On Finite Domains in First-Order Linear Temporal Logic

Denis Kuperberg[1], Julien Brunel[2](✉), and David Chemouil[2]

[1] TU Munich, Munich, Germany
[2] DTIM, Université fédérale de Toulouse, ONERA, Toulouse, France
julien.brunel@onera.fr

Abstract. We consider First-Order Linear Temporal Logic (FO-LTL) over linear time. Inspired by the success of formal approaches based upon finite-model finders, such as Alloy, we focus on finding models with finite first-order domains for FO-LTL formulas, while retaining an infinite time domain. More precisely, we investigate the complexity of the following problem: given a formula φ and an integer n, is there a model of φ with domain of cardinality at most n? We show that depending on the logic considered (FO or FO-LTL) and on the precise encoding of the problem, the problem is either NP-complete, NEXPTIME-complete, PSPACE-complete or EXPSPACE-complete. In a second part, we exhibit cases where the Finite Model Property can be lifted from fragments of FO to their FO-LTL extension.

Keywords: FO · LTL · Finite model property · Bounded satisfiability · Fragments

1 Introduction

1.1 Context

First-Order Logic (FO) has proven to be useful in numerous applications in computer science such as formal specification, databases, ontology languages, etc. It is particularly well-suited to reason about objects of a domain, their relations and the properties they satisfy. However, since "full" FO is undecidable, the formal *verification* of properties implies a relaxation of the problem *e.g.* considering less expressive fragments. Thus, one can restrict the specification language (*e.g.* Prolog) or impose some form of interaction for verification (*e.g.* theorem provers, proof assistants).

Another form of trade-off is to keep the whole logic and full automation but to rely on a sound but incomplete decision procedure. For instance, the Alloy Analyzer[1] for the Alloy [10] language (based upon relational first-order logic)

Research partly funded by ANR/DGA project Cx (ref. ANR-13-ASTR-0006) and by *fondation STAE* project BRIefcaSE.

[1] Available at http://alloy.mit.edu/alloy.

C. Artho et al. (Eds.): ATVA 2016, LNCS 9938, pp. 211–226, 2016.
DOI: 10.1007/978-3-319-46520-3_14

implements a *bounded-satisfiability* decision procedure. That is, the tool relies on a *finite-model finder*: it first bounds the number of objects in the domain and then runs a classical propositional SAT procedure [19]. Thanks to the performance of modern SAT engines, this approach has shown to be very efficient in practice to find counterexamples quickly when assessing specifications. This is one of the reasons for the success of Alloy, in the formal methods community [3,17,21].

However, in most software and systems specifications, one needs to represent the evolution of modeled entities along time. In Alloy, the common way to do so is to model time by adding a specific set of time instants [10], by giving axioms describing its structure (as traces for instance) and by adding an extra time parameter to every dynamic predicate. This is tedious and cumbersome, if not error-prone.

This shortcoming has long been identified and several propositions [7,16,20] have been made to extend Alloy with facilities for fancier modeling of *behavior*. Still, in all these approaches, the verification remains bounded (because the set of instants is, for instance). [6] makes a step further by implementing a bounded model-checking approach in Alloy allowing time loops. However, up to our knowledge, no Alloy extension leverages a temporal logic, such as LTL for instance, that enjoys a complete decision procedure. The idea of adding temporal logic to FO has been implemented in the tool TLA$^+$ [12], where the FO signature is that of ZFC, instead of the arbitrary signatures allowed in Alloy. These remarks led us to study the combination of FO and LTL, in particular to draw questions about the relation between the satisfiability of a FO-LTL formula and the fact that the first-order part of the model is finite. In the literature, the logic FO-LTL has drawn a lot of interest, for decidability questions as well as in database theory [2]. For instance [8,9] study decidable fragments, while [11,15] give incompleteness results.

1.2 Contributions

The first question we address here is that of the complexity of satisfiability for FO and FO-LTL when the FO part of the model is bounded (we call this problem BSAT(N) for a given bound N). We are interested in the cost in terms of algorithmic complexity of adding an LTL component to the FO specification language. In Sect. 3, we consider BSAT for FO and FO-LTL depending on whether the quantifier rank (*i.e.* the maximum number of nested quantifiers) of formulas is bounded and whether the bound on the domain is given in unary or binary encoding.

- For pure FO, BSAT(N) is NP-complete if the rank is bounded and if N is given in unary; NEXPTIME-complete otherwise. This case can admittedly be considered *folklore* but it seems it has not been published formally, so we detail it in the paper for the sake of completeness. We also provide detailed results, showing that formulas of rank 2 with unary predicates are sufficient for NEXPTIME-completeness.

- For FO-LTL, which has been less studied from the point of view of BSAT, we show that this division goes the same except that BSAT is PSPACE-complete in the first case and EXPSPACE-complete otherwise (recall that satisfiability for LTL alone is PSPACE-complete [14,18]). Again, rank 2 formulas with unary predicates are sufficient.

Secondly, since we are only interested in finite models of FO-LTL formulas, it is natural to study which fragments of FO-LTL enjoy the *finite model property* (FMP). Recall that a formula has the FMP if the existence of a model implies the existence of a *finite* one. Many fragments of FO have been shown to enjoy the FMP in the past decades [1,4].

- In Sect. 4.1, we show that any fragment of FO enjoying the FMP (as well as a mild assumption often met in practice) can be "lifted" as a fragment of FO-LTL using also **X** and **F** and still enjoying the FMP (provided the removal of the temporal operators leads back to the original FO fragment).
- We finally show in Sect. 4.2 that with temporal operators **U** or **G**, the FMP is lost, even with strong constraints on the way temporal operators interact with first-order quantifiers.

All these results provide a theoretical insight on the combination of LTL with bounded FO which may be useful in the context of decision procedures based upon SAT or SMT, or in formal methods such as extensions of Alloy or TLA$^+$. Another possible application may be in the analysis of *software product lines* [5] where various, related transition systems (which may be described using FO) represent a product family.

Due to space constraints, the detailed proofs for some results can be found in an extended version of this article available from the authors' homepages.

2 The Logic FO-LTL

In this section, we define precisely the logic FO-LTL and provide some elements on its expressiveness.

2.1 Syntax

Definition 1 (FO-LTL Syntax). *We define the syntax of FO-LTL in the standard way from the following elements (function symbols will also be considered in Sect. 4):*

- *a tuple of predicates $\mathcal{P} = (P_1, \ldots, P_k)$ (each of which is of any arity) which define relations, between elements of the system,* that can vary in time,
- *equality $=$ is considered as a particular binary predicate which is static,* i.e., *its value does not depend on time,*
- *an infinite countable set Var of variables,*
- *a finite set Const of constants, representing elements of the system,*
- *the Boolean connectives $\neg, \vee$,*

- *the existential quantifier $\exists x$ for each variable $x \in Var$,*
- *the temporal operators* $\mathbf{X}$ *(next) and* $\mathbf{U}$ *(until).*

We also add the usual syntactic sugar: $\top, \bot, \wedge, \forall, \Rightarrow, \mathbf{G}, \mathbf{F}, \mathbf{R}$, *where*

$$F\varphi = \top\mathbf{U}\varphi, \qquad G\varphi = \neg(F(\neg\varphi)), \qquad \psi\,\mathbf{R}\,\varphi = \neg(\neg\varphi\mathbf{U}\neg\psi).$$

Example 1. Let us consider $\mathcal{P} = \{OK, \mathit{fail}\}$, where OK is a nullary predicate, *i.e.*, an atomic proposition, and *fail* is an unary predicate. We can define the following formula: $\mathbf{G}(\exists x.\mathit{fail}(x) \Rightarrow \mathbf{F}\,\mathbf{G}\,\neg OK)$. Intuitively, it expresses that a local bug endangers the whole system and no recovery is possible: if one element of the system fails at some point, then later the system must enter a state where it is not OK and remain in this state forever.

2.2 Semantics

Variables and constants (and more generally terms if we consider functions) are interpreted over a domain D. We consider that the domain and the interpretation of variables and constants do not vary in time. Only the interpretation of predicates can change. The time domain considered throughout the paper is $\mathbb{N}$.

Definition 2 (FO-LTL Structure). *An interpretation structure of FO-LTL is a tuple $\mathcal{M} = (D, \sigma_{Const}, \rho)$ where:*

- *D is the domain,*
- *$\sigma_{Const} : Const \rightarrow D$ is a valuation for constants,*
- *$\rho = (P_1^i, \ldots, P_k^i)_{i \in \mathbb{N}}$ gives the semantics of each predicate in $\mathcal{P}$ at each instant $i \in \mathbb{N}$. If P_j is a l-ary predicate, then $P_j^i \subseteq D^l$ for each instant $i \in \mathbb{N}$.*

We now define the satisfaction of a formula in a structure, in which case the latter is called a *model* of the former.

Definition 3 (Satisfaction Relation). *Given a structure $\mathcal{M}$, we inductively define the satisfaction relation $\mathcal{M}, \sigma, i \models \varphi$, where σ maps free variables of φ to elements in D, and $i \in \mathbb{N}$ is the current point in time.*

For ease of reading, x and y stand for both variables and constants in this definition. Moreover, we use $\overline{\sigma}$ to denote the interpretation of both variables and constants: $\overline{\sigma}(x) = \sigma(x)$ if $x \in Var$ and $\overline{\sigma}(x) = \sigma_{Const}(x)$ if $x \in Const$.

- $\mathcal{M}, \sigma, i \models x = y$ *if* $\overline{\sigma}(x) = \overline{\sigma}(y)$
- $\mathcal{M}, \sigma, i \models P_j(x_1, \ldots, x_n)$ *if* $(\overline{\sigma}(x_1), \ldots, \overline{\sigma}(x_n)) \in P_j^i$
- $\mathcal{M}, \sigma, i \models \neg\varphi if \mathcal{M}, \sigma, i \not\models \varphi$
- $\mathcal{M}, \sigma, i \models \varphi \vee \psi$ *if* $\mathcal{M}, \sigma, i \models \varphi$ *or* $\mathcal{M}, \sigma, i \models \psi$
- $\mathcal{M}, \sigma, i \models \exists x.\varphi$ *if there exists* $a \in D$ *such that* $\mathcal{M}, \sigma[x \mapsto a], i \models \varphi$
- $\mathcal{M}, \sigma, i \models \mathbf{X}\,\varphi$ *if* $\mathcal{M}, \sigma, i+1 \models \varphi$
- $\mathcal{M}, \sigma, i \models \varphi\mathbf{U}\psi$ *if there exists* $j \geqslant i$ *such that* $\mathcal{M}, \sigma, j \models \psi$, *and for all* p *such that* $i \leqslant p < j$, *we have* $\mathcal{M}, \sigma, p \models \varphi$

A formula φ without free variables is satisfiable if and only if there exists a structure $\mathcal{M}$ such that $\mathcal{M}, \emptyset, 0 \models \varphi$, and in this case we just note $\mathcal{M} \models \varphi$. (The semantics with function symbols is defined in a similar straightforward way.)

Notice that FO-LTL can be viewed as a fragment of a first-order logic called 2FO, where quantifiers can range either over D or over time. It was shown in [11] that FO-LTL is strictly less expressive than 2FO, as opposed to the classical result that LTL and FO have the same expressive power over discrete time. Detailed definitions and examples regarding 2FO are provided in the extended version of this article.

3 Complexity of Bounded Satisfiability

We are interested in a problem occurring in practice, where a formula φ of FO or FO-LTL is given together with a bound N, and we want to check the existence of a model of φ with domain of size at most N. This problem is decidable, but its complexity is an interesting question that, as far as we know, has been overlooked (though the FO case can be considered unpublished folklore). We call this problem BSAT and we investigate its complexity for several variants. As explained earlier, this question is of practical interest given the success of formal methods based upon finite model-finding and considering possible temporal extensions of these.

To analyze the complexity of this problem in different settings, we first recall the usual notion of *(quantifier) rank* [13].

Definition 4 (Quantifier Rank). *The* (quantifier) rank *of a FO-LTL formula is defined by structural recursion as follows:*

- $\mathrm{rk}(x = y) = \mathrm{rk}(P(x_1, \ldots, x_k)) = 0$
- $\mathrm{rk}(\neg\varphi) = \mathrm{rk}(\mathbf{X}\,\varphi) = \mathrm{rk}(\varphi)$
- $\mathrm{rk}(\varphi \vee \psi) = \mathrm{rk}(\varphi \mathbf{U} \psi) = \max(\mathrm{rk}(\varphi), \mathrm{rk}(\psi))$
- $\mathrm{rk}(\exists x, \varphi) = 1 + \mathrm{rk}(\varphi)$.

We are interested in settings where the rank of formulas is bounded, or on the contrary any formula is allowed as input. Restricting rank to a certain bound is a natural assumption in practice, and allows a finer analysis of the parameterized complexity of the BSAT problem. As is standard practice, we write FO[k] (resp. FO-LTL[k]) for all FO (resp. FO-LTL) formulae of quantifier rank up to k.

This rank is not to be confused with the alternation depth, which increases only with alternations between $\forall$ and $\exists$ quantifiers (or in our case between $\exists$ and $\neg$). We chose here to use quantifier rank to reflect the limited number of variables specified in real-life examples by users, for instance using tools such as Alloy. Notice that bounding the quantifier rank does not trivialize the problem, because we allow arbitrary signatures (again, similarly to the Alloy syntax). We recall that most complexity results on logical formalisms in the literature are relative to fixed signatures.

The following theorem classifies the complexity of BSAT according to three parameters: FO alone versus FO-LTL, N given in unary or binary, and $\mathrm{rk}(\varphi)$ bounded or unbounded. Some of these results regarding FO may be part of folklore, but we reproduce them here for completeness.

Theorem 1. *We consider BSAT for three parameters: logic, encoding, bound on* $\mathrm{rk}(\varphi)$ (ranked)*. The corresponding complexities are given in the following table (N is the bound on the model size, k the bound on* $\mathrm{rk}(\varphi)$*):*

	N unary	N binary
FO$[k]$	*NP-complete*	*NEXPTIME-complete*
FO	*NEXPTIME-complete* (*even* $N = 2$)	*NEXPTIME-complete*
FO-LTL$[k]$	*PSPACE-complete*	*EXPSPACE-complete*
FO-LTL	*EXPSPACE-complete* (*even* $N = 2$)	*EXPSPACE-complete*

Proofs are given in the remaining of this very Sect. 3.

3.1 First-Order Logic

Lemma 1. *The BSAT(N) problem for FO$[k]$ with N in unary is NP-complete.*

Proof.

Membership in NP. We show membership in NP by polynomially reducing the problem to SAT. The reduction is informally described here, see the extended version of this article for the formal construction.

The input formula φ is turned into a quantifier-free formula φ' where quantifiers have been expanded: $\forall x$ (resp. $\exists x$) is replaced by $\bigwedge_{x\in[1,N]}$ (resp. $\bigvee_{x\in[1,N]}$). Constants are turned into integers in the same way, using an initial disjunction on their possible values.

We then turn φ' into a SAT instance φ'' by replacing every occurrence of predicate $R(\vec{a})$, where $\vec{a}$ is an integer vector, by a Boolean variable $x_{R,\vec{a}}$.

This reduction is polynomial because of the unary encoding of N and the bound on $\mathrm{rk}(\varphi)$, and preserves satisfiability.

NP-hardness. We now show that BSAT for unary FO$[k]$ is NP-hard.

We perform a reduction from SAT: given a SAT instance with variables $x_1, \dots x_n$, we build an instance of BSAT where $x_1, \dots x_n$ are predicates of arity 0. We can then ask for the existence of a structure of size 0 (or 1), and this will answer the SAT problem. Since we do not need any quantifier to reduce to SAT, we obtain NP-hardness even if the bound on the rank is 0. □

Lemma 2. *The BSAT(N) problem for FO$[k]$ with N in binary is NEXPTIME-complete if $k \geqslant 2$, even restricted to unary predicates. It is NP-complete for $k = 1$.*

Proof. The proof is only sketched here, the detailed version can be found in the extended version of this article. The idea is to reduce directly from a non-deterministic Turing Machine running in exponential time.

Given such a machine M together with an input word u, we want to build a formula φ of FO[2] describing the run of M over u, such that φ has a model of size at most N if and only if M accepts u in at most N steps. Variables in φ will be used to describe positions of the tape of M as well as time instants in the computation of M. For this, we use unary predicates to encode the bits of the cell position $p(x)$ and time instant $t(x)$ described by an element x of the domain. We additionally use predicates $a(x)$ for a in the alphabet of the machine, and $q(x)$ for q in the state space of the machine to specify the content of the cell $p(x)$ at time $t(x)$. To avoid using formulas of rank 3, we also introduce a predicate $a'(x)$ to say that cell $p(x)$ is labelled a at time $t(x)+1$. We can express that this encoding is sound, and specify the existence of an accepting run of the machine using a formula φ of rank 2. Since N can be specified in binary, and since $|\varphi|$ is polynomial in the size of M, we can show that φ has a model of size N if and only if M has an accepting run of size exponential (2^{n^k}) in its input of size n.

The fact that the problem is in NEXPTIME is proven similarly as in Lemma 1, and is shown for a more general version of the problem in Lemma 4.

On the contrary, if $k = 1$, we show that any satisfiable formula φ of rank 1 has a model of size at most $|\varphi|$, therefore it is in NP to verify the existence of such a model. NP-hardness follows from Lemma 1. □

Lemma 3. *The BSAT(N) problem for unranked FO with N in unary is NEXPTIME-hard, even for $N = 2$.*

Proof. We show that this case is also NEXPTIME-hard.

As before, let M be a non-deterministic Turing machine running in exponential time 2^{n^k}.

This time, we will use predicates of unbounded arity, and encode positions in the tape using binary code. We will actually need only two elements in the structure, named 0 and 1.

To state that a position of binary encoding $\vec{x}$ is labelled by a letter a (resp. a state q), we will use a predicate $a(\vec{x})$ (resp. $q(\vec{x})$) of arity n^k, where each coordinate of $\vec{x}$ is given as a distinct argument.

To mimic the previous proof, we need to be able to compare 2 positions, using a predicate $\vec{x} < \vec{y}$ of arity $2n^k$. Once this order is axiomatized, the reduction can be done as in the previous case.

Therefore, we will only give the relevant new material here, *i.e.* the axioms for $\leqslant$ of arity $2n^k$ being a total order. These axioms must all be of polynomial length in n, in order to keep the overall reduction polynomial.

We use $\forall\vec{x}$ as a shorthand for $\forall x_1, \forall x_2, \ldots, \forall x_{n^k}$. In this way, it suffices to rewrite the axioms of total order using vectors instead of elements. This keeps the size of axioms polynomial, making it grow only by a factor n^k. Note that this does not guarantee that $\leqslant$ describes the lexicographic order on vectors, in particular the first position could be any vector, but this is not a problem.

Replacing all variables by vectors in the previous proof yields the required reduction. □

The membership in NEXPTIME will be shown in the proof of Lemma 4.

Lemma 4. *The BSAT(N) problem for unranked FO with N in binary is in NEXPTIME.*

Proof. This result implies NEXPTIME-completeness for 3 variants of the First-Order BSAT problem.

Let $\varphi, \vec{e}$ be the input of the problem, where $\vec{e}$ is a binary encoding of N, so $N = O(2^{|\vec{e}|})$. Let $n = |\varphi| + |\vec{e}|$ be the size of the input, and $r = \mathrm{rk}(\varphi)$. The algorithm from the proof of Lemma 1 can be adapted as follows:

- Guess a structure and write it on the tape: a predicate of arity k takes up to N^k cells, so the operation uses time (and space) $O(2^{nk})$.
- Unfold quantifiers of the formula and check predicates. This operation takes time $O(|\varphi|N^r) = O(2^{nr})$.

Overall, the time complexity is in $O(2^{n(k+r)}) = O(2^{n^2})$, since both k and r are bounded by n.

This ends the proof that the most "difficult" FO case of BSAT still has NEXPTIME complexity. Hardness (even for $N = 2$) follows from Lemma 2. □

3.2 An Algorithm for the BSAT Problem for FO-LTL

We now turn to the BSAT problem for FO-LTL, and describe a generic algorithm that we will use for various settings of the problem.

Lemma 5. *The BSAT(N) problem for FO-LTL is in PSPACE if the rank is bounded and N is given in unary, and in EXPSPACE all three other cases.*

The algorithm consists in trying all sizes up to N, and for each of them expand the formula φ into a LTL formula, then use a PSPACE algorithm for LTL satisfiability.

Definition 5 (Expansion of an FO-LTL Formula). *Let us consider a finite domain D, a finite set of constants Const, a valuation $\sigma_{Const} : Const \to D$, a closed FO-LTL formula φ with constants in Const and predicate symbols $P_1, \ldots, P_k$, of arities $\alpha_1, \ldots, \alpha_k$ respectively. We define the expansion* $\exp(\varphi)$ *of φ given the domain D as an LTL formula, using alphabet $A = \{A_i(\vec{a}) \mid 1 \leqslant i \leqslant k, \vec{a} \in D^{\alpha_i}\}$ by induction on φ. We assume that φ can use elements of D as constants, and σ_{Const} is extended to these new constants in the natural way.*

$$\exp(a = b) = \top \textit{ if } \sigma_{Const}(a) = \sigma_{Const}(b) \textit{ and } \bot \textit{ otherwise}$$
$$\exp(P_i(a_1, \ldots, a_k)) = A_i(\sigma_{Const}(a_1), \ldots, \sigma_{Const}(a_k))$$
$$\exp(\neg\varphi) = \neg \exp(\varphi) \qquad \exp(\varphi \vee \psi) = \exp(\varphi) \vee \exp(\psi)$$
$$\exp(\mathbf{X}\,\varphi) = \mathbf{X} \exp(\varphi) \qquad \exp(\varphi \mathbf{U} \psi) = \exp(\varphi) \mathbf{U} \exp(\psi)$$
$$\exp(\exists x, \varphi) = \bigvee_{a \in D} \exp(\varphi[x \leftarrow a])$$

It is easy to show by induction that for any φ and D, we have

$$|\exp(\varphi)| = \Theta(|\varphi| \cdot |D|^{\mathrm{rk}(\varphi)}).$$

We can now adapt the algorithm from Lemma 1 to this new setting. In the case where the rank is bounded, we rewrite the formula to bound arity of predicates if the rank is bounded, and guess a structure of size D of size at most N together with σ_{Const}, using space polynomial in $|D|$ (so exponential in the input N is in binary). We then expand φ into $\exp(\varphi)$, of size $O(|\varphi| \cdot N^{\mathrm{rk}(\varphi)})$.

It remains to decide whether the LTL formula $\exp(\varphi)$ is satisfiable, which can be done using space polynomial in $|\exp(\varphi)|$ [18]. Therefore this algorithm uses space $O(|\varphi| \cdot N^{\mathrm{rk}(\varphi)})$. It is in PSPACE if the rank is bounded and N is in unary, and EXPSPACE in the other three cases.

3.3 Completeness Results for FO-LTL BSAT

We now show that this algorithm is optimal, by showing that BSAT for FO-LTL is either PSPACE-hard or EXPSPACE-hard depending on the setting.

Lemma 6. *The BSAT(N) problem for ranked and unranked FO-LTL with N in binary is EXPSPACE-complete, even for $N = 2$. In the ranked case, the bound must be at least 2. The BSAT(N) problem for FO-LTL[k] with N in unary is PSPACE-complete.*

Proof. The main idea of the proof is to directly encode the run of a Turing machine using exponential space (polynomial space in the ranked case with N in unary), similarly as in the proof of Lemma 2. The main difference is that we now have additional LTL operators, that allow us to encode computation steps without any bound on the number of time instants. Therefore, the first-order domain D will only be used to encode positions of the tape via unary predicates specifying the bits of the position, and that is why we can now encode runs of machines using exponentially more time than space. The detailed reduction can be found in the extended version of this article. □

Finally, the last case is treated in the following lemma.

Lemma 7. *The BSAT(N) problem for unranked FO-LTL with N in unary is EXPSPACE-complete.*

Proof. We will show that this case is also EXPSPACE-hard, although we can no longer use an element of the structure for each cell of the Turing machine.

We can reuse ideas from Lemma 3, and encode positions using vectors of bits with predicates of unbounded arities. This time, only positions will be encoded this way, as time can be taken care of by LTL. Thus we can start from a machine where only space is exponentially bounded, and time can be doubly exponential.

The construction is then similar to the one from Lemma 3, and yields a reduction showing that this variant of BSAT is also EXPSPACE-complete, even for structures with only 2 elements. □

Other examples of EXPSPACE-complete problems related to deciding small fragments of FO-LTL can be found in [8].

4 Finite Model Property

Since we are only interested in finite models of FO-LTL formulas, it is natural to study which fragments of FO-LTL enjoy the *finite model property* (FMP). We say that a formula has the FMP if the existence of a model implies the existence of a *finite* model (*i.e.*, with finite first-order domain but still infinite time structure). We also say that a fragment *Frag* of some logic has the FMP if all the formulas from *Frag* have the FMP. Many such fragments of FO were exhibited in the past decades.

Function Symbols. In this Section we will enrich the syntax of FO-LTL with function symbols. Each function has an arbitrary arity like a predicate, but yields a *term*, which will be interpreted as an element of the domain, as variables and constants. In this case, the parameters of the predicates (including equality) can be arbitrary terms, built by composing variables and constants with functions. For instance, if x and y are variables, a is a constant, f and g are functions, then $f(x, g(x), a) = g(y)$ is a formula.

Example 2 [1,4]. The following fragments of *FO*, named following the notation of [4], have the FMP:

- $[\exists^*\forall^*, all]_=$ (Ramsey 1930) the class of all sentences with quantifier prefix $\exists^*\forall^*$ over arbitrary relational vocabulary with equality.
- $[\exists^*\forall\exists^*, all]_=$ (Ackermann 1928) the class of all sentences with quantifier prefix $\exists^*\forall\exists^*$ over arbitrary relational vocabulary with equality.
- $[\exists^*\forall^2\exists^*, all]$ (Gödel 1932, Kalmár 1933, Schütte 1934) the class of all sentences with quantifier prefix $\exists^*\forall^2\exists^*$ over arbitrary relational vocabulary without equality.
- $[\exists^*, all, all]_=$ (Gurevich 1976) the class of all sentences with quantifier prefix $\exists^*$ over arbitrary vocabulary with equality.
- $[\exists^*\forall, all, (1)]_=$ (Grädel 1996) the class of all sentences with quantifier prefix $\exists^*\forall$ over vocabulary that contains one unary function and arbitrary predicate symbols with equality.
- $[all, (\omega), (\omega)]$ (Gurevich 1969, Löb 1967) the class of all sentences with arbitrary quantifier prefix over vocabulary that contains an arbitrary number of unary predicates and unary functions without equality
- FO_2 (Mortimer 1975) the class of all sentences of relational vocabulary that contains two variables and equality.

4.1 Lifting FMP from FO to FO-LTL

In this section, we first present general results that allow to lift the finite model property from FO fragments to their temporal extension with operators **X** and **F**. Then, we focus on two particular fragments: the well known Ramsey fragment, for which the extension can be generalized to full LTL, and a fragment that does not fulfill the hypotheses of our general result, but for which the temporal extension with operators **X** and **F** still has the FMP.

Remark 1. In the following, we will only consider formulas in *negation normal form* (NNF), *i.e.* where negations have been pushed to the leaves. This means negations can only be applied to predicates. Notice that the syntactic sugar mentioned in Sect. 2.1, in particular the operator $\mathbf{R}$ (dual of $\mathbf{U}$) now becomes necessary to retain full expressiveness.

Definition 6. *If Frag is a fragment of FO, and OP $\subseteq \{\mathbf{X}, \mathbf{F}, \mathbf{G}, \mathbf{U}, \mathbf{R}\}$ is a set of temporal operators, we define the fragment Frag + OP of FO-LTL as the formulas with temporal operators from OP, where the formula(s) obtained by removing unary temporal operators and replacing binary ones by $\vee$ or $\wedge$ (indifferently), is in Frag.*

A General Extension Result for Fragments with the FMP

Definition 7 ((Plus-)Replacement of a Formula). *If φ, ψ are FO-formulas, we say that ψ is a* replacement *of φ if ψ can be obtained from φ by* replacing *predicates and functions,* i.e.*, by allowing different occurrences of the same predicate (resp. function) of φ to become distinct predicates (resp. functions) of same arity in ψ, but distinct predicates (resp. functions) in φ are always mapped to distinct predicates (resp. functions) in ψ.*

Additionally, we define the notion of plus-replacement *where the new predicates and functions can have increased arity.*

For instance $\forall x.\exists y.P(x) \vee Q(y)$ is a replacement of $\forall x.\exists y.P(x) \vee P(y)$. Likewise, the formula $\forall x.\exists y.P(x) \vee Q(y,x)$ is a plus-replacement of $\forall x.\exists y.P(x) \vee P(y)$.

Definition 8 (Stability Under (Plus-)Replacement). *We say that a fragment Frag of FO with FMP is* stable under replacement *(resp.* plus-replacement*) if for all $\varphi \in$ Frag and for all replacement (resp.* plus-replacement*) ψ of φ, we have that ψ has the FMP.*

In practice, many fragments with FMP considered in the literature are stable under (plus-)replacement. This is for example the case for most of the fragments from Example 2 (see Corollary 1).

Theorem 2 (*Frag* + X). *Let Frag be a fragment of FO with the FMP, stable under replacement. Then the fragment Frag + $\mathbf{X}$ of FO-LTL has the FMP.*

The proof of this theorem is presented in the extended version of this article. The following theorem, along the same lines, allows more temporal operators but has the stronger assumption of plus-replacement.

Theorem 3 (*Frag* + {X, F}). *Let Frag be a fragment of FO with FMP, stable under plus-replacement. Then Frag + $\{\mathbf{X}, \mathbf{F}\}$ also has the FMP.*

Proof. Let φ be a satisfiable formula of *Frag* + $\{\mathbf{X}, \mathbf{F}\}$. Let V be the set of variables used in φ and $\{\mathbf{F}_j, j \in J\}$ be the set of $\mathbf{F}$-operators in φ, for some finite labeling set $J = \{1, 2, \ldots, |J|\}$ such that if $\mathbf{F}_j$ is under the scope of $\mathbf{F}_i$ then $i < j$.

For $\vec{x}$ a list of variables from V, $j \in J \cup \{0\}$, $k \in \mathbb{N}$, and θ a subformula of φ, we define $[\![\theta]\!]^{j}_{\vec{x}}$ inductively as follows:

$$\begin{array}{ll}
[\![P(\vec{y})]\!]^{j,k}_{\vec{x}} = P_{j,k}(\vec{y},\vec{x}) \quad \text{variables can appear in both } \vec{y} \text{ and } \vec{x} & \\
[\![f(\vec{y})]\!]^{j,k}_{\vec{x}} = f_{j,k}(\vec{y},\vec{x}) \quad \text{variables can appear in both } \vec{y} \text{ and } \vec{x} & \\
[\![\exists z.\theta(\vec{y})]\!]^{j,k}_{\vec{x}} = \exists z [\![\theta(\vec{y})]\!]^{j,k}_{\vec{x}} & [\![\forall z.\theta(\vec{y})]\!]^{j,k}_{\vec{x}} = \forall z [\![\theta(\vec{y})]\!]^{j,k}_{\vec{x},z} \\
[\![\theta(\vec{y}) \wedge \theta'(\vec{y}')]\!]^{j,k}_{\vec{x}} = [\![\theta(\vec{y})]\!]^{j,k}_{\vec{x}} \wedge [\![\theta'(\vec{y}')]\!]^{j,k}_{\vec{x}} & \\
[\![\theta(\vec{y}) \vee \theta'(\vec{y}')]\!]^{j,k}_{\vec{x}} = [\![\theta(\vec{y})]\!]^{j,k}_{\vec{x}} \vee [\![\theta'(\vec{y}')]\!]^{j,k}_{\vec{x}} & \\
[\![\mathbf{X}\,\theta(\vec{y})]\!]^{j,k}_{\vec{x}} = [\![\theta(\vec{y})]\!]^{j,k+1}_{\vec{x}} & [\![\mathbf{F}_{j'}\,\theta(\vec{y})]\!]^{j,k}_{\vec{x}} = [\![\theta(\vec{y})]\!]^{j',0}_{\vec{x}}
\end{array}$$

To sum up, we index predicates and functions by the label j of the innermost occurrence of $\mathbf{F}$ that has it in its scope, as well as the number k of nested $\mathbf{X}$ since this occurrence. We also add all universally quantified variables as arguments. We additionally remove $\mathbf{F}$ and $\mathbf{X}$ operators in the process.

Let $\psi = [\![\varphi]\!]^{0,0}_{\emptyset}$. We show that ψ is satisfiable. Let $\mathcal{M} = (D, \rho)$ be a model of φ. This means that for each subformula $\mathbf{F}_j\,\theta(\vec{y})$ of φ under universally quantified variables $\vec{x}$, there is a function $t_j : D^{|\vec{x}|} \to \mathbb{N}$ such that $\theta(\vec{y})$ is true at time $t_j(\vec{x})$. We build a model of ψ by setting the value of $P_{j,k}(\vec{y},\vec{x})$ to $P(\vec{y})$ at time $t_j(\vec{x}) + k$, where $\vec{x}$ is the list of new arguments of P_j (and same with functions). It is straightforward to verify that this is indeed a model of ψ.

Let φ' be φ where the $\mathbf{F}$'s and $\mathbf{X}$'s have been removed, by definition we have $\varphi' \in \mathit{Frag}$. Since ψ is a plus-replacement of φ' and *Frag* is stable under plus-replacement, we have ψ has the FMP. Since ψ is satisfiable, there exists a finite model M_f of ψ. Finally, we build from M_f a finite model of φ. For this, we have to choose new values for the $t_j(\vec{x})$, so that no conflicts occur: if $(j,k,\vec{x}) \neq (j',k',\vec{x}')$, then $t_j(\vec{x}) + k \neq t_j(\vec{x}') + k'$. Let K be the maximal number of nested $\mathbf{X}$ (not interleaved with $\mathbf{F}$), and $(\vec{x}_i)_{0 \leqslant i \leqslant R}$ be an ordering of all possible vectors $\vec{x}$. We choose $t_j(\vec{x}) = (K+1) \times (R^j + i)$, in order to satisfy the injectivity condition: for all $j, j' \in [0, |J|], k, k' \in [0, K]$, and $\vec{x}, \vec{x}' \in \{\vec{x}_i | 0 \leqslant i \leqslant R\}$, we have $t_j(\vec{x}) + k = t_{j'}(\vec{x}') + k'$ if and only if $(j,k,\vec{x}) = (j',k',\vec{x}')$. Notice moreover that we respect the condition that if $\mathbf{F}_j\,\varphi_j$ is a subformula of $\mathbf{F}_i\,\varphi_i$, then $j > i$ and thus for any value of $\vec{x}, \vec{y}$, we have $t_j(\vec{x}) > t_i(\vec{y})$.

Finally, we build a finite model of φ by setting the value of $P(\vec{y})$ (resp. $f(\vec{y})$) at time i to $P_{j,k}(\vec{y},\vec{x})$ (resp. $f_{j,k}(\vec{y},\vec{x})$) if $i = t_j(\vec{x}) + k$ for some $j, k, \vec{x}$, and choosing any values for other time instants.

So φ has a finite model and therefore $\mathit{Frag} + \{\mathbf{X}, \mathbf{F}\}$ has the FMP. □

Remark 2. It is enough to consider plus-replacement where new arguments are only quantified universally, which is a weaker condition.

Corollary 1. *The following FO-LTL fragments, extending FO fragments mentioned in Example* 2*, have the FMP:*

$$\begin{array}{ll}
[\exists^*\forall^*, \mathit{all}]_= + \{\mathbf{X}, \mathbf{F}\} & [\exists^*\forall\exists^*, \mathit{all}]_= + \{\mathbf{X}, \mathbf{F}\} \\
[\exists^*\forall^2\exists^*, \mathit{all}] + \{\mathbf{X}, \mathbf{F}\} & [\exists^*, \mathit{all}, \mathit{all}]_= + \{\mathbf{X}, \mathbf{F}\} \\
FO_2 + \{\mathbf{X}, \mathbf{F}\} &
\end{array}$$

Specific Extensions for Two Fragments. In this section, we focus on two fragments of FO: a fragment for which our general result (Theorem 3) does not apply and a fragment for which our general result can be extended to full LTL.

The FMP of the following fragment, even if it is not stable under plus-replacement, can be lifted to its temporal extension with $\mathbf{X}$ and $\mathbf{F}$.

Theorem 4. $[all, (\omega), (\omega)] + \{\mathbf{X}, \mathbf{F}\}$ *has the FMP.*

Proof. We show that any formula of this fragment has the FMP. We proceed by induction on the number of nested $\mathbf{F}$. The induction hypothesis is actually stronger than the FMP: we show by induction that for such a formula φ, if there is a model then there is a model M with finite domain D and a finite set of time instants T such that M only "looks at T", i.e. changing the values of predicates and functions outside of T does not change the truth value of φ.

We start with the base case where there is no $\mathbf{F}$. By Theorem 2 (and its proof), and since $[all, (\omega), (\omega)]$ is stable under replacement (even though it is not stable under plus-replacement), if φ has a model it has a finite one where only the values on a finite set of instants matter.

We now turn to the induction step, and consider a formula φ with $n+1$ nested $\mathbf{F}$. By considering the outermost occurrences of $\mathbf{F}$, the formula φ can be written $\varphi'(\mathbf{F}\,\varphi_1, \mathbf{F}\,\varphi_2, \ldots, \mathbf{F}\,\varphi_k)$, where φ' contains no $\mathbf{F}$ but may contain quantifiers, and for every $i \in [1, k]$, φ_i has at most n nested $\mathbf{F}$ and may contain free variables.

We assume that φ has a model M, and without loss of generality we note j the index in $[1, k]$ such that $\mathbf{F}\,\varphi_i$ is true in M (for at least one valuation of its free variables) if and only if $i \leqslant j$. This means in particular that for all $i \leqslant j$, φ_i has a model. By the induction hypothesis, for all $i \leqslant j$, φ_i can be set to true in a model M_i with a finite domain D_i and that only looks at a finite set of instants T_i.

Moreover, $\varphi'' = \varphi'(\top, \ldots, \top, \bot, \ldots, \bot)$ (with j times $\top$) is satisfiable, and by the base case has a model M' with a finite domain D' that only looks at a finite set of instants T' (that will be used as the first instants of the model).

We now build a model M_f for φ with a finite domain D, that we define as a set of cardinality $\max(|D'|, |D_1|, |D_2|, \ldots, |D_j|)$.

We define a sequence of time instants $(t_i)_{1 \leqslant i \leqslant j}$ such that at time t_i the formula φ_i is true for a particular valuation of its free variables, and at $t_i + |T_i|$, it is true for another valuation of its free variables that are universally quantified, and so on, until we have considered all the possible values in D for these universally quantified variables. So we define the t_i inductively as follows: $t_1 = |T'|$ and $t_i = t_{i-1} + |T_{i-1}| \times |D|^r$, where r is the number of nested universal quantifiers in φ'.

We now describe the predicates and function values in M_f. At times $t \in [0, t_1 - 1]$, we mimic the model M'. This gives the value of predicates and functions for $|D'|$ elements of D. All the remaining elements can be set to behave as any element of D', for instance the first one. Since equality cannot be tested, and predicates and functions are monadic, the truth value of φ'' is preserved.

For all $i \in [1, j]$, we use M_i to fix the valuation of predicates and functions at times $[t_i, t_i + |T_i| - 1]$. Then, from $t_i + |T_i|$, we consider another possible assignment of universally quantified variables and define the valuation of predicates and functions accordingly. This way, we obtain a model of φ_i starting at time t_i, and therefore a model of $\mathbf{F}\,\varphi_i$ starting at time 0.

Since φ' is monotonous in its arguments (no $\mathbf{F}$ can be under a negation), and we preserved the value $\top$ for all $\mathbf{F}\,\varphi_i$ with $i \leqslant j$, the truth value of φ on M_f is that of φ', which is true thanks to the valuations on $[0, t_0]$.

We have therefore built a model M_f of φ with finite domain, and only looking at a finite number of time instants. □

The result of Ramsey that the $\exists^*\forall^*$ fragment has the FMP is generalized in the following theorem. See the extended version of this article for a proof, omitted here due to space constraints.

Theorem 5. *We consider here FO-LTL without function symbols. Let* $\varphi = \exists x_1 \ldots \exists x_n . \psi(x_1, \ldots, x_n)$*, where* ψ *is a FO-LTL formula without any* $\exists$ *quantifiers. Then if* φ *is satisfiable, it has a model with domain of size at most* $n + c$*, where* c *is the number of constants in the vocabulary.*

4.2 Axioms of Infinity Using LTL

We now give examples showing that adding LTL to fragments of FO with the FMP allows to write axioms of infinity, therefore losing the FMP. This holds even when strong restrictions are enforced on the way LTL operators interact with first-order quantifiers.

Extending the Ramsey Fragment. First, let us remark that the constraint from Theorem 5 that existential quantifiers are not under the scope of temporal operators is necessary, as showed by the following formula which is only satisfiable by infinite models, using a unary predicate P:

$$\mathbf{G}(\exists x . P(x) \wedge \mathbf{X}\,\mathbf{G}\,\neg P(x)).$$

Indeed, it is straightforward to show that a different x_n is needed to satisfy the formula at each different time instant $n \in \mathbb{N}$, as the condition on the predicate P guarantees that the same x can never be used twice.

Separating Quantifiers and LTL. We now give examples where a fragment of FO loses the FMP when extended with LTL, even without nesting quantifiers under temporal operators.

The following FO-LTL formula is an axiom of infinity with a $\forall\exists$ quantifier prefix, and where no first-order quantifiers are under the scope of LTL operators. It uses one constant c and one unary predicate P:

$$\forall x \exists y . P(c) \wedge \mathbf{G}(P(x) \Rightarrow \mathbf{X}(P(y) \wedge \mathbf{G}\,\neg P(x))).$$

This sentence only has infinite models, as the predicate P must be true on a different element at each instant of time. However, as recalled in Example 2, in FO without LTL, if only one quantifier $\forall$ is used the FMP is guaranteed (or alternatively, formulas with two variables also have the FMP).

This example can actually be replaced using $\mathbf{U}$ instead of $\mathbf{G}$, showing that it suffices to be able to refer to an "unbounded" (as opposed to "infinite") number of time instants to force models to be infinite, as showed by the following example:

$$\forall x \exists y. P(c) \wedge ((P(x) \wedge P(y))\mathbf{U}(\neg P(x) \wedge P(y))).$$

This time, we used values of the predicate P in the past instead of the future to guarantee that the same x cannot be used twice.

5 Conclusion

Motivated by the possible extension of formal methods based upon finite model finding (such as Alloy or various decision procedures based upon SAT or SMT techniques) with temporal reasoning, we have investigated FO-LTL with finite FO domains in two ways: (1) we studied the complexity of the satisfiability for FO-LTL (and FO alone) when the FO part of the model is bounded; (2) we studied cases where we can lift the FMP of FO fragments to their temporal extensions.

Several question are still open. On the complexity side, it remains to settle the case of FO-LTL[1] with N in binary. Related to the FMP, even if we showed in Sect. 4.1 that for a particular FO fragment that is not stable under plus-replacement, the FMP can still be lifted to its temporal extension with operators $\mathbf{X}$ and $\mathbf{F}$, it is not clear whether this assumption can be dropped in Theorem 3. Another open question is whether we can find a reasonable condition under which we can extend an FO fragment with temporal operators $\mathbf{G}$ or $\mathbf{U}$ without losing the FMP. Indeed, these operators bring an expressiveness that is very useful in practice but we showed in Sect. 4.2 that they behave badly with respect to the FMP.

References

1. Abadi, A., Rabinovich, A., Sagiv, M.: Decidable fragments of many-sorted logic. In: Dershowitz, N., Voronkov, A. (eds.) LPAR 2007. LNCS (LNAI), vol. 4790, pp. 17–31. Springer, Heidelberg (2007). doi:10.1007/978-3-540-75560-9_4
2. Abiteboul, S., Herr, L., den Bussche, J.V.: Temporal versus first-order logic to query temporal databases. In: Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 3–5 June 1996, Montreal, Canada, pp. 49–57 (1996)
3. Bagheri, H., Kang, E., Malek, S., Jackson, D.: Detection of design flaws in the android permission protocol through bounded verification. In: Bjørner, N., de Boer, F. (eds.) FM 2015. LNCS, vol. 9109, pp. 73–89. Springer, Heidelberg (2015). doi:10.1007/978-3-319-19249-9_6

4. Börger, E., Grädel, E., Gurevich, Y.: The Classical Decision Problem. Perspectives in Mathematical Logic. Springer, Heidelberg (1997)
5. Classen, A., Heymans, P., Schobbens, P., Legay, A., Raskin, J.: Model checking lots of systems: efficient verification of temporal properties in software product lines. In: ICSE 2010, pp. 335–344. ACM (2010)
6. Cunha, A.: Bounded model checking of temporal formulas with alloy. In: Ait Ameur, Y., Schewe, K.-D. (eds.) ABZ 2014. LNCS, vol. 8477, pp. 303–308. Springer, Heidelberg (2014)
7. Frias, M.F., Galeotti, J.P., Pombo, C.L., Aguirre, N.: DynAlloy: upgrading alloy with actions. In: ICSE, vol. 2005, pp. 442–451 (2005)
8. Hodkinson, I.M., Kontchakov, R., Kurucz, A., Wolter, F., Zakharyaschev, M.: On the computational complexity of decidable fragments of first-order linear temporal logics. In: TIME-ICTL, vol. 2003, pp. 91–98 (2003)
9. Hodkinson, I.M., Wolter, F., Zakharyaschev, M.: Decidable fragments of first-order temporal logics. Ann. Pure Appl. Logic **106**(1–3), 85–134 (2000)
10. Jackson, D.: Software Abstractions - Logic, Language, and Analysis. MIT Press, Cambridge (2006). http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=10928
11. Kamp, H.W.: Tense logic and the theory of linear order. Ph.D. thesis, University of Warsaw (1968)
12. Lamport, L.: Specifying Systems, the TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley, Boston (2002)
13. Libkin, L.: Elements of Finite Model Theory. Texts in Theoretical Computer Science. An EATCS Series. Springer, Heidelberg (2004). http://dx.doi.org/10.1007/978-3-662-07003-1
14. Lichtenstein, O., Pnueli, A.: Checking that finite state concurrent programs satisfy their linear specification. In: Proceedings of the 12th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, pp. 97–107. ACM (1985)
15. Merz, S.: Decidability and incompleteness results for first-order temporal logics of linear time. J. Appl. Non-Class. Logics **2**(2), 139–156 (1992)
16. Near, J.P., Jackson, D.: An imperative extension to alloy. In: Frappier, M., Glässer, U., Khurshid, S., Laleau, R., Reeves, S. (eds.) ABZ 2010. LNCS, vol. 5977, pp. 118–131. Springer, Heidelberg (2010). doi:10.1007/978-3-642-11811-1_10
17. Newcombe, C., Rath, T., Zhang, F., Munteanu, B., Brooker, M., Deardeuff, M.: How Amazon web services uses formal methods. Commun. ACM **58**(4), 66–73 (2015). http://doi.acm.org/10.1145/2699417
18. Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. J. ACM **32**(3), 733–749 (1985)
19. Torlak, E., Jackson, D.: Kodkod: a relational model finder. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 632–647. Springer, Heidelberg (2007). doi:10.1007/978-3-540-71209-1_49
20. Vakili, A., Day, N.A.: Temporal logic model checking in alloy. In: Derrick, J., Fitzgerald, J., Gnesi, S., Khurshid, S., Leuschel, M., Reeves, S., Riccobene, E. (eds.) ABZ 2012. LNCS, vol. 7316, pp. 150–163. Springer, Heidelberg (2012). doi:10.1007/978-3-642-30885-7_11
21. Zave, P.: Using lightweight modeling to understand Chord. SIGCOMM Comput. Commun. Rev. **42**(2), 49–57 (2012). http://doi.acm.org/10.1145/2185376.2185383