtt{cps}, \mathbf{n}, \hat{\psi}, f_0$) with the algorithm FOSAT defined below (f_0 is a zero assignment function). Note that the polynomial space bound is obtained since the recursion depth is linear in size(ψ) and positions in $[0, \mathrm{len}(u) + \mathrm{len}(v) \times 2^{qh(\psi)}]$ can be encoded in polynomial space in size($\mathtt{cps}$)+size(ψ). Furthermore, since model-checking ultimately periodic words with first-order logic is PSPACE-hard [17], we deduce directly the lower bound for the membership problem with FO. □

H Proof of Theorem 10

First, we establish the result below.

Lemma 20. *Let $w \in \mathrm{L}(\mathcal{B})$ for a Büchi automaton $\mathcal{B} = \langle Q, \Sigma, q_0, \Delta, F\rangle$, such that $w = w_1.u^{2.|Q|^k}.w_2$ for some k, then there exist an integer $K \in [1, |Q|]$ such that for all $N \in [1, |Q|^{k-2}]$, $w_1.u^{2.|Q|^k - (K \times N)}.w_2 \in \mathrm{L}(\mathcal{B})$.*

Proof. Let $\mathcal{B} = \langle Q, \Sigma, q_0, \Delta, F\rangle$. Since $w = w_1.u^{2.\mathrm{card}(Q)^k}.w_2 \in \mathrm{L}(\mathcal{B})$, there exists an accepting run $\rho \in Q^\omega$ for w. We will construct an accepting run for $w' = w_1.u^{2.\mathrm{card}(Q)^k - (K \times N)}.w_2$ in $\mathcal{B}$ using ρ. In w, u is repeated $2.\mathrm{card}(Q)^k$ times. Consider the first $\mathrm{card}(Q) + 1$ iterations of u. Let the positions where the iterations of u starts be $m_1, m_2, \cdots, m_{\mathrm{card}(Q)+1}$. By pigeon-hole principle, there exists some states $q \in Q$ such that for some $i < j \in [1, \mathrm{card}(Q) + 1]$, $\rho(m_i) = \rho(m_j) = q$. Let $\alpha_1 = j - i + 1$. We consider $\mathrm{card}(Q) + 1$ iterations of u after m_j. We proceed as before to obtain α_2 and so on. Since u is repeated $2.\mathrm{card}(Q)^k$ times, we will obtain at least $\mathrm{card}(Q)^{k-1}$ (possibly different) values as $\alpha_1, \alpha_2, \cdots, \alpha_{\mathrm{card}(Q)^{k-1}} \in [1, \mathrm{card}(Q)]$ because $\mathrm{card}(Q)^{k-1} \times (\mathrm{card}(Q) + 1) \leq 2.\mathrm{card}(Q)^k$. Again, by pigeon-hole principle, we know that there exists $j_1, j_2, \ldots, j_{\mathrm{card}(Q)^{k-2}} \in [1, \mathrm{card}(Q)^{k-1}]$ such that $\alpha_{j_1} = \alpha_{j_2} = \ldots = \alpha_{j_{\mathrm{card}(Q)^{k-2}}} = K$ for some $K \in [1, \mathrm{card}(Q)]$ because $K \times \mathrm{card}(Q)^{k-2} \leq \mathrm{card}(Q)^{k-1}$. Note that for each such α_j, $j \in \{j_1, j_2, \ldots, j_{\mathrm{card}(Q)^{k-2}}\}$, we have a corresponding different loop structure in ρ where we have positions a and b in w such that $w[a, b] = (u)^K$ and $\rho(a) = \rho(b) = q$ for some $q \in Q$ as shown in Figure 1. Hence, the run $\rho(1) \ldots \rho(a).\rho(b) \ldots$ is still an accepting run in $\mathcal{B}$ for $w_1.u^{2.\mathrm{card}(Q)^k - (K)}.w_2$. Since, there are $\mathrm{card}(Q)^{k-2}$ such loops, it is easy to see that for every $N \in [1, \mathrm{card}(Q)^{k-2}]$, we can remove the loops corresponding to $\alpha_{j_1}, \alpha_{j_2}, \ldots, \alpha_{j_N}$ and have an accepting run for the word $w_1.u^{2.\mathrm{card}(Q)^k - (K \times N)}.w_2$ in $\mathcal{B}$. $\square$

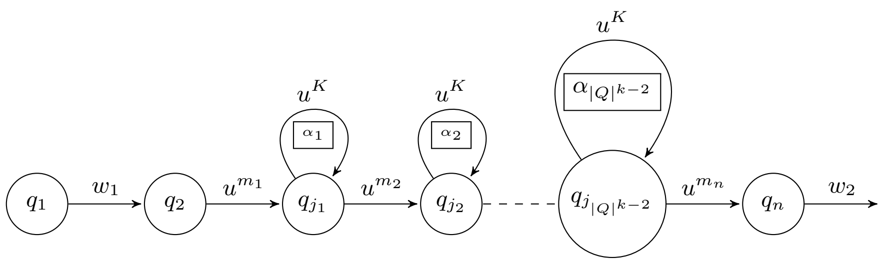


Fig. 1. Shape of a sample run

Proof (Theorem 10). Since $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(\mathcal{B}) \neq \emptyset$, there exist an infinite word w such that $w \in \mathrm{L}(\mathcal{B}) \cap \mathrm{L}(\mathtt{cps})$. Let $\mathbf{y} \in \mathbb{N}^{k-1}$ be the vector such that $w = p_1(l_1)^{\mathbf{y}[1]} \ldots p_{k-1}(l_{k-1})^{\mathbf{y}[k-1]} p_k l_k^{\omega}$. We will now prove that either $\mathbf{y} \in [0, 2^{pol_1(\mathrm{size}(\mathtt{cps}))} + 2.\mathrm{size}(\mathcal{B})^k \times 2^{\mathtt{pol}_1(\mathrm{size}(\mathtt{cps})) + \mathtt{pol}_2(\mathrm{size}(\mathtt{cps}))}]^{k-1}$ or we can construct another word

$$w' = p_1(l_1)^{\mathbf{y}'[1]} \ldots p_{k-1}(l_{k-1})^{\mathbf{y}'[k-1]} p_k l_k^{\omega}$$

such that $\mathbf{y}' \in [0, 2^{pol_1(\mathrm{size}(\mathtt{cps}))} + 2.\mathrm{size}(\mathcal{B})^k \times 2^{\mathtt{pol}_1(\mathrm{size}(\mathtt{cps})) + \mathtt{pol}_2(\mathrm{size}(\mathtt{cps}))}]^{k-1}$ and $w' \in \mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(\mathcal{B})$. Since, $\mathbf{y} \models \phi(\mathsf{x}_1, \ldots, \mathsf{x}_{k-1})$ and $\phi(\mathsf{x}_1, \ldots, \mathsf{x}_{k-1})$ is a quantifier-free Presburger formula, we know that there exist $\mathbf{b}, \mathbf{P}_1, \mathbf{P}_2 \cdots \mathbf{P}_\alpha \in [0, 2^{\mathtt{pol}_1(\mathrm{size}(\mathtt{cps}))}]^{k-1}$ and $\alpha \leq 2^{pol_2(\mathrm{size}(\mathtt{cps}))}$ such that $\mathbf{y} = \mathbf{b} + \Sigma_{i \in [1,\alpha]} a_i.\mathbf{P}_i$ for some $(a_1, a_2, \ldots, a_\alpha) \in \mathbb{N}^\alpha$. Let us assume that $\mathbf{y} \notin [0, 2^{\mathtt{pol}_1(\mathrm{size}(\mathtt{cps}))} + 2.\mathrm{size}(\mathcal{B})^k \times 2^{\mathtt{pol}_1(\mathrm{size}(\mathtt{cps})) + \mathtt{pol}_2(\mathrm{size}(\mathtt{cps}))}]^{k-1}$ and hence there exists some a_j, $j \in [1, \alpha]$ such that $a_j > 2.\mathrm{card}(Q)^k$. We would like to find a such that $\mathbf{y}' = \mathbf{b} + \Sigma_{i \in [1,j-1]} a_i.\mathbf{P}_i + (a_j - a)\mathbf{P}_j + \Sigma_{i \in [j+1,\alpha]} a_i.\mathbf{P}_i$ and $w' = p_1(l_1)^{\mathbf{y}'[1]} \ldots p_{k-1} (l_{k-1})^{\mathbf{y}'[k-1]} p_k l_k^{\omega} \in \mathrm{L}(\mathcal{B}) \cap \mathrm{L}(\mathtt{cps})$.

1. For any $a \leq a_j$, we have that, w' with $\mathbf{y}' = \mathbf{b} + \Sigma_{i \in [1,j-1]} a_i.\mathbf{P}_i + (a_j - a) \mathbf{P}_j + \Sigma_{i \in [j+1,\alpha]} a_i.\mathbf{P}_i$, $w' \in \mathrm{L}(\mathtt{cps})$. Indeed by selecting any $a \in [a_j, a_j - 2.\mathrm{card}(Q)^k]$, we will obtain $\mathbf{y}' \in [0, 2^{\mathtt{pol}_1(\mathrm{size}(\mathtt{cps}))} + 2.\mathrm{size}(\mathcal{B})^k \times 2^{\mathtt{pol}_1(\mathrm{size}(\mathtt{cps})) + \mathtt{pol}_2(\mathrm{size}(\mathtt{cps}))}]^{k-1}$ where $w' \in \mathrm{L}(\mathtt{cps})$.

2. For showing that there exists a value for a such that $\mathbf{y}' \in [0, 2^{\texttt{pol}_1(\text{size}(\texttt{cps}))} + 2.\text{size}(\mathcal{B})^k \times 2^{\texttt{pol}_1(\text{size}(\texttt{cps}))+\texttt{pol}_2(\text{size}(\texttt{cps}))}]^{k-1}$ and $w' \in \mathrm{L}(\mathcal{B})$ we will use Lemma 20. For each $m \in [1, k-1]$, we take $r_m = (l_m)^{\mathbf{P}_j[m]}$ i.e. r_m is $\mathbf{P}_j[m]$ copies of l_m. Note that by our assumption, for each $m \in [1, k-1]$ we can factor w as $w = w_1^m.(r_m)^{a_m}.w_2^m$ where $a_m \geq 2.\text{card}(Q)^k$. Thus, applying Lemma 20, we get that there exist $K_m \in [1, \text{card}(Q)]$ such that for any $N_m \in [1, \text{card}(Q)^{k-2}]$, $w'' = w_1^m.(r_m)^{a_m-(N_m \times K_m)}.w_2^m \in \mathrm{L}(\mathcal{B})$. For each $m \in [1, k]$ we take $N_m = K_1 \times K_2 \cdots K_{m-1} \times K_{m+1} \cdots K_{k-1}$ which is less than or equal to $\text{card}(Q)^{k-2}$. It is clear that for each $m \in [1, k-1]$, the number of iteration of r_m we reduce is $N_m \times K_m$ and is same for all m, $N_m \times K_m = K_1 \times K_2 \cdots K_{k-1}$. Combining the result of Lemma 20 for every loop l_m, $m \in [1, k-1]$ and taking $a = K_1 \times K_2 \cdots K_{k-1}$ in $\mathbf{y}'$ we obtain w' such that $w' \in \mathrm{L}(\mathcal{B})$.

We can continue the process to obtain $\mathbf{y}' \in [0, 2^{\texttt{pol}_1(\text{size}(\texttt{cps}))} + 2.\text{size}(\mathcal{B})^k \times 2^{\texttt{pol}_1(\text{size}(\texttt{cps}))+\texttt{pol}_2(\text{size}(\texttt{cps}))}]^{k-1}$ and $w' \in \mathrm{L}(\mathcal{B})$. □

I Proof of Lemma 11

Proof. Consider any given specification A in BA, a constrained path schema `cps`.We would first construct the Büchi automaton $\mathcal{B}_A$ corresponding to A as explained in Section 2.3. Recall, that A in BA has transitions labelled with Boolean combination over $at \cup ag_n$ whereas the equivalent $\mathcal{B}_A$ has transitions labelled with elements of $\Sigma = 2^{at \cup ag_n}$. Hence, in effect, $\mathcal{B}_A$ could have an exponential number of transitions. On the other hand by definition `cps` is defined over an alphabet $\Sigma' \subseteq \Sigma$. By Lemma 3, we know that BA has the nice subalphabet property. Hence, we can transform A over Σ to A' over Σ' in polynomial time. The Büchi automata $\mathcal{B}_{A'}$ obtained from A' following the construction in Section 2.3 has transitions labelled by letters from Σ'. Clearly in this case the number of transitions in $\mathcal{B}_{A'}$ is polynomial in size(`cps`). We obtain the following equivalences,

– using Lemma 3 and the fact that $\mathrm{L}(\texttt{cps}) \subseteq (\Sigma')^\omega$, $\mathrm{L}(A) \cap \mathrm{L}(\texttt{cps})$ is non-empty iff $w \in \mathrm{L}(A') \cap \mathrm{L}(\texttt{cps})$ is non-empty.
– Since, $\mathcal{B}_{A'}$ is obtained from A' following the construction from Section 2.3, $\mathrm{L}(\mathcal{B}_{A'}) \cap \mathrm{L}(\texttt{cps})$ is non-empty iff $w \in \mathrm{L}(A') \cap \mathrm{L}(\texttt{cps})$ is non-empty.

Checking $\mathrm{L}(\mathcal{B}_{A'}) \cap \mathrm{L}(\texttt{cps})$ amounts to guessing $\mathbf{n} \in f_{\mathrm{BA}}(\mathcal{B}_{A'}, \texttt{cps})$ and checking for $w = p(l_1)^{\mathbf{n}[1]} \cdots p_{k-1}(l_{k-1})^{\mathbf{n}[k-1]} p_k l_k^\omega$, $w \in \mathrm{L}(\mathcal{B}_{A'}) \cap \mathrm{L}(\texttt{cps})$. We know that checking $w \in \mathrm{L}(\mathcal{B}_{A'}) \cap \mathrm{L}(\texttt{cps})$ is in PTime and the construction of $\mathcal{B}_{A'}$ from A takes only polynomial time. Thus, checking $\mathrm{L}(A) \cap \mathrm{L}(\texttt{cps})$ is non-empty can be done in polynomial time. □

J proof of Lemma 13

Proof. First we prove that the membership problem for $\mathcal{L}$ having the nice BA property is in PSpace. Let $A \in \mathcal{L}$ over the constrained alphabet $\langle at, ag_n, \Sigma \rangle$ and let $w = p_1(l_1)^{\mathbf{n}[1]} \cdots p_{k-1}(l_{k-1})^{\mathbf{n}[k-1]} p_k(l_k)^\omega$ be a word over Σ. We would like to check whether $w \in \mathrm{L}(A)$ which is equivalent, thanks to the nice BA property, to check whether $w \in \mathrm{L}(\mathcal{B}_A)$.

To verify if $w \in \mathrm{L}(\mathcal{B}_A)$, we try to find a "lasso" structure in Büchi automaton. Assume $\mathcal{B}_A = \langle Q, \Sigma, \Delta, q_i, F\rangle$. We proceed as follows. First, we guess a state $q_f \in F$ and a position $j \in [1, \mathrm{len}(l_k)]$. Then we consider the two finite state automata $\mathcal{A}_1 = \langle Q, \Sigma, q_0, \Delta, \{q_f\}\rangle$ and $\mathcal{A}_2 = \langle Q, \Sigma, q_f, \Delta, \{q_f\}\rangle$. And our method returns true iff both the following conditions are true:

1. $p_1(l_1)^{\mathbf{y}[1]} \dots p_{k-1}(l_{k-1})^{\mathbf{y}[k-1]} p_k l_k[1, j] \in \mathrm{L}(\mathcal{A}_1)$.
2. $\mathrm{L}(\mathcal{A}_2) \cap \mathrm{L}(l_k[j+1, \mathrm{len}(l_k)] l_k^* l_k[1, j]) \neq \emptyset$.

We will now show the correctness of the above procedure. First let us assume that $w = p_1(l_1)^{\mathbf{y}[1]} \dots p_{k-1}(l_{k-1})^{\mathbf{y}[k-1]} p_k l_k^\omega \in \mathrm{L}(\mathcal{B}_A)$. Thus, there is an accepting run $\rho \in \Delta^\omega$ for w in $\mathcal{B}_A$. According to the Büchi acceptance condition there exists a state $q_f \in F$ which is visited infinitely often. In w only l_k is taken infinitely many times. Thus, l_k being of finite size, there exists a position $j \in [1, \mathrm{len}(l_k)]$ such that transitions of the form $q \xrightarrow{l_k(j)} q_f$ for some $q \in Q$ occurs infinitely many times in ρ. Thus, for ρ to be an accepting run, there exists $w' = \mathrm{L}(p_1(l_1)^{\mathbf{y}[1]} \dots p_{k-1}(l_{k-1})^{\mathbf{y}[k-1]} p_k l_k[1, j])$, which has a run in $\mathcal{B}_A$ from q_i to q_f and there must exists words $w'' \in \mathrm{L}(l_k[j+1, \mathrm{len}(l_k)] l_k^* l_k[1, j])$ which has a run from q_f to q_f. Hence we deduce that $w' \in \mathrm{L}(\mathcal{A}_1)$ and $\mathrm{L}(\mathcal{A}_2) \cap \mathrm{L}(l_k[j + 1, \mathrm{len}(l_k)] l_k^* l_k[1, j]) \neq \emptyset$. Thus, there exists at least one choice of q_f and j, for which both the checks return true and hence the procedure returns true.

Now let us assume that the procedure returns true. Thus, there exists $q_f \in F$ and $j \in [1, \mathrm{len}(l_k)]$ such that $w_1 = p_1(l_1)^{\mathbf{y}[1]} \dots p_{k-1}(l_{k-1})^{\mathbf{y}[k-1]} p_k l_k[1, j]$ is in $\mathrm{L}(\mathcal{A}_1)$ and $\mathrm{L}(\mathcal{A}_2) \cap \mathrm{L}(l_k[j+1, \mathrm{len}(l_k)] l_k^* l_k[1, j]) \neq \emptyset$. From the second point, we deduce that there exists a word $w_2 = l_k[j+1, \mathrm{len}(l_k)] l_k^n l_k[1, j] \in \mathrm{L}(\mathcal{A}_2)$ for some n. Consider the word $w = w_1.(w_2)^\omega$. First we have directly that $w \in \mathrm{L}(\mathtt{cps})$. And by construction of $\mathcal{A}_1$ and $\mathcal{A}_2$ we know that w_1 has a run in $\mathcal{B}_A$ starting from q_i to q_f and w_2 has a run in $\mathcal{B}_A$ starting from q_f to q_f. Since q_f is an accepting state of $\mathcal{B}_A$, we deduce that $w \in \mathrm{L}(\mathcal{B}_A)$.

The proof that the above procedure belongs to PSPACE is standard and used the nice BA property which allows us to perform the procedure "on-the-fly". First note that for A in $\mathcal{L}$ having the nice BA property, the corresponding Büchi automaton $\mathcal{B}_A$ can be of exponential size in the size of the A, so we cannot construct the transition relation of $\mathcal{B}_A$ explicitly, instead we do it on-the-fly. We consider the different steps of the procedure and show that they can be done in polynomial space.

1. $\mathcal{A}_1$ and $\mathcal{A}_2$ are essentially copies of $\mathcal{B}_A$ and hence their transition relations are also not constructed explicitly. But, by the nice BA property, their states can be represented in polynomial space.
2. Checking $p_1(l_1)^{\mathbf{y}[1]} \dots p_{k-1}(l_{k-1})^{\mathbf{y}[k-1]} p_k l_k[1, j] \in \mathrm{L}(\mathcal{A}_1)$ can be done by simulating $\mathcal{A}_1$ on this word. Note that for simulating $\mathcal{A}_1$, at any position we only need to store the previous state and the letter at current position to

obtain the next state of $\mathcal{A}_1$. Thus, this can be performed in polynomial space in size(A) and size($\mathtt{cps}$) + size($\mathbf{y}$).

3. Checking $\mathrm{L}(\mathcal{A}_2) \cap \mathrm{L}(l_k[j+1, \mathrm{len}(l_k)] l_k^* l_k[1,j]) \neq \emptyset$ can be done by constructing a finite state automaton $\mathcal{A}_{loop}$ for $\mathrm{L}(l_k[j+1, \mathrm{len}(l_k)] l_k^* l_k[1,j])$ and by checking for reachability of final state in the automaton $\mathcal{A}_2 \times \mathcal{A}_{loop}$. Note that size($\mathcal{A}_{loop}$) is polynomial, but since size($\mathcal{A}_2$) can be of exponential size, size($\mathcal{A}_2 \times \mathcal{A}_{loop}$) can also be of exponential magnitude. However, the graph accessibility problem (GAP) is in NLogSpace, so $\mathrm{L}(\mathcal{A}_2) \cap \mathrm{L}(\mathcal{A}_{loop}) \neq \emptyset$ can also be done in nondeterministic polynomial space.

Thus, the whole procedure can be completed in nondeterministic polynomial space and by applying Savitch's theorem, we obtain that for any $\mathcal{L}$, satisfying the nice BA property, the membership problem for $\mathcal{L}$ having the nice BA property is in PSpace.

Now we will prove that the intersection non-emptiness problem for $\mathcal{L}$ having the nice BA property is in PSpace too. Let A in $\mathcal{L}$ with the nice BA property and let $\mathtt{cps} = \langle p_1(l_1)^* \cdots p_{k-1}(l_{k-1})^* p_k(l_k)^\omega, \phi(\mathsf{x}_1, \ldots, \mathsf{x}_{k-1})\rangle$ be a constrained path schema. Thanks to the nice BA property we have that $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(A) \neq \emptyset$ iff $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(\mathcal{B}_A) \neq \emptyset$. Using Theorem 10 we have $\mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(\mathcal{B}_A) \neq \emptyset$ iff there exists $\mathbf{y} \in [0, f_{BA}(\mathcal{B}_A, \mathtt{cps})]^{k-1}$ such that $p_1(l_1)^{\mathbf{y}[1]} \ldots p_{k-1}(l_{k-1})^{\mathbf{y}[k-1]} p_k l_k^\omega \in \mathrm{L}(\mathcal{B}_A) \cap \mathrm{L}(\mathtt{cps})$ where $f_{BA}(\mathcal{B}_A, \mathtt{cps})$ is equal to $2^{\mathtt{pol}_1(\mathrm{size}(\mathtt{cps}))} + 2.\mathrm{card}(Q)^{\mathrm{size}(\mathtt{cps})} \times 2^{\mathtt{pol}_1(\mathrm{size}(\mathtt{cps})) + pol_2(\mathrm{size}(\mathtt{cps}))}$ (Q being the set of states of $\mathcal{B}_A$ whose cardinality is, thanks to the nice BA property, at most exponential in the size of A). Hence our algorithm amounts to guess some $\mathbf{y} \in [0, f_{BA}(\mathcal{B}_A, \mathtt{cps})]^{k-1}$ and check whether $w = p_1(l_1)^{\mathbf{y}[1]} \cdots p_{k-1}(l_{k-1})^{\mathbf{y}[k-1]} p_k(l_k)^\omega \in \mathrm{L}(\mathtt{cps}) \cap \mathrm{L}(A)$. Since the membership problem for A can be done in PSpace and since the $\mathbf{y}[i]$ can be encoded in polynomial space in the size of A and $\mathcal{CPS}$, we deduce that the intersection non-emptiness problem for $\mathcal{L}$ with the nice BA property is in PSpace. $\square$

K Proof of Theorem 15

Proof. (II) is direct consequence of (I).

First, we recall that an *alternating finite automaton* is a structure of the form $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ such that Q and Σ are finite nonempty sets, $\delta : Q \times \Sigma \to \mathbb{B}^+(Q)$ is the transition function ($\mathbb{B}^+(Q)$ is the set of positive Boolean formulae built over Q), $q_0 \in Q$ and $F \subseteq Q$. The *acceptance predicate* $Acc \subseteq Q \times \Sigma^*$ is defined by induction on the length of the second component so that (1) $\langle q_f, \varepsilon\rangle \in Acc$ whenever $q_f \in F$ and (2) $\langle q, a \cdot w\rangle \in Acc$ iff $v \models \delta(q, a)$ where v is the Boolean assignment such that $v(q') = \top$ iff $\langle q', w\rangle \in Acc$. We write $\mathrm{L}(\mathcal{A})$ to denote the language $\{w \in \Sigma^* : \langle q_0, w\rangle \in Acc\}$ and more generally, $\mathrm{L}(\mathcal{A}, q) \stackrel{\text{def}}{=} \{w \in \Sigma^* : \langle q, w\rangle \in Acc\}$. It has been shown in [9] that checking whether an alternating finite automaton $\mathcal{A}$ with a singleton alphabet has a non-empty language $\mathrm{L}(\mathcal{A})$ is PSpace-hard. Without loss of generality, we can assume that ($\star$) $q_0 \notin F$, ($\star\star$) for every $q_f \in F$, $\delta(q_f, a) = \perp$ and ($\star\star\star$) for every $q \in Q$,

$\delta(q, \{a\}) \neq \top$, assuming that a is the only letter, and still preserves PSPACE-hardness. Indeed, let $\mathcal{A} = (Q, \{a\}, \delta, q_0, F)$ be an alternating finite automaton and $\mathcal{A}' = (Q', \{a\}, \delta', q_0^{\rm new}, \{q_f^{\rm new}\})$ be its variant such that $Q' = Q \uplus \{q_0^{\rm new}, q_f^{\rm new}\}$, $\delta'(q_0^{\rm new}, a) = q_0$, $\delta'(q_f^{\rm new}, a) = \perp$ and for every $q \in Q$, $\delta'(q, a)$ is obtained from $\delta(q, a)$ by simultaneously replacing every occurrence of $q_f \in F$ by $(q_f \vee q_f^{\rm new})$. In the case, $\delta(q, a) = \top$ with $q \in Q$, $\delta'(q, a)$ is defined as $q \vee q_f^{\rm new}$. It is clear from the construction that $\mathcal{A}'$ follows the conditions of our assumption and $\mathrm{L}(\mathcal{A}') = a \cdot \mathrm{L}(\mathcal{A})$; whence $\mathrm{L}(\mathcal{A})$ is non-empty iff $\mathrm{L}(\mathcal{A}')$ is non-empty.

In order to prove the result for ABA, it is sufficient to observe that given an alternating finite automaton $\mathcal{A}$ built over the singleton alphabet $\{a\}$, one can build in logarithmic space an alternating Büchi automaton $\mathcal{A}'$ over the alphabet $\{a, b\}$ such that $\mathrm{L}(\mathcal{A}') = \mathrm{L}(\mathcal{A}) \cdot \{b\}^\omega$. Roughly speaking, the reduction consists in taking the accepting states of $\mathcal{A}$ and in letting them accept $\{b\}^\omega$ in $\mathcal{A}'$. PSPACE-hardness of the intersection non-emptiness problem for ABA is obtained by noting that $\mathrm{L}(\mathcal{A})$ is non-empty iff $\mathrm{L}(\langle a^* \cdot b^\omega, \top\rangle) \cap \mathrm{L}(\mathcal{A}') \neq \emptyset$.

Now, let us deal with μTL. The PSPACE-hardness is essentially obtained by reducing nonemptiness problem for alternating finite automata with a singleton alphabet (see e.g. [9]) into the vectorial linear μ-calculus with a fixed simple constrained path schema. Reduction in polynomial-time into linear μ-calculus is then possible when formula sizes are measured in terms of numbers of subformulae. This is a standard type of reduction (see e.g. [27, Section 5.4]); we provide details below not only to be self-contained but also because we need a limited number of resources: no greatest fixed point operator (e.g. no negation of least fixed point operator) and we then use a simple path schema. In the sequel, for ease of presentation, we consider this latter class of alternating finite automata and we present a logarithmic-space reduction into the intersection non-emptiness problem with linear μ-calculus. More precisely, for every alternating finite automaton $\mathcal{A}$ built over the singleton alphabet $\{\{\mathtt{p}\}\}$, we build a formula $\phi_{\mathcal{A}}$ in the linear μ-calculus (without $\mathtt{X}^{-1}$ and the greatest fixed-point operator ν) such that $\mathrm{L}(\mathcal{A})$ is non-empty iff there is $\{\mathtt{p}\} \cdot \{\mathtt{p}\}^{n_1} \cdot \emptyset^\omega$ in $\mathrm{L}(\mathtt{cps})$ with the constrained path schema $\mathtt{cps} = \langle \{\mathtt{p}\} \cdot \{\mathtt{p}\}^* \cdot \emptyset^\omega, \top\rangle$ and $\{\mathtt{p}\} \cdot \{\mathtt{p}\}^{n_1} \cdot \emptyset^\omega \models \phi_{\mathcal{A}}$. In order to define $\phi_{\mathcal{A}}$, we build first an intermediate formula in the vectorial version of the linear μ-calculus, see e.g. similar developments in [27, Section 5.4], and then we translate it into an equivalent formula in the linear μ-calculus by using the well-known *Bekič's Principle.*

Let $\mathcal{A} = (Q, \{\{\mathtt{p}\}\}, \delta, q_0, F)$ be a alternating finite automaton with a singleton alphabet such that $q_0 \notin F$, and for every $q_f \in F$, $\delta(q_f, \{\mathtt{p}\}) = \perp$. We order the states of $Q \setminus F$ with $q_1, \ldots, q_\alpha$ such that q_1 is the initial state.

We define the formulae in the vectorial version of linear μ-calculus $\psi_1^0, \ldots, \psi_\alpha^0, \psi_1^1, \ldots, \psi_{\alpha-1}^1, \ldots, \psi_1^i, \ldots, \psi_{\alpha-i}^i, \ldots, \psi_1^{\alpha-1}$ and such that $\mu\ \mathtt{z}_1\ \cdot \psi_1^{\alpha-1}$ belongs to the (standard) linear μ-calculus. Such formulae will satisfy the following conditions.

(I) For all $n \geq 1$, $\{\mathtt{p}\}^n \in \mathrm{L}(\mathcal{A})$ iff $\{\mathtt{p}\}^n \cdot \emptyset^\omega \models \mu\langle \mathtt{z}_1, \ldots, \mathtt{z}_\alpha\rangle\ \langle \psi_1^0, \ldots, \psi_\alpha^0\rangle \cdot \mathtt{z}_1$.

(II) For all $j \in [0, \alpha - 1]$, $\mu\langle \mathtt{z}_1, \ldots, \mathtt{z}_{\alpha-j}\rangle\ \langle \psi_1^j, \ldots, \psi_{\alpha-j}^j\rangle \cdot \mathtt{z}_1$ is equivalent to $\mu\langle \mathtt{z}_1, \ldots, \mathtt{z}_{\alpha-j-1}\rangle\ \langle \psi_1^{j+1}, \ldots, \psi_{\alpha-j-1}^{j+1}\rangle \cdot \mathtt{z}_1$.

(III) Consequently, for all $n \geq 1$, $\{\mathtt{p}\}^n \in \mathrm{L}(\mathcal{A})$ iff $\{\mathtt{p}\}^n \cdot \emptyset^\omega \models \mu\ \mathtt{z}_1 \ \cdot \psi_1^{\alpha-1}$ and we pose $\phi_{\mathcal{A}} = \mu\ \mathtt{z}_1 \ \cdot \psi_1^{\alpha-1}$.

Let us define below the formulae: the substitutions are simple and done hierarchically.

(init) For every $i \in [1,\alpha]$, ψ_i^0 is obtained from $\delta(q_i, \{\mathtt{p}\})$ by substituting each $q_j \in Q \backslash F$ by $\mathtt{X}\mathtt{z}_j$ and each $q_f \in F$ by $\mathtt{X}\neg\mathtt{p}$, and then by taking the conjunction with $\mathtt{p}$. So, ψ_i^0 can be written schematically as $\mathtt{p} \wedge \delta(q_i, \{\mathtt{p}\})[q_j \leftarrow \mathtt{X}\mathtt{z}_j, q_f \leftarrow \mathtt{X}\neg\mathtt{p}]$.

(ind) For every $j \in [1, \alpha-1]$, for every $i \in [1, \alpha - j]$, ψ_i^j is obtained from ψ_i^{j-1} by substituting every occurrence of $\mathtt{z}_{\alpha-j+1}$ by $\mu\mathtt{z}_{\alpha-j+1}\ \psi_{\alpha-j+1}^{j-1}$.

Note that $\mu\ \mathtt{z}_1 \ \cdot \psi_1^{\alpha-1}$ can be built in logarithmic space in the size of $\mathcal{A}$ since formulae are represented as DAGs (their size is the number of subformulae) and for all $j \in [1, \alpha-1]$ and $i \in [1, \alpha-j]$, ψ_i^j has no free occurrences of $\mathtt{z}_{\alpha-j+1}, \ldots, \mathtt{z}_\alpha$.

It remains to check that (I)–(III) hold. First, observe that (III) is a direct consequence of (I) and (II). By *Bekič's Principle*, see e.g. [1, Section 1.4.2], $\mu\langle \mathtt{z}_1, \ldots, \mathtt{z}_j\rangle\ \langle \varphi_1(\mathtt{z}_1, \ldots, \mathtt{z}_j), \ldots, \varphi_j(\mathtt{z}_1, \ldots, \mathtt{z}_j)\rangle \cdot \mathtt{z}_1$ is equivalent to

$$\mu\langle \mathtt{z}_1, \ldots, \mathtt{z}_{j-1}\rangle\ \langle \varphi_1(\mathtt{z}_1, \ldots, \mathtt{z}_{j-1}, \varphi'), \ldots, \varphi_{j-1}(\mathtt{z}_1, \ldots, \mathtt{z}_{j-1}, \varphi')\rangle \cdot \mathtt{z}_1$$

where $\varphi' = \mu\mathtt{z}_j\ \varphi_j(\mathtt{z}_1, \ldots, \mathtt{z}_j)$. Note that the substitution performed to build the formula follows exactly the same principle. For every $j \in [1, \alpha-1]$, we obtain $\langle \psi_1^j, \ldots, \psi_{\alpha-j}^j\rangle$ by replacing $\mathtt{z}_{\alpha-j+1}$ by $\mu\mathtt{z}_{\alpha-j+1}\ \psi_{\alpha-j+1}^{j-1}$ in $\langle \psi_1^{j-1}, \ldots, \psi_{\alpha-j+1}^{j-1}\rangle$. Thus by Bekič's Principle,

$$\mu\langle \mathtt{z}_1, \ldots, \mathtt{z}_{\alpha-j}\rangle\ \langle \psi_1^j, \ldots, \psi_{\alpha-j}^j\rangle \cdot \mathtt{z}_1 \Leftrightarrow \mu\langle \mathtt{z}_1, \ldots, \mathtt{z}_{\alpha-j-1}\rangle\ \langle \psi_1^{j+1}, \ldots, \psi_{\alpha-j-1}^{j+1}\rangle \cdot \mathtt{z}_1$$

is valid for all $j \in [0, \alpha-1]$. It remains to verify that (I) holds true.

In vectorial linear μ-calculus, formulae with outermost fixed-point operators are of the form $\mu\langle \mathtt{z}_1, \ldots, \mathtt{z}_\beta\rangle\langle \phi_1, \ldots, \phi_\beta\rangle \cdot \mathtt{z}_j$ with $j \in [1, \beta]$. Whereas fixed points in linear μ-calculus are considered for monotone functions over the complete lattice $\langle 2^{\mathbb{N}}, \subseteq\rangle$, fixed points in vectorial linear μ-calculus are considered for monotone functions over the complete lattice $\langle (2^{\mathbb{N}})^\beta, \subseteq\rangle$, where $\langle Y_1, \ldots, Y_\beta\rangle \subseteq \langle Y_1', \ldots, Y_\beta'\rangle$ iff for every $i \in [1,\beta]$, we have $Y_i \subseteq Y_i'$. So, the satisfaction relation is defined as follows. Given a model $\sigma \in (2^{\mathrm{AT}})^\omega$, $\sigma, i \models_f \mu\langle \mathtt{z}_1, \ldots, \mathtt{z}_\beta\rangle\langle \phi_1, \ldots, \phi_\beta\rangle \cdot \mathtt{z}_j$ (assuming that the variables $\mathtt{z}_k$ occurs positively in the ϕ_l's) iff $i \in Z_j^\mu$ where $\langle Z_1^\mu, \ldots, Z_\beta^\mu\rangle$ is the least fixed point of the monotone function $\mathcal{F}_{f,\sigma} : (2^{\mathbb{N}})^\beta \to (2^{\mathbb{N}})^\beta$ defined by $\mathcal{F}_{f,\sigma}(\mathcal{Y}_1, \ldots, \mathcal{Y}_\beta) = \langle \mathcal{Y}_1', \ldots, \mathcal{Y}_\beta'\rangle$ where

$$\mathcal{Y}_l' \stackrel{\text{def}}{=} \{i' \in \mathbb{N} : \sigma, i' \models_{f[\mathtt{z}_1 \leftarrow \mathcal{Y}_1, \ldots, \mathtt{z}_\beta \leftarrow \mathcal{Y}_\beta]} \phi_l\}$$

It is well-known that the least fixed point $\langle Z_1^\mu, \ldots, Z_\beta^\mu\rangle$ can be obtained by an iterative process: $\langle Z_1^0, \ldots, Z_\beta^0\rangle \stackrel{\text{def}}{=} \langle \emptyset, \ldots, \emptyset\rangle$, $\langle Z_1^{i+1}, \ldots, Z_\beta^{i+1}\rangle \stackrel{\text{def}}{=} \mathcal{F}_{f,\sigma}(Z_1^i, \ldots, Z_\beta^i)$ for all $i \geq 0$ and, $\langle Z_1^\mu, \ldots, Z_\beta^\mu\rangle = \bigcup_i \langle Z_1^i, \ldots, Z_\beta^i\rangle$.

Let σ_n be the model $\{\mathtt{p}\}^n \cdot \emptyset^\omega$ with $n > 0$, $f_\emptyset$ be the constant assignment equal to $\emptyset$ everywhere and $\mathcal{F}_{f_\emptyset,\sigma_n}$ be the monotone function $\mathcal{F}_{f_\emptyset,\sigma_n} : (2^{\mathbb{N}})^\alpha \rightarrow (2^{\mathbb{N}})^\alpha$ defined from $\mu\langle \mathtt{z}_1, \ldots, \mathtt{z}_\alpha \rangle\ \langle \psi_1^0, \ldots, \psi_\alpha^0 \rangle \cdot \mathtt{z}_1$.

Let us show by induction that for every $i \in [1, n]$, the ith iterated tuple $\langle Z_1^i, \ldots, Z_\alpha^i \rangle$ verifies that for every $l \in [1, \alpha]$, $u \in Z_l^i$ iff $u \in [n-i, n-1]$ and $\{\mathtt{p}\}^{n-u} \in \mathrm{L}(\mathcal{A}, q_l)$.
Base Case: $i = 1$. The propositions below are equivalent ($l \in [1, \alpha]$):

- $u \in Z_l^1$,
- $\sigma_n, u \models_{f_\emptyset[\mathtt{z}_1 \leftarrow \emptyset, \ldots, \mathtt{z}_\alpha \leftarrow \emptyset]} \psi_l^0$ (by definition of $\mathcal{F}_{f_\emptyset,\sigma_n}$),
- $\sigma_n, u \models_{f_\emptyset[\mathtt{z}_1 \leftarrow \emptyset, \ldots, \mathtt{z}_\alpha \leftarrow \emptyset]} \mathtt{p} \wedge \delta(q_i, \{\mathtt{p}\})[q_j \leftarrow \mathtt{X}\mathtt{z}_j, q_f \leftarrow \mathtt{X}\neg\mathtt{p}]$ (by definition of ψ_l^0),
- $\sigma_n, u \models \mathtt{p} \wedge \delta(q_l, \{\mathtt{p}\})[q_j \leftarrow \perp, q_f \leftarrow \mathtt{X}\neg\mathtt{p}]$ (by definition of $\models$),
- $\sigma_n, u \models \mathtt{p}$ and there is a Boolean valuation $v : Q \rightarrow \{\perp, \top\}$ such that for every $q \in (Q \setminus F)$, we have $v(q) = \perp$ and $v \models \delta(q_l, \{\mathtt{p}\})$,
- $\sigma_n, u \models \mathtt{p}$, $\sigma_n, u+1 \models \neg\mathtt{p}$ and $\langle q_l, \{\mathtt{p}\} \rangle \in \mathit{Acc}$ (by definition of Acc and by assumption $(\star\star\star)$),
- $u = n-1$ and $\{\mathtt{p}\}^{n-u} \in \mathrm{L}(\mathcal{A}, q_l)$ (by definition of σ_n and $\mathrm{L}(\mathcal{A}, q_l)$).

Before proving the induction, we observe that we can also show by induction, that for all $l \in [1, \alpha]$ and i, $Z_l^i \subseteq [n-i, n-1]$ (†).
Induction Step: Now let us assume that for some $i \in [1, n]$, the ith iterated tuple $\langle Z_1^i, \ldots, Z_\alpha^i \rangle$ verifies that for every $l \in [1, \alpha]$, $u \in Z_l^i$ iff $u \in [n-i, n-1]$ and $\{\mathtt{p}\}^{n-u} \in \mathrm{L}(\mathcal{A}, q_l)$. We will show that the same holds true for $(i+1)$th iteration $\langle Z_1^{i+1}, \ldots, Z_\alpha^{i+1} \rangle$. Since $Z_l^i \subseteq Z_l^{i+1}$ (monotonicity), for every $u \in Z_l^{i+1} \cap Z_l^i$, we have $u \in [n-i-1, n-1]$ and $\{\mathtt{p}\}^{n-u} \in \mathrm{L}(\mathcal{A}, q_l)$ (since $[n-i, n-1] \subseteq [n-i-1, n-1]$). Similarly, if $u \in [n-i, n-1]$ and $\{\mathtt{p}\}^{n-u} \in \mathrm{L}(\mathcal{A}, q_l)$, then $u \in Z_l^i$ by induction hypothesis and therefore $u \in Z_l^{i+1}$. Hence, it remains to show that $u \in (Z_l^{i+1} \setminus Z_l^i)$ iff $u = n-i-1$ and $\{\mathtt{p}\}^{n-u} \in \mathrm{L}(\mathcal{A}, q_l)$ (i.e. $\{\mathtt{p}\}^{i+1} \in \mathrm{L}(\mathcal{A}, q_l)$). By (†), it is sufficient to show that $(n-i-1) \in Z_l^{i+1}$ iff $\{\mathtt{p}\}^{i+1} \in \mathrm{L}(\mathcal{A}, q_l)$.

The propositions below are equivalent ($l \in [1, \alpha]$, $i \geq 1$, $n-i-1 \geq 0$):

- $(n-i-1) \in Z_l^{i+1}$,
- $\sigma_n, n-i-1 \models_{f_\emptyset[\mathtt{z}_1 \leftarrow Z_1^i, \ldots, \mathtt{z}_\alpha \leftarrow Z_1^i]} \psi_l^0$ (by definition of $\mathcal{F}_{f_\emptyset,\sigma_n}^{i+1}$),
- $\sigma_n, n-i-1 \models_{f_\emptyset[\mathtt{z}_1 \leftarrow Z_1^i, \ldots, \mathtt{z}_\alpha \leftarrow Z_1^i]} \mathtt{p} \wedge \delta(q_i, \{\mathtt{p}\})[q_j \leftarrow \mathtt{X}\mathtt{z}_j, q_f \leftarrow \mathtt{X}\neg\mathtt{p}]$ (by definition of ψ_l^0),
- $\sigma_n, n-i-1 \models \mathtt{p}$ and there is a Boolean valuation $v : Q \rightarrow \{\perp, \top\}$ such that
  1. for every $q_{l'} \in (Q \setminus F)$, we have $v(q_{l'}) = \top$ iff $n-i \in Z_{l'}^i$,
  2. for every $q_f \in F$, $v(q_f) = \perp$,

  $v \models \delta(q_l, \{\mathtt{p}\})$ (by definition of $\models$ and $i \geq 1$),
- there is $v : Q \rightarrow \{\perp, \top\}$ such that
  1. for every $q_{l'} \in (Q \setminus F)$, we have $v(q_{l'}) = \top$ iff $n-i \in [n-i, n-1]$ and $\langle q_{l'}, \{\mathtt{p}\}^i \rangle \in \mathit{Acc}$,
  2. for every $q_f \in F$, $v(q_f) = \perp$,

  and $v \models \delta(q_l, \{\mathtt{p}\})$ (by induction hypothesis and since $n-i-1 \in [0, n-1]$),
- there is $v : Q \rightarrow \{\perp, \top\}$ such that
  1. for every $q_{l'} \in (Q \setminus F)$, we have $v(q_{l'}) = \top$ iff $\langle q_{l'}, \{\mathtt{p}\}^i \rangle \in \mathit{Acc}$,

2. for every $q_f \in F$, $v(q_f) = \perp$

and $v \models \delta(q_l, \{\mathtt{p}\})$ (by propositional reasoning),

– there is $v : Q \rightarrow \{\perp, \top\}$ such that

1. for every $q_{l'} \in (Q \setminus F)$, we have $v(q_{l'}) = \top$ iff $\langle q_{l'}, \{\mathtt{p}\}^i \rangle \in Acc$,
2. for every $q_f \in F$, $v(q_f) = \top$ iff $\langle q_f, \{\mathtt{p}\}^i \rangle \in Acc$,

and $v \models \delta(q_l, \{\mathtt{p}\})$ (since $i \geq 1$, $\delta(q_f, \{\mathtt{p}\}) = \perp$ and $\langle q_f, \{\mathtt{p}\}^i \rangle \notin Acc$),

– $\langle q_l, \{\mathtt{p}\}^{i+1} \rangle \in Acc$ (by definition of Acc),

– $\{\mathtt{p}\}^{i+1} \in \mathrm{L}(\mathcal{A}, q_l)$.

Thus, for every $i \in [1, n]$, the ith iterated tuple $\langle Z^i_1, \ldots, Z^i_\alpha \rangle$ verifies that for every $l \in [1, \alpha]$, $u \in Z^i_l$ iff $u \in [n - i, n - 1]$ and $\{\mathtt{p}\}^{n-u} \in \mathrm{L}(\mathcal{A}, q_l)$. So, $\{\mathtt{p}\}^n \in \mathrm{L}(\mathcal{A})$ iff $0 \in Z^n_1$. Since $\langle Z^\mu_1, \ldots, Z^\mu_\alpha \rangle$ is precisely equal to $\langle Z^n_1, \ldots, Z^n_\alpha \rangle$ because of the simple structure of σ_n (see (†), we conclude that $\{\mathtt{p}\}^n \in \mathrm{L}(\mathcal{A})$ iff $\sigma_n, 0 \models \mu \langle \mathtt{z}_1, \ldots, \mathtt{z}_\alpha \rangle \ \langle \psi^0_1, \ldots, \psi^0_\alpha \rangle \cdot \mathtt{z}_1$, whence (I) holds.

From (III), we conclude that $\mathrm{L}(\mathcal{A})$ is non-empty iff there is $\{\mathtt{p}\} \cdot \{\mathtt{p}\}^{n_1} \cdot \emptyset^\omega$ in $\mathrm{L}(\mathtt{cps})$ with $\mathtt{cps} = \langle \{\mathtt{p}\} \cdot \{\mathtt{p}\}^* \cdot \emptyset^\omega, \top \rangle$ such that $\{\mathtt{p}\} \cdot \{\mathtt{p}\}^{n_1} \cdot \emptyset^\omega \models \phi_\mathcal{A}$. Since $\mathtt{cps}$ and $\phi_\mathcal{A}$ can be computed in logarithmic space in the size of $\mathcal{A}$, this provides a reduction from the nonemptiness problem for alternating finite automata with a singleton alphabet to the intersection non-emptiness problem with linear μ-calculus. Hence, the intersection non-emptiness problem is PSpace-hard (we use only a fixed constrained path schema and a formula without past-time operators and without greatest fixed-point operator). ☐

L Proof of Corollary 16

Proof. The proof takes advantage of a variant of Theorem 2 (whose proof is also based on developments from [5]) in which initial counter values are replaced by variables. Below, we prove the results for BA, which immediately leads to a similar result for ABA, ETL and μTL.

Let S be a flat counter system of dimension n built over atomic constraints in $at \cup ag_n$, q be a control state and A be a specification in BA (i.e. a Büchi automaton whose underlying constrained alphabet is $\langle at, ag_n, \Sigma \rangle$). A *parameterized* constraint path schema (PCPS) is defined as a constrained path schema except that the second argument (a guard) has also the free variables $\mathtt{z}_1, \ldots, \mathtt{z}_n$ dedicated to the initial counter values. Remember that a constrained path schema has already a constraint about the number of times loops are visited. In its parameterized version, this constraint expresses also a requirement on the initial counter values. Following the proof of Theorem 2, one can construct in exponential time a set X of parameterized constrained path schemas such that:

– Each parameterized constrained path schema $\mathtt{pcps}$ in X has an alphabet of the form $\langle at, ag_n, \Sigma' \rangle$ (Σ' may vary) and $\mathtt{pcps}$ is of polynomial size.
– Checking whether a parameterized constrained path schema belongs to X can be done in polynomial time.

– For every run ρ from $\langle q, \mathbf{v} \rangle$, there is a parameterized constrained path schema $\mathtt{pcps}$ and $w \in \mathrm{L}(\mathtt{pcps}[\mathbf{v}])$ such that $\rho \models w$ where $\mathtt{pcps}[\mathbf{v}]$ is the contrained path obtained from $\mathtt{pcps}$ by replacing the variables $\mathsf{z}_1, \ldots, \mathsf{z}_n$ by the counter values from $\mathbf{v}$.
– For every parameterized constrained path schema $\mathtt{pcps}$, for every counter values $\mathbf{v}$, for every $w \in \mathrm{L}(\mathtt{pcps}[\mathbf{v}])$, there is a run ρ from $\langle q, \mathbf{v} \rangle$ such that $\rho \models w$.

The existential Presburger formula $\phi(\mathsf{z}_1, \ldots, \mathsf{z}_n)$ has the form below

$$\bigvee_{\mathtt{pcps}=\langle \cdot, \psi \rangle \in X} \ \bigvee_{q_{init}, q, (l_k)^\omega \in \mathrm{L}(A'_q)} (\exists\ \mathsf{y}_1, \ldots, \mathsf{y}_M\ \exists\ \mathsf{x}_1, \ldots, \mathsf{x}_{k-1}$$

$$\psi_{q_{init},q}(\mathsf{y}_1, \ldots, \mathsf{y}_M) \wedge (\mathsf{y}_1 = \alpha_0^1 + \alpha_1^1 \mathsf{x}_1 + \cdots + \alpha_{k-1}^1 \mathsf{x}_{k-1}) \wedge \cdots$$
$$\ldots \wedge (\mathsf{y}_M = \alpha_0^M + \alpha_1^M \mathsf{x}_1 + \cdots + \alpha_{k-1}^M \mathsf{x}_{k-1}) \wedge \psi(\mathsf{x}_1, \ldots, \mathsf{x}_{k-1}, \mathsf{z}_1, \ldots, \mathsf{z}_n))$$

where

1. $\langle at, ag_n, \Sigma' \rangle$ is the alphabet of $\mathtt{pcps}$, $M = \mathrm{card}(\Sigma')$ and by the *nice subalphabet property*, there is a specification A' such that $\mathrm{L}(A') = \mathrm{L}(A) \cap (\Sigma')^\omega$.
2. q_{init} is an initial state of A' and q is a state of A'.
3. $\psi_{q_{init},q}(\mathsf{y}_1, \ldots, \mathsf{y}_M)$ is the quantifier-free Presburger formula for the Parikh image of finite words over the alphabet Σ' accepted by A' (viewed as a finite-state automaton) with initial state q_{init} and final state q. $\psi_{q_{init},q}(\mathsf{y}_1, \ldots, \mathsf{y}_M)$ is of polynomial size in the size of Büchi automaton.
4. $\mathtt{pcps} = \langle p_1(l_1)^* \cdots p_{k-1}(l_{k-1})^* p_k(l_k)^\omega, \psi(\mathsf{x}_1, \ldots, \mathsf{x}_{k-1}, \mathsf{z}_1, \ldots, \mathsf{z}_n) \rangle$.
5. For each letter a_j, we write α_0^M, ..., α_{k-1}^M to denote the natural numbers such that if each loop i in $\mathtt{pcps}$ is taken x_i times, then the letter a_j is visited $\alpha_0^j + \alpha_1^j \mathsf{x}_1 + \cdots + \alpha_{k-1}^j \mathsf{x}_{k-1}$ times along $p_1(l_1)^* \cdots p_{k-1}(l_{k-1})^* p_k$. Those coefficients can be easily computed from $p_1(l_1)^* \cdots p_{k-1}(l_{k-1})^* p_k$ (for instance α_1^j is the number of times the letter a_j is present in the first loop).
6. Finally, observe that checking whether $(l_k)^\omega \in \mathrm{L}(A'_q)$ where A'_q is defined as the specification A' in which the unique initial state is q, amounts to perform a nonemptiness test between two Büchi automata.

FO admits a similar proof but it is based on Theorem 6 (actually the proof is much simpler because the number of times loops can be visited depends essentially on a threshold value). For FO, it is sufficient to consider the formula below:

$$\bigvee_{\mathtt{pcps}=\langle p_1(l_1)^* \cdots p_{k-1}(l_{k-1})^* p_k(l_k)^\omega, \psi \rangle \in X} \ \bigvee_{\boldsymbol{y}\ s.t.\ p_1 l_1^{\boldsymbol{y}[1]} p_2 l_2^{\boldsymbol{y}[2]} \ldots l_{k-1}^{\boldsymbol{y}[k-1]} p_k l_k^\omega \models A'}$$

$$\exists\ \mathsf{y}_1 \cdots\ \mathsf{y}_{k-1}\ \psi(\mathsf{y}_1, \ldots, \mathsf{y}_{k-1}, \mathsf{z}_1, \ldots, \mathsf{z}_n) \wedge \psi_1 \wedge \cdots \wedge \psi_{k-1}$$

where

1. $\mathtt{pcps} = \langle p_1(l_1)^* \cdots p_{k-1}(l_{k-1})^* p_k(l_k)^\omega, \psi(\mathsf{x}_1, \ldots, \mathsf{x}_{k-1}, \mathsf{z}_1, \ldots, \mathsf{z}_n) \rangle$,

2. $\langle at, ag_n, \Sigma' \rangle$ is the alphabet of `pcps` and by the *nice subalphabet property*, there is a specification A' such that $\mathrm{L}(A') = \mathrm{L}(A) \cap (\Sigma')^\omega$.
3. the third generalized disjunction deals with $\boldsymbol{y} \in [0, 2^{\mathrm{size}(A')+1} + 1]^{k-1}$.
4. For $i \in [1, k-1]$, $\psi_i \stackrel{\text{def}}{=} (\mathsf{y}_1 = \alpha)$ if $\boldsymbol{y}[i] < 2^{\mathrm{size}(A')+1} + 1$ otherwise $\psi_i \stackrel{\text{def}}{=} (\mathsf{y}_i \geq 2^{\mathrm{size}(A')+1} + 1$.

□