Past is for Free: on the Complexity of Verifying Linear Temporal Properties with Past

Nicolas Markey

Département d'Informatique
Université Libre de Bruxelles
Boulevard du Triomphe – CP 212
1050 Bruxelles – BELGIUM

nmarkey@ulb.ac.be

Abstract We study the complexity of satisfiability and model checking problems for fragments of linear-time temporal logic with past (PLTL). We consider many fragments of PLTL, obtained by restricting the set of allowed temporal modalities, the use of negations or the nesting of future formulas into past formulas. Our results strengthen the widely accepted fact that "past is for free", in the sense that allowing symmetric past-time modalities does not bring additional theoretical complexity. This result holds even for small fragments and even when nesting future formulas into past formulas.

Introduction

Temporal logics. In 1977, Pnueli introduced *temporal logics* as a tool for reasoning about concurrent programs [29]. Those logics provide powerful methods for specifying and verifying properties of reactive systems. We refer to [7, 9, 25, 26] for more motivations and background.

Linear-time propositional temporal logic (called LTL) is the most used framework in this area: An LTL formula expresses properties about the ordering of events along a run of a system. For instance, the fact that, at all times, a `request` will eventually be granted can be expressed with:

$$\mathbf{G}\ (\mathtt{request} \Rightarrow \mathbf{F}\ \mathtt{grant}) \tag{S1}$$

Temporal logics with past. LTL is a *pure-future* temporal logic, *i.e.* a logic where modalities only refer to the future of the current state.

It is possible, however, to define *past-time modalities* [15,11,23]. For example, for expressing that a `grant` may only occur if some `request` *has been* issued, we would write

$$\mathbf{G}\ (\mathtt{grant} \Rightarrow \mathbf{F}^{-1}\ \mathtt{request}) \tag{S2}$$

It is well-known that past-time modalities do not increase the expressiveness of LTL [15,12]. In [11], Gabbay gives a method for translating LTL+Past formulas into equivalent pure-future LTL formulas. For instance, an equivalent pure-future formula for (S2) would be

$$\neg\big((\neg\mathtt{request})\ \mathbf{U}\ (\mathtt{grant} \wedge \neg\mathtt{request})\big) \tag{S3}$$

expressing that we cannot reach a `grant` without encountering a `request` in the meantime. By concern of minimality, since they do not add expressive power, past modalities have not been widely studied, and model-checkers such as Spin or Cadence-SMV do not handle LTL+Past specifications. However, over the last few years, past has been more and more studied: Several methods have been proposed for model-checking LTL+Past [33,16,13,3], and some of them are being implemented.

The benefits of the past. Allowing past-time modalities makes specifications easier and more natural [23]. Furthermore, there is a sense in which past really brings more expressive power: there is a succinctness gap between LTL and LTL+Past, *i.e.* there exists LTL+Past formulas that only have LTL equivalents of exponential size [21,28]. Finally, since model checking and satisfiability are not more difficult for LTL+Past (PSPACE-complete in both cases [32]), one could argue that LTL+Past should be preferred.

These arguments seem to indicate that past is for free. Can this observation be made stronger and more systematic? In this paper, we investigate if this line of argument still holds for different fragments of LTL+Past, in order to characterize fragments that are more expressive but not harder to verify.

Looking for simpler fragments. PSPACE-hardness occurs in the general case, but some fragments of LTL+Past have lower complexity (*e.g.* $\mathbf{L}^{+}(\mathbf{F}, \mathbf{X})$ [32] or $\mathbf{L}(\mathbf{F}, \mathbf{F}^{-1})$ [10]). Identifying such "simpler" fragments could lead to improved algorithms for special cases, and help understand where the precise boundary lies between hard and easy fragments. Several fragments have already been considered [32, 8], but not much is known about those with past-time modalities.

Our contribution. We provide a systematic study of fragments of LTL+Past obtained by three kinds of restrictions: on the set of allowed modalities, on the use of negations, and on nesting of past and future modalities. This includes the pure-future fragments. We sum up our results in table 1, on page 8. They rely on a few basic techniques that are used throughout the paper.

Several results are somewhat surprising. The first one is that our simple techniques were sufficient to characterize the exact complexity classes for all the problems we considered: All of them are either (co)NP-complete or PSPACE-complete. While many people would think that there is no room for model checking problems between NP and PSPACE, the gap is actually quite large and is populated by a few rare model checking problems [19,20,31].

We also prove that, in many cases, restrictions on the set of allowed modalities or on the use of negations decrease the theoretical complexity of verification problems. This means that there are many fragments for which specialized algorithms could be more efficient than classical ones.

Concerning past-time modalities, it should be remarked that if a given future modality is allowed, adding its symmetric past-time modality does not increase the complexity. Moreover, allowing or disallowing the nesting of future modalities in the scope of past modalities does not change the complexity of the verification problems. To sum up, past comes with no extra cost.

Last, we remark that, in some (positive) fragments, existential and universal problems may have different complexities. This had probably not been remarked by Sistla and Clarke when they erroneously claim (in [32]) that validity of $\mathbf{L}^+(\mathbf{F}, \mathbf{X})$ is coNP-complete, when it is in fact PSPACE-complete.

Related work. As regards fragments of LTL+Past, Ramakrishna et al. [30] studied LUSAT, the fragment of LTL+Past with only $\mathbf{U}$ and $\mathbf{S}$, and they provide an optimal (PSPACE) automata-theoretic algorithm for model checking with this fragment. Other fragments of LTL (with no past) are addressed in [8], namely fragments obtained by bounding the temporal height and the number of atomic propositions of formulas. Branching-time temporal logics with past have been investigated in [18,22].

Outline of the paper. In the sequel, we first formally define the structures, logics and problems under study, and sum up our results. We prove NP-completeness results in section 2, and PSPACE-completeness results in section 3. We summarize our study and conclude in section 4.

1 PLTL: Linear Temporal Logic with Past

Syntax of PLTL. Let $AP = \{P_1, P_2, \ldots\}$ be a countable set of atomic propositions. We define the syntax of PLTL as follows:

$$\text{PLTL} \ni \phi, \psi ::= \psi \vee \phi \mid \neg\phi \mid \mathbf{X}\,\phi \mid \psi\,\mathbf{U}\,\phi \mid \mathbf{X}^{-1}\,\phi \mid \psi\,\mathbf{S}\,\phi \mid P_1 \mid P_2 \mid \ldots$$

where $\mathbf{U}$ reads "until", $\mathbf{S}$ reads "since", $\mathbf{X}$ is "next" and $\mathbf{X}^{-1}$ is "previous".

Some very useful abbreviations are commonly defined: $\top \equiv P_1 \vee \neg P_1$, $\Rightarrow$, $\Leftrightarrow$... As regards temporal modalities, we will use the classical $\mathbf{F}$ (eventually) and $\mathbf{G}$ (always), as well as their past counterparts $\mathbf{F}^{-1}\,\phi \equiv \top\,\mathbf{S}\,\phi$ and $\mathbf{G}^{-1}\,\phi \equiv \neg\mathbf{F}^{-1}\,\neg\phi$, read "*eventually in the past*" and "*always in the past*" respectively.

Modalities $\mathbf{S}$, $\mathbf{X}^{-1}$, $\mathbf{F}^{-1}$ and $\mathbf{G}^{-1}$ are called "*past modalities*", while $\mathbf{U}$, $\mathbf{X}$, $\mathbf{F}$, $\mathbf{G}$ are "*future modalities*".

Subformulas, temporal height and size of a formula. Given a formula $\Phi \in$ PLTL, the set of its subformulas, denoted by $\mathrm{sf}(\Phi)$, is defined inductively as follows:

$$\begin{array}{rlrl}
\mathrm{sf}(P) = \{P\} & \text{for all } P \in AP & & \\
\mathrm{sf}(\neg\phi) = \{\neg\phi\} \cup \mathrm{sf}(\phi) & & \mathrm{sf}(\phi \vee \psi) = \{\phi \vee \psi\} \cup \mathrm{sf}(\phi) \cup \mathrm{sf}(\psi) & \\
\mathrm{sf}(\mathbf{X}\ \phi) = \{\mathbf{X}\ \phi\} \cup \mathrm{sf}(\phi) & & \mathrm{sf}(\phi\ \mathbf{U}\ \psi) = \{\phi\ \mathbf{U}\ \psi\} \cup \mathrm{sf}(\phi) \cup \mathrm{sf}(\psi) & \\
\mathrm{sf}(\mathbf{X}^{-1}\ \phi) = \{\mathbf{X}^{-1}\ \phi\} \cup \mathrm{sf}(\phi) & & \mathrm{sf}(\phi\ \mathbf{S}\ \psi) = \{\phi\ \mathbf{S}\ \psi\} \cup \mathrm{sf}(\phi) \cup \mathrm{sf}(\psi) &
\end{array}$$

Obviously, the number of subformulas is bounded by the size (*i.e.* the number of symbols) of the formula. The closure of the formula Φ, written $\overline{\mathrm{sf}(\Phi)}$, is the least set containing $\mathrm{sf}(\Phi)$ and stable under negation. It contains at most twice as many formulas as $\mathrm{sf}(\Phi)$.

The temporal height of a formula $\Phi \in$ PLTL, for a given set M of modalities, is defined recursively as follows:

$$\begin{aligned}
\mathrm{h}_M(P) &= 0\\
\mathrm{h}_M(\neg\phi) &= \mathrm{h}_M(\phi)\\
\mathrm{h}_M(\phi \vee \psi) &= \max(\mathrm{h}_M(\phi), \mathrm{h}_M(\psi))
\end{aligned}$$

and, for any modality O of arity p,

$$\mathrm{h}_M(O(\phi_1,\ldots,\phi_p)) = \max(\mathrm{h}_M(\phi_1),\ldots,\mathrm{h}_M(\phi_p)) + \begin{cases} 1 & \text{if } O \in M \\ 0 & \text{otherwise} \end{cases}$$

The temporal height of a formula is its temporal height for the set of all modalities. The past temporal height is the temporal height for the set of past-time modalities. The temporal height is obviously bounded by the size of the formula.

Semantics. Formulas of PLTL are interpreted over paths. A path is a pair (π,ξ) in which π is an infinite sequence of states $\pi(0)$, $\pi(1)$, ... and ξ is a mapping $\{\pi(0),\pi(1),\ldots,\pi(n),\ldots\} \to 2^{AP}$. This way, the states of π are labelled with atomic propositions.

Given a path (π,ξ), a natural i and a formula ϕ, we inductively define the relation[1] $(\pi,\xi), i \models \phi$ (read "*ϕ holds at position i along (π,ξ)*") as follows:

$\pi, i \models P$	iff $P \in \xi(\pi(i))$,
$\pi, i \models \phi \wedge \psi$	iff $\pi, i \models \phi$ and $\pi, i \models \psi$,
$\pi, i \models \neg\phi$	iff $\pi, i \not\models \phi$,
$\pi, i \models \mathbf{X}\,\phi$	iff $\pi, i+1 \models \phi$,
$\pi, i \models \psi\,\mathbf{U}\,\phi$	iff there exists some $j \geq i$ s.t. $\pi, j \models \phi$ and for all $i \leq k < j,\ \pi, k \models \psi$,
$\pi, i \models \mathbf{X}^{-1}\,\phi$	iff $i > 0$ and $\pi, i-1 \models \phi$,
$\pi, i \models \psi\,\mathbf{S}\,\phi$	iff there exists some $j \leq i$ s.t. $\pi, j \models \phi$ and for all $j < k \leq i,\ \pi, k \models \psi$.

Equivalence of formulas. Two formulas are *(globally) equivalent over a class Π of paths* (which we denote $\phi \equiv^{\Pi} \psi$) if for any path $\pi \in \Pi$ and any integer i, the equivalence $\pi, i \models \phi \Leftrightarrow \pi, i \models \psi$ holds. The formulas are *initially equivalent* over Π ($\phi \equiv_i^{\Pi} \psi$) if for all paths $\pi \in \Pi$, $\pi, 0 \models \phi \Leftrightarrow \pi, 0 \models \psi$ is true. We omit Π when the equivalence has to hold along all paths in $(2^{AP})^{\mathbb{N}}$.

Obviously, two equivalent formulas are initially equivalent. The converse does not hold. For instance, $P_1\,\mathbf{S}\,P_2$ and P_2 are initially equivalent, but they clearly are not globally equivalent.

A formula ϕ is said to be *initially* (resp. *globally*) *valid* over Π if it is initially (resp. globally) equivalent to $\top$ over Π. It is *initially* (resp. *globally*) *satisfiable* over Π if its negation is not initially (resp.

[1] In the sequel, we won't mention ξ whenever it is not ambiguous.

globally) valid over Π. This means that there exists a path $\pi \in \Pi$ (resp. a path $\pi \in \Pi$ and a position i along that path) such that $\pi, 0 \models \phi$ (resp. $\pi, i \models \phi$).

These definitions formalize the results we mentioned about expressive power in the introduction: that PLTL is as expressive as LTL [15, 12, 11] means that for any PLTL formula, there exists an initially equivalent LTL formula. The exponential succinctness gap in [21] can be expressed as follows: there exists a sequence of PLTL formulas (ϕ_n), s.t. $|\phi_n| \in O(n)$, and for which any sequence of initially equivalent LTL formulas (ψ_n) verifies that $|\psi_n| \in \Omega(2^n)$.

Verification problems. In this paper, we are concerned with the following problems:

- Initial satisfiability and validity, as defined above;
- Universal model checking, *i.e.* initial validity over a given set Π of paths;
- Existential model checking, *i.e.* initial satisfiability over a given set Π of paths.

Note that we only study "initial" problems, since these are the most interesting ones as regards verification. Moreover, in the general case, initial problems and their global counterparts are interreducible: for instance, a formula ϕ is globally satisfiable if, and only if, $\mathbf{F}\,\phi$ is initially satisfiable, and conversely, ϕ is initially satisfiable if, and only if, $\mathbf{G}^{-1}\,\mathbf{F}^{-1}\,\phi$ is globally satisfiable.

Kripke Structures. For model checking, the set Π is often defined through a Kripke Structure (KS for short), that is, a 4-tuple $K = (Q, Q_0, l, R)$ in which Q is a finite set of states, Q_0 is the set of initial states, $l \in (2^{AP})^Q$ indicates the propositions that are true in each state of Q, and $R \subseteq Q \times Q$ is a total relation representing the set of allowed transitions. The *size* of K, which we denote $|K|$, is $|Q| + |R|$. A KS generates a set Π of infinite paths in the obvious way.

It should be remarked right now that, in the general case, both model checking problems are dual: indeed, there exists a path satisfying a formula ϕ if, and only if, it is not the case that every path satisfies the negation of ϕ. But this equivalence only holds for fragments allowing negation.

Loops. A path (π, ξ) is said to be *ultimately periodic* if there exist two integers m and p, with $p > 0$, such that for any integer $n \geq m$, $\pi_n = \pi_{n+p}$. Such a path can be *finitely* represented by a *loop*, that is a *deterministic* KS, or, equivalently, by a couple of finite words

(u, v) over the alphabet 2^{AP}, where $|u| = m$ and $|v| = p$. The path correponding to a loop $L = (u, v)$, which we denote by π_L, is uv^ω. The type of a loop (u, v) is $(|u|, |v|)$. The size of a loop of type (m, p) is the integer $m + p$. The size of an ultimately periodic path π is the size of the smallest loop encoding that path[2].

Subpath, subloop. Given a path (π, ξ), a *subpath* is a path $(\pi', \xi_{|\pi'})$ where π' is a subsequence of π. We equivalently say that π' is a subpath of π, or that π contains π'. If π' is a subpath of π, there exists an increasing function f such that, for all i, $\pi'_i = \pi_{f(i)}$. We will write $\pi' \sqsubseteq_f \pi$ when we need the function f. Otherwise, we simply write $\pi' \sqsubseteq \pi$.

In the same way, given a loop L, a *subloop* is a loop L' whose associated path is a subpath of the path associated to L. We also write $L' \sqsubseteq L$ in that case. Note that a subloop could be bigger than its original loop. For instance, from the loop (ε, ab) (where ε is the empty word), whose size is 2, we can extract the loop $(aaba, aab)$, whose size is 7.

Let $L = (u, v)$ be a loop. We say that a subloop $L' = (u', v')$ of L is *acceptable*, and we write $L' \preccurlyeq L$, whenever u' is a subpath of u and v' a subpath of v. The size of an acceptable subloop is always lower than or equal to the size of the original loop.

Fragments of PLTL. We consider three types of restrictions: first of all, restrictions about the allowed modalities. For denoting the fragment of LTL where only $\mathbf{M}_1, \ldots, \mathbf{M}_p$ are allowed, we use the classical notation $\mathbf{L}(\mathbf{M}_1, \ldots, \mathbf{M}_p)$. For instance, $\mathbf{L}(\mathbf{F})$ is the logic where $\mathbf{F}$ is the only allowed temporal modality[3]. The second restriction we deal with affects negations: we write $\mathbf{L}^+(\mathbf{F}, \mathbf{X})$, for example, for the logic where the only modalities are $\mathbf{F}$ and $\mathbf{X}$, and where modalities can not occur in the scope of a negation. Last, a formula is said to be *stratified* if it has no future modality in the scope of past modalities [24]. Sets of stratified formulas are denoted by $\mathbf{L}_s(\ldots)$. We write $\mathbf{L}_s^+(\mathbf{F}, \mathbf{S})$ when combining restrictions about negation and stratification.

For example, $\mathbf{F}\,(a \wedge \mathbf{G}^{-1}\,(b \vee \mathbf{F}\,c))$ lies in $\mathbf{L}^+(\mathbf{F}, \mathbf{G}^{-1})$, but it is not stratified. It is initially equivalent to $(b \vee \mathbf{F}\,c)\ \mathbf{U}\ (a \wedge (b \vee \mathbf{F}\,c))$, which is in $\mathbf{L}^+(\mathbf{U})$. And it is globally equivalent to $\mathbf{F}\,(a \wedge (\mathbf{F}\,c \vee \mathbf{G}^{-1}\,b \vee b\ \mathbf{S}\ c))$, which belongs to $\mathbf{L}_s(\mathbf{F}, \mathbf{S})$.

---

[2] This "smallest loop" does exist, since if two loops of type (m, p) and (m', p') represent the path π, then we can build a loop of type $(\min(m, m'), \gcd(p, p'))$ encoding π.

[3] In this case, we see $\mathbf{F}$ as a modality, and not as an abbreviation for $\top\,\mathbf{U}\,\cdot$.

Our results. In the sequel, we get the following results:

	Exist. MC	Univ. MC	Satisf.	Validity
$\mathbf{L}^+(\mathbf{F})$,$\mathbf{L}^+(\mathbf{G})$,$\mathbf{L}^+(\mathbf{X})$	**NP-c.**	**coNP-c.**	**NP-c.**	**coNP-c.**
$\mathbf{L}^+(\mathbf{F},\mathbf{X})$	(NP-c.)	**PSPACE-c.**	(NP-c.)	PSPACE-c.[4]
$\mathbf{L}^+(\mathbf{G},\mathbf{X})$	PSPACE-c.	coNP-c.	**PSPACE-c.**	coNP-c.
$\mathbf{L}^+_s(\mathbf{F},\mathbf{X}^{-1})$	NP-c.	**PSPACE-c.**	NP-c.	**PSPACE-c.**
$\mathbf{L}^+_s(\mathbf{G},\mathbf{X}^{-1})$	**PSPACE-c.**	coNP-c.	NP-c.	coNP-c.
$\mathbf{L}^+(\mathbf{F},\mathbf{X},\mathbf{F}^{-1},\mathbf{X}^{-1})$	**NP-c.**	PSPACE-c.	**NP-c.**	PSPACE-c.
$\mathbf{L}^+(\mathbf{G},\mathbf{X},\mathbf{G}^{-1},\mathbf{X}^{-1})$	PSPACE-c.	**coNP-c.**	PSPACE-c.	**coNP-c.**
$\mathbf{L}^+_s(\mathbf{G},\mathbf{S},\mathbf{X}^{-1})$	PSPACE-c.	PSPACE-c.	NP-c.	PSPACE-c.
$\mathbf{L}^+(\mathbf{G},\mathbf{S},\mathbf{X}^{-1})$	PSPACE-c.	PSPACE-c.	**NP-c.**	PSPACE-c.
$\mathbf{L}(\mathbf{X},\mathbf{S},\mathbf{X}^{-1})$	**NP-c.**	**coNP-c.**	**NP-c.**	**coNP-c.**
$\mathbf{L}(\mathbf{F},\mathbf{F}^{-1})$	**NP-c.**	coNP-c.	**NP-c.** [10]	coNP-c.
$\mathbf{L}^+_s(\mathbf{F},\mathbf{S})$	**PSPACE-c.**	**PSPACE-c.**	**PSPACE-c.**	coNP-c.
$\mathbf{L}^+_s(\mathbf{G},\mathbf{S})$	**PSPACE-c.**	**PSPACE-c.**	NP-c.	**PSPACE-c.**
$\mathbf{L}^+(\mathbf{G},\mathbf{S})$	PSPACE-c.	PSPACE-c.	**NP-c.**	PSPACE-c.
$\mathbf{L}^+(\mathbf{U})$	**PSPACE-c.**	**PSPACE-c.**	**PSPACE-c.**	coNP-c.
$\mathbf{L}^+(\mathbf{U},\mathbf{S})$	PSPACE-c.	PSPACE-c.	PSPACE-c.	**coNP-c.**
PLTL	(PSPACE-c.)	(PSPACE-c.)	(PSPACE-c.)	(PSPACE-c.)

Table 1: Complexity of PLTL verification

The results in bold are proved in this paper, the ones in parentheses were proved in [32], and the other ones are corollaries, deduced by inclusion or duality. For instance, existential model checking for $\mathbf{L}^+_s(\mathbf{F},\mathbf{X}^{-1})$ is NP-complete since it is a subcase of existential model checking for $\mathbf{L}^+(\mathbf{F},\mathbf{F}^{-1},\mathbf{X},\mathbf{X}^{-1})$, and since it is more general than existential model checking for $\mathbf{L}^+(\mathbf{F})$. PSPACE-completeness of validity for $\mathbf{L}(\mathbf{F},\mathbf{X})$ comes by duality from PSPACE-completeness of satisfiability for $\mathbf{L}^+(\mathbf{G},\mathbf{X})$. It should be remarked that $\mathbf{X}$ *is* self-dual (since we consider infinite paths) but $\mathbf{X}^{-1}$ is not: for example, the argument above for validity of $\mathbf{L}(\mathbf{F},\mathbf{X})$ cannot be applied to $\mathbf{L}^+_s(\mathbf{F},\mathbf{X}^{-1})$.

2 NP-complete problems

In this section, we first prove that verifying any non-trivial fragments of PLTL is at least NP-hard. "Non trivial" means fragments allowing at least one future modality. For "trivial" fragments, model checking

[4] Contrary to a claim in [32] that it is co-NP-complete.

problems amount to evaluating a boolean formula, which is ALOGTIME-complete [4], and satisfiability and validity are the classical satisfiability and validity problems for boolean formulas, which are (co)NP-complete.

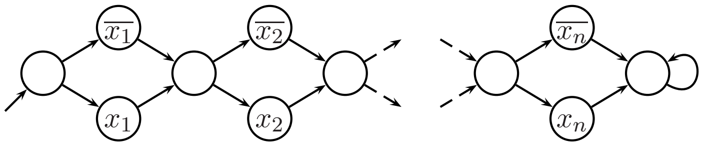
We then provide NP algorithms for several fragments of PLTL, which therefore are NP-complete.

2.1 NP*-hardness of verifying linear temporal properties*

Our first result is a small extension of a result given in [32]:

Theorem 1. *For* $\mathbf{L}^+(\mathbf{F})$, $\mathbf{L}^+(\mathbf{G})$ *and* $\mathbf{L}^+(\mathbf{X})$, *the existential model checking problem is* NP*-hard.*

Proof. – We adapt Sistla and Clarke's proof [32] of NP-hardness for model checking $\mathbf{L}(\mathbf{F})$. To a 3-SAT instance $\bigwedge_i \bigvee_j \alpha_{i,j}$, where the $\alpha_{i,j}$ are literals on $\{x_1, x_2, \ldots, x_n\}$, we associate the following KS:



To one path in that structure corresponds one valuation of the variables x_i: Variable x_i is evaluated to true if, and only if, the path runs through state x_i. Satisfiability of our 3-SAT instance is equivalent to the existence of a path verifying $\bigwedge_i \bigvee_j \mathbf{F}\,\alpha_{i,j}$ in the above KS.

- in the same way, satisfiability of $\bigwedge_i \bigvee_j \alpha_{i,j}$ is equivalent to the existence of a path satisfying $\bigwedge_i \bigvee_j \mathbf{G}\,\neg(\overline{\alpha_{i,j}})$ in the same structure.
- for $\mathbf{L}^+(\mathbf{X})$, our 3-SAT instance $\bigwedge_i \bigvee_j \alpha_{i,j}$ is satisfiable if, and only if, the above KS contains a path verifying $\bigwedge_i \bigvee_j \mathbf{X}^{\,2n(\alpha_{i,j})-1}\alpha_{i,j}$, where $n(x_k) = n(\overline{x_k}) = k$. $\square$

By duality (since, for instance, verifying that "all paths satisfy a formula $\phi \in \mathbf{L}^+(\mathbf{F})$" amounts to verifying that "there does not exist a path satisfying $\neg\phi$", with $\neg\phi \in \mathbf{L}^+(\mathbf{G})$), we get

Theorem 2. *Universal model checking for fragments* $\mathbf{L}^+(\mathbf{G})$, $\mathbf{L}^+(\mathbf{F})$ *and* $\mathbf{L}^+(\mathbf{X})$ *is co*NP*-hard.*

2.2 NP*-easy problems*

What we saw in the previous section entails that any non-trivial verification problem concerning linear temporal logics is NP-hard. We know from [32] that this lower bound is optimal for $\mathbf{L}(\mathbf{F})$ and $\mathbf{L}^+(\mathbf{F}, \mathbf{X})$, that is, the satisfiability and (existential) model checking problems for these logics are NP-complete. In this section, we prove NP-easiness of satisfiability for four other fragments: $\mathbf{L}(\mathbf{X}, \mathbf{X}^{-1}, \mathbf{S})$, $\mathbf{L}(\mathbf{F}, \mathbf{F}^{-1})$, $\mathbf{L}^+(\mathbf{F}, \mathbf{F}^{-1}, \mathbf{X}, \mathbf{X}^{-1})$, and $\mathbf{L}^+(\mathbf{G}, \mathbf{S})$. These results carry on to existential model checking, except for $\mathbf{L}^+(\mathbf{G}, \mathbf{S})$, which we will prove is PSPACE-complete.

We systematically use the small witness method in order to do this, consisting in

- providing a polynomial time algorithm for verifying that an ultimately periodic path may be checked in polynomial time against any PLTL formula;
- showing that the fragments listed above have the "polynomial witness property", *i.e.* if a formula is satisfiable, it is satisfiable along a polynomial size ultimately periodic path.

This obviously gives an NP-algorithm for satisfiability: First guess an ultimately periodic witness, and then check it. In several cases, this technique also provides a proof for NP-easiness of the model checking problems: Indeed, we show that, for some fragments, we can add arbitrary states in the polynomial witness, which ensures that we can find a polynomial witness in the KS under study. In these cases, the algorithm for existential model checking is as follows: First pick the candidate witness *in the Kripke structure*, and check that it satisfies the formula.

2.2.1 Model checking a loop. We first recall the following result:

Theorem 3. *Given a pure-future formula ϕ and a loop L, one can check in time $O(|L| \cdot |\phi|)$ whether $\pi_L, 0 \models \phi$.*

For a deterministic KS, the CTL model checking algorithm [5,6] can be applied to LTL formulas too, since path quantification will always refer to the only possible execution.

This simple approach does not extend to the problem of checking whether a loop satisfies a PLTL formula: In a loop, future is "deterministic" but past is not, since the first state of the periodic part has two predecessors. The following lemma gives a way to overcome that problem[5].

[5] Another algorithm, reducing to the problem of model checking a *finite* path against PLTL, has been proposed in [27].

Lemma 4. *Let ϕ be a* PLTL*-formula. For any loop L of type (m,p), for all $k \geq m + h_P(\phi)p$,*

$$\pi_L, k \models \phi \textit{ iff } \pi_L, k+p \models \phi.$$

This lemma indicates that, after some initial fluctuations, a state of the periodic part always satisfies the same subformulas, irrespective of how many times the periodic part has been traversed.

Proof. The proof is by induction on the structure of the formula ϕ:

- For $\phi = P$, $\phi = \neg\phi_1$ and $\phi = \phi_1 \vee \phi_2$, the result is obvious;
- If $\phi = \mathbf{X}\,\phi_1$ or $\phi = \phi_1\,\mathbf{U}\,\phi_2$, we can apply the induction hypothesis to states occurring after the k-th one;
- If $\phi = \mathbf{X}^{-1}\,\phi_1$, then since $k-1 \geq m + (h_P(\phi_1)+1)p - 1 \geq m + h_P(\phi_1)p$, by induction hypothesis, we get the equivalence $\pi_L, k-1 \models \phi_1 \Leftrightarrow \pi_L, k-1+p \models \phi_1$. Thus $\pi_L, k \models \mathbf{X}^{-1}\,\phi_1 \Leftrightarrow \pi_L, k+p \models \mathbf{X}^{-1}\,\phi_1$;
- If $\phi = \phi_1\,\mathbf{S}\,\phi_2$, we have $h_P(\phi) = \max(h_P(\phi_1), h_P(\phi_2))+1$. Suppose that $\pi_L, k \models \phi$. There exists some $k' \leq k$ s.t. $\pi_L, k' \models \phi_2$, and for $k' < l \leq k$, $\pi_L, l \models \phi_1$. Two cases may arise:
  - If $k' < k - p$, then we know that the states from π_{k-p} to π_k satisfy ϕ_1. By induction hypothesis, so do the states from π_k to π_{k+p}. Thus, $\pi_L, k+p \models \phi$, since $\pi_L, k' \models \phi_2$ and all the states between $\pi_{k'+1}$ and π_{k+p} satisfy ϕ_1;
  - Otherwise, $k' \geq k - p \geq m + h_P(\phi_1)p$, and the induction hypothesis directly applies to the states between $\pi_{k'}$ and π_k.

  Thus $\pi_L, k \models \phi \Rightarrow \pi_L, k+p \models \phi$. The reverse implication may be proved similarly. □

Corollary 5. *Model-checking* PLTL *over a loop can be done in polynomial time.*

Proof. Given a loop L of type (m,p) and a PLTL formula ψ, we use dynamic programming in order to compute which subformulas of ϕ are true in the different states of the path π_L. For this purpose, we apply the following labelling algorithm:

- First unwind the periodic part of the loop h times with h greater than $\mathrm{h}_P(\phi)$. This gives a loop of type $(m+hp, p)$.
- Then label that loop with the subformulas in $\mathrm{sf}(\phi)$ inductively, according to the following rules:
  - If ϕ is an atomic proposition P, label the state k with P if, and only if, $P \in l(k)$,
  - For boolean combinations, as well as for $\mathbf{X}$- and $\mathbf{U}$-modalities, use the classical CTL labelling algorithm,

- For a formula $\phi = \mathbf{X}^{-1}\,\phi_1$, whenever a state i is labelled with ϕ_1, then label state $i+1$ (if it exists) with ϕ.
- For a formula $\phi = \phi_1\,\mathbf{S}\,\phi_2$, if a state i is labelled with ϕ_2, then label it with ϕ. Else if state $i-1$ has been labelled with ϕ and state i has been labelled with ϕ_1, then label state i with ϕ.

This algorithm runs in time $O(|L|^2 \cdot |\phi|^2)$. It can easily been proved, using Lemma 4, that a state is labelled with the set of subformulas it satisfies.

2.2.2 Looking for ultimately periodic paths. We recall the existence of an ultimately-periodic witness for any satisfiable formula of PLTL [32].

Theorem 6. *A pure-future formula $\phi \in$ LTL is satisfiable if, and only if, it is satisfiable in a loop. A Kripke structure K "existentially" satisfies a formula ϕ if, and only if, it contains an ultimately-periodic path satisfying ϕ. These results also hold for PLTL formulas.*

Proof. For $\mathbf{L}(\,\mathbf{U}\,,\mathbf{X}\,,\,\mathbf{S}\,)$, the first statement is shown in [32]. The second one can be shown by a classical reduction from model checking to satisfiability (see [32, lemma 4.3]): Given a KS $K = (Q, Q_0, l, R)$ and a formula ϕ, we add a new atomic propositions P_q for each $q \in Q$. We define the following formulas (we will use them several times in the sequel):

$$\psi_{\text{state}} = \bigvee_{q \in Q} P_q \wedge \bigwedge_{q \in Q} (P_q \Rightarrow \neg(\bigvee_{q' \neq q} P_{q'}))$$

$$\psi_{\text{label}} = \bigwedge_{q \in Q} \big(P_q \Rightarrow (\bigwedge_{p \in l(q)} p \wedge \bigwedge_{p' \notin l(q)} \neg p)\big)$$

$$\psi_{\text{trans}} = \bigwedge_{q \in Q} (P_q \Rightarrow \bigvee_{\substack{q' \text{ s.t.} \\ (q,q') \in R}} \mathbf{X}\, P_{q'})$$

The first formula ensures that exactly one "state" proposition is true at a time, the second one means that atomic propositions labelling that state are true, and the third one means that transitions are respected.

Now, the formula

$$\hat{\phi} = \Big[(\bigvee_{q_0 \in Q_0} P_{q_0}) \wedge \mathbf{G}\ \psi_{\text{state}} \wedge \mathbf{G}\ \psi_{\text{label}} \wedge \mathbf{G}\ \psi_{\text{trans}} \Big] \wedge \phi$$

is satisfiable if, and only if, K contains a path starting from q_0 along which ϕ is true. But if $\hat{\phi}$ is satisfiable, it is satisfiable along an ultimately-periodic path. This path witnesses the existence of an ultimately-periodic path in K along which ϕ is true.

The result is extended to PLTL thanks to the following Theorem:

Theorem 7 ([15, 11]). *For any* PLTL *formula ϕ, there exists a boolean combination of pure-future and pure-past formulas which is initially equivalent to ϕ.* □

2.2.3 NP*-easy fragments.* Lemmas 8 to 12 are four technical lemmas that directly entail the polynomial witness property for several PLTL fragments. They prove NP-easiness of satisfiability or validity for these fragments. Some of them also entail NP-easiness of model checking.

Lemma 8. *The truth of an $\mathbf{L}(\mathbf{X}, \mathbf{X}^{-1}, \mathbf{S})$-formula ϕ in the initial state of a path π only depends on the first $\mathrm{h}_{\mathbf{X}}(\phi)$ states of π.*

This result is obvious.

Lemma 9. *Let $\phi \in \mathbf{L}(\mathbf{F}, \mathbf{F}^{-1})$, and L be a loop s.t. $\pi_L, i \models \phi$ for some integer i. Then there exists an acceptable subloop $L' \preccurlyeq L$, whose size is polynomial in $|\phi|$, containing π_i, and s.t. any acceptable subloop L'' s.t. $L' \preccurlyeq L'' \preccurlyeq_f L$ satisfies $\pi_{L''}, f^{-1}(i) \models \phi$.*

This lemma means that a path satisfying an $\mathbf{L}(\mathbf{F}, \mathbf{F}^{-1})$ formula has polynomialy many "important" states, the other states being removable.

Proof. We suppose that the loop L is of type (m, p), and that, for some i, $\pi_L, i \models \phi$. We write $h = h_P(\phi)$.

For each subformula of ϕ of type $\mathbf{F}^{-1}\xi$, if there exists a position where ξ is satisfied, then we know (from Lemma 4) that there is one lower than $m + hp$. In this case, we write

$$i_{\mathbf{F}^{-1}\xi} = \min\{i \mid \pi_L, i \models \xi\}$$

The same holds for subformulas of type $\mathbf{F}\xi$: if there exists a state satisfying ξ, we write

$$i_{\mathbf{F}\xi} = \max\{j \in [\![0; m + hp - 1]\!] \mid \pi_L, j \models \xi\}$$

For each $\mathbf{F}$- or $\mathbf{F}^{-1}$-subformula ψ, we define j_ψ to be either equal to i_ψ if $i_\psi \leq m$, or congruent to i_ψ modulo p and between m and $m + p - 1$ otherwise. We define L' to be the acceptable subloop of L built by keeping states j_ψ for all $\mathbf{F}$- and $\mathbf{F}^{-1}$-subformulas ψ of ϕ. We also add the current state π_i. The acceptable subloop L'' is defined from L' by possibly inserting some other states of L. We let f be the function s.t. $\pi_{L''} \preccurlyeq_f \pi_L$. Remark that

- L'' has type (m'', p'') with $m'' \leq m$ and $p'' \leq p$,

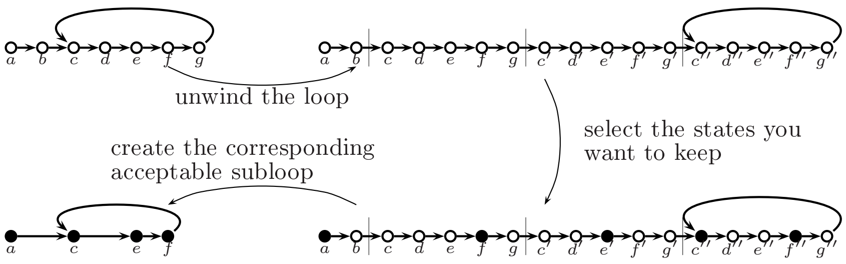


Figure 1 Construction of L'

- $f(m'') \leq m \leq f(m''+1)$,
- for all k, $f(m''+kp'') \leq m+kp \leq f(m''+kp''+1)$.

The example shown on Figure 1 explains this construction.

We now have to prove that this construction is correct. For this, we show that

$$\forall \psi \in \overline{\mathrm{sf}(\phi)}, \forall j \in \mathbb{N},\ \pi_{L'}, j \models \psi \Leftrightarrow \pi_L, f(j) \models \psi.$$

We prove this by induction on the structure of ψ:

- for atomic propositions and boolean combinations, the result is straightforward;
- if $\psi = \mathbf{F}\,\psi_1$, suppose that for some j, $\pi_{L'}, j \models \mathbf{F}\,\psi_1$. Then for some state $j' \geq j$, we have $\pi_{L'}, j' \models \psi_1$. By ind. hyp. and since f is increasing, we get a state $f(j') \geq f(j)$ s.t. $\pi_L, f(j') \models \psi_1$, and $\pi_L, f(j) \models \mathbf{F}\,\psi_1$.
  Conversely, if we suppose that $\pi_L, f(j) \models \mathbf{F}\,\psi_1$, then there exists a state $j' \geq f(j)$ s.t. $\pi_L, j' \models \psi_1$. Two cases may arise:
  - either $j' \leq m+hp-1$, then we have $j' \leq i_{\mathbf{F}\,\psi_1}$. By ind. hyp. we get that $\pi_{L'}, f^{-1}(i_{\mathbf{F}\,\psi_1}) \models \psi_1$, and since f is increasing, we have $f^{-1}(i_{\mathbf{F}\,\psi_1}) \geq j$.
  - or $j' \geq m+hp$. In that case, Lemma 4 ensures that there exists a state k satisfying ψ_1 s.t. k is between $m+(h-1)p$ and $m+hp-1$. Then $k \leq i_{\mathbf{F}\,\psi_1}$, and $\pi_{L'}, f^{-1}(i_{\mathbf{F}\,\psi_1}) \models \psi_1$. The remark above ensures that $f^{-1}(i_{\mathbf{F}\,\psi_1})$ lies between $m'+(h-1)p'$ and $m'+hp'$. Since Lemma 4 also applies to L', we get that for all $l \geq 0$, $\pi_{L'}, f^{-1}(i_{\mathbf{F}\,\psi_1})+lp' \models \psi_1$. Thus for all m, $\pi_{L'}, m \models \mathbf{F}\,\psi_1$, especially for $m=j$.
- if $\psi = \mathbf{F}^{-1}\,\psi_1$, the proof is similar. ☐

Lemma 10. *Let $\phi \in \mathbf{L}^+(\mathbf{F}, \mathbf{F}^{-1}, \mathbf{X}, \mathbf{X}^{-1})$ be a satisfiable formula, and L a loop s.t. $\pi_L, i \models \phi$ for some integer i. Then there exists* *an acceptable subloop L' of L, whose size is polynomial in $|\phi|$, containing π_i, and s.t. any loop L'' for which $L' \preccurlyeq L'' \preccurlyeq_f L$ satisfies $\pi_{L''}, f^{-1}(i) \models \phi$.*

Proof. The proof is similar to the proof of Lemma 9: we first unwind the loop $h_P(\phi)$ times. Then, by induction on the structure of the formula, we build a set S of "witness states". At each step, we prove that

$$\forall j \leq m + h_P(\phi) \cdot p, \forall \psi \in \mathrm{sf}(\phi), \quad (j, \psi) \in S \Rightarrow \pi_L, j \models \psi \quad (1)$$

- initially, S contains $\{(i, \phi)\}$. The property (1) is satisfied by hypothesis;
- while S contains pairs of the form (j, ψ) where ψ is not (a negation of) an atomic proposition, we remove (j, ψ) from S, put it in T and
  - if $\psi = \alpha \vee \beta$, then either $\pi_L, j \models \alpha$ or $\pi_L, j \models \beta$. We add (j, α) or (j, β) to S in order to keep (1) true;
  - if $\psi = \alpha \wedge \beta$, then add (j, α) and (j, β) to S;
  - if $\psi = \mathbf{X}\,\alpha$, then add $(j+1, \alpha)$ (or $(j+1-p, \alpha)$ if $j+1 > m + h_P(\phi) \cdot p$) to S;
  - if $\psi = \mathbf{X}^{-1}\,\alpha$, then add $(j-1, \alpha)$ to S. We know that $j \geq 1$ since $\pi_L, j \models \mathbf{X}^{-1}\,\alpha$. Thus (1) still holds;
  - if $\psi = \mathbf{F}\,\alpha$, we know that α is true in some state k greater than j and smaller than $m + (h(\phi)+1)p$. If k is greater than $m + h(\phi)p + 1$, then we can substract p in order to remains lower than $m + h(\phi)p + 1$ (thanks to Lemma 4). Thus we add (k, α) to S, so that (1) is still satisfied;
  - if $\psi = \mathbf{F}^{-1}\,\alpha$, the argument is the same.

This process clearly ends, since the sum of sizes of formulas in S decreases at each step. Moreover, $|S \cup T| \leq |\phi|$ at the end.

Now consider the acceptable subloop L' of L containing the states we kept in $S \cup T$. We also possibly add some other states (this construction is the same as the one shown in Figure 1). We write f for the function s.t. $\pi_{L'} \preccurlyeq_f \pi_L$. The remarks of the previous proof still apply. Then $\pi_{L'}, f^{-1}(i) \models \phi$.

Indeed, we prove that, for any $(j, \psi) \in T \cup S$, we have $\pi_{L'}, f^{-1}(j) \models \psi$: clearly, for each (j, ψ) in S, we have $\pi_{L'}, f^{-1}(j) \models \psi$ since ψ is an atomic proposition. For $(j, \psi) \in T$, several cases may arise:

- if $\psi = \alpha \vee \beta$, then either (j, α) or (j, β) is in $T \cup S$, and the result comes by ind. hyp.,
- if $\psi = \alpha \wedge \beta$, then (j, α) and (j, β) are in $T \cup S$, and the result also comes from the ind. hyp.,

- if $\psi = \mathbf{X}\,\alpha$, then $(j+1, \alpha)$ (or $(j+1-p, \alpha)$) is in $T \cup S$. In the first case, the result is immediate. In the second case, it comes from Lemma 4,
- for the other modalities $\mathbf{F}$, $\mathbf{F}^{-1}$ and $\mathbf{X}^{-1}$, the argument is the same. □

This proof can easily be adapted to $\mathbf{L}^+(\mathbf{F}, \mathbf{X}, \mathbf{F}^{-1}, \neg\mathbf{X}^{-1}\neg)$, where $\neg\mathbf{X}^{-1}\neg$ is the dual modality of $\mathbf{X}^{-1}$. Indeed, we simply have to handle that dual modality slightly differently, by differentiating the case when $j = 0$ and the other cases when building S. By duality from this remark, we get a coNP algorithm for universal model checking and validity for $\mathbf{L}^+(\mathbf{G}, \mathbf{X}, \mathbf{G}^{-1}, \mathbf{X}^{-1})$.

Lemma 11. *Let $\phi \in \mathbf{L}^+(\mathbf{G}, \mathbf{S}, \mathbf{X}^{-1})$, π a path and a a state. Then*

- *If $a^\omega, 0 \models \phi$, then for all i, $a^\omega, i \models \phi$;*
- *If $\pi, 0 \models \phi$, then $(\pi_0)^\omega, 0 \models \phi$.*

Proof. We use structural induction once again: Assume that $a^\omega, 0 \models \phi$. Then:

- If ϕ is an atomic proposition, a conjunction or a disjunction, the result is obvious;
- If $\phi = \mathbf{G}\,\psi$, then $a^\omega, 0 \models \psi$. By induction hypothesis, $a^\omega, i \models \psi$ for all i, and $a^\omega, i \models \phi$ for all i;
- If $\phi = \psi_1\,\mathbf{S}\,\psi_2$, then $a^\omega, 0 \models \psi_2$. By induction hypothesis, all position i satisfy ψ_2 along a^ω, and $a^\omega, i \models \phi$ for all i;
- If $\phi = \mathbf{X}^{-1}\,\psi$, we cannot have $a^\omega, 0 \models \phi$.

The second statement is proved in the same way: We assume that $\pi, 0 \models \phi$, and prove that $(\pi_0)^\omega, 0 \models \phi$:

- It is obvious for atomic propositions and positive boolean combinations;
- If $\phi = \mathbf{G}\,\psi$, then $\pi, 0 \models \psi$. By induction hypothesis, $(\pi_0)^\omega, 0 \models \psi$, and the first statement of the lemma ensures that for positions i, $(\pi_0)^\omega, i \models \psi$. Hence the result;
- If $\phi = \psi_1\,\mathbf{S}\,\psi_2$, the same arguments apply;
- The case when $\phi = \mathbf{X}^{-1}\,\psi$ is trivial, as previously.

Lemma 12. *Let $\phi \in \mathbf{L}^+(\mathbf{U}, \mathbf{S})$. For all path π and for all state s, we have*

- *if for some i, $s^\omega, i \not\models \phi$, then for all i, $s^\omega, i \not\models \phi$;*
- *if for some i, $\pi, i \not\models \phi$, then for all j, $(\pi_i)^\omega, j \not\models \phi$.*

Thus a non-valid $\mathbf{L}^+(\mathbf{U}, \mathbf{S})$-formula has a small counterexample, and validity for that fragment is coNP-complete.

Proof. We begin with the first statement, by induction:

- For atomic propositions, or (positive) boolean combinations of subformulas, the result is obvious;
- If $\phi = \phi_1 \mathbf{U} \phi_2$, then $s^\omega, i \not\models \phi$ entails that $s^\omega, i \not\models \phi_2$. By induction, for all j, $s^\omega, j \not\models \phi_2$, and ϕ cannot be satisfied along s^ω;
- A similar argument may be used if $\phi = \phi_1 \mathbf{S} \phi_2$.

We prove the second part of the lemma in the same way:

- It is obvious for atomic propositions, as well as for conjunctions or disjunctions;
- If $\phi = \phi_1 \mathbf{U} \phi_2$, then $\pi, i \not\models \phi$ entails that $\pi, i \not\models \phi_2$. By induction, $(\pi_i)^\omega, i \not\models \phi_2$, and the first part of the lemma ensures that $(\pi_i)^\omega, j \not\models \phi_2$ for all j. Hence $(\pi_i)^\omega, j \not\models \phi$ for all j;
- In the same way, when $\phi = \phi_1 \mathbf{S} \phi_2$, then $(\pi_i)^\omega, i \not\models \phi_2$, then $(\pi_i)^\omega, j \not\models \phi_2$ for all j, and $(\pi_i)^\omega, j \not\models \phi$ for all j.

Theorem 13. *Satisfiability and existential model checking are* NP*-complete for* $\mathbf{L}(\mathbf{X}, \mathbf{S}, \mathbf{X}^{-1})$, $\mathbf{L}(\mathbf{F}, \mathbf{F}^{-1})$, $\mathbf{L}^{+}(\mathbf{F}, \mathbf{F}^{-1}, \mathbf{X}, \mathbf{X}^{-1})$, *and for their non-trivial fragments.*

*Validity is co*NP*-complete for* $\mathbf{L}^{+}(\mathbf{U}, \mathbf{S})$ *and its non-trivial fragments.*

Satisfiability is NP*-complete for* $\mathbf{L}^{+}(\mathbf{G}, \mathbf{S}, \mathbf{X}^{-1})$ *and its non-trivial fragments.*

NP-easiness is a direct consequence of the previous results. NP-hardness was proved in Section 2.1. These results are summarized in the table on page 8.

3 PSPACE-complete problems

In this section, we prove PSPACE-hardness of verification problems for several fragments of PLTL.

The proofs are reductions from two tiling problems we now define: Let C be a finite set of colors. A *domino-type* is a 4-tuple $\langle d^{up}, d^{down}, d^{left}, d^{right}\rangle$ of colors of C. Given a set $T \subseteq C^4$ of domino-types, and two integers m and n, tiling the $m \times n$-grid amounts to finding a function $f\colon [1, m] \times [1, n] \to T$ s.t.

$$\forall (i, j) \in [1, m-1] \times [1, n],\ f(i, j)^{right} = f(i+1, j)^{left}$$
$$\forall (i, j) \in [1, m] \times [1, n-1],\ \ f(i, j)^{up} = \ f(i, j+1)^{down}$$

We consider the following tiling problem, which is a slightly modified version of [14, prob. B_2]:

> Given a set T of domino-types, a natural m (in unary), and two colors c_0 and c_1 of C, does there exist a natural n s.t. the $m \times n$-grid can be tiled, with the additional conditions that $f(1,1)^{down} = c_0$ and $f(m,n)^{up} = c_1$?

This problem is PSPACE-complete. The second problem we will use is the following:

> Given a set T of domino-types, a natural m (in unary), and two colors c_0 and c_1 of C, do all correct tiling satisfying $f(1,1)^{down} = c_0$ eventually satisfy $f(m,n)^{up} = c_1$ for some n?

This problem is also PSPACE-complete since it can encode the universality problem for a polynomial space Turing machine.

Let $(C, T = \{d_1, \dots d_p\}, m, c_0, c_1)$ be an instance of B_2. W.l.o.g., we may assume that the domino-types whose d^{up}-color is c_1 are numbered from 1 to q, and the other ones from $q+1$ to p.

We build the Kripke structure shown on Figure 2. The set of atomic propositions is $T \cup \{E\} \cup \{i = k \mid k = 1, \dots, m\}$. The initial states are all the states where the d^{down}-color is c_0 and the value of i is 1. All the transitions from a state labelled with $i = k$ to a state labelled with $i = k+1$ are enabled for $k \leq m-1$. For $i = m$, if the d^{up}-color is not c_1, then it is only possible to go to states labelled with $i = 1$, else it is only possible to go to state E.

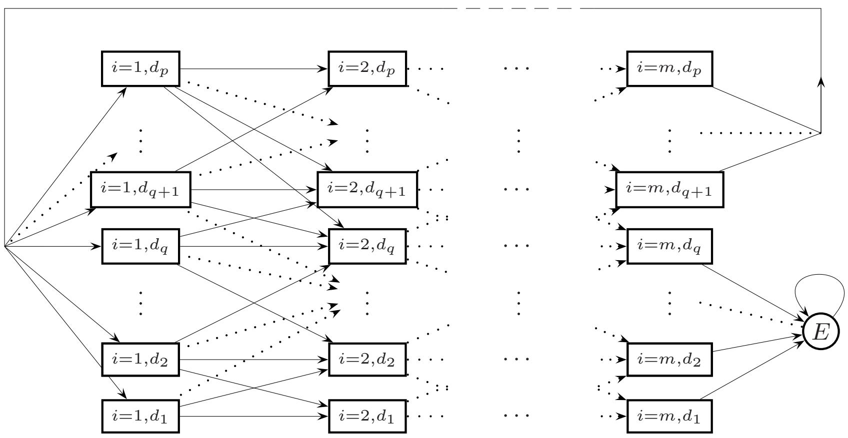


Figure 2 The Kripke structure K associated with our tiling problem.

We now have to write formulas stating that:

- Colors are respected from left to right;
- Colors are respected from top to bottom;
- The initial and final conditions are fulfilled.

3.1 The fragment $\mathbf{L}^+(\mathbf{U})$

It is well-known that model-checking and satisfiability are PSPACE-complete for $\mathbf{L}(\mathbf{U})$ [32]. The result here is a little stronger since we cannot, for instance, encode the $\mathbf{G}$ modality in $\mathbf{L}^+(\mathbf{U})$.

Theorem 14. *Existential model-checking for* $\mathbf{L}^+(\mathbf{U})$ *is* PSPACE*-hard.*

Proof. We simply have to express the three properties stated above with $\mathbf{L}^+(\mathbf{U})$ formulas:

- Both "initial" and "final" conditions are satisfied:

$$\phi_{\text{if}} \stackrel{\text{def}}{=} \top\,\mathbf{U}\,E$$

- The sequence of colors from left to right is correct:

$$\phi_{\text{horiz}} \stackrel{\text{def}}{=} \left(\bigwedge_{k=1}^{m-1} \bigwedge_{d \in T} (i = k \wedge d) \Rightarrow \right.$$
$$\left. \big(i = k\ \mathbf{U}\,(i = k + 1 \wedge \bigvee_{\substack{d' \in T \\ d'^{left} = d^{right}}} d')\big) \right) \mathbf{U}\,E$$

- The sequence is also correct from bottom to top:

$$\phi_{\text{vert}} \stackrel{\text{def}}{=} \left(\bigwedge_{k=1}^{m} \bigwedge_{d \in T} (i = k \wedge d) \Rightarrow \Big(i = k\ \mathbf{U}\,\big(\neg i = k \wedge \right.$$
$$\left. (\neg i = k)\,\mathbf{U}\,\big(E \vee (i = k \wedge \bigvee_{\substack{d' \in T \\ d'^{down} = d^{up}}} d')\big)\big)\Big) \right) \mathbf{U}\,E$$

A path in K satisfying the conjunction of those formulas eventually reaches E, after having run n times through a state where $i = 1$ holds. The path gives rise to a function $f\colon [1, m] \times [1, n] \to T$ in the obvious way. This function is a tiling function since the path satisfies the ϕ_{horiz} and ϕ_{vert} conditions. Thus the (PSPACE-complete) problem B_2 is (polynomialy) reducible to model-checking $\mathbf{L}^+(\mathbf{U})$, and model checking $\mathbf{L}^+(\mathbf{U})$ is PSPACE-hard.

Corollary 15. *Satisfiability for* $\mathbf{L}^+(\mathbf{U})$ *is* PSPACE*-hard.*

Proof. A classical method for such a proof is to reduce model checking to satisfiability. However, since we cannot use or express $\mathbf{G}$ in $\mathbf{L}^+(\mathbf{U})$, we cannot encode the behaviour of a (general) Kripke structure. Thus we will reduce our tiling problem to the satisfiability problem, by encoding the Kripke structure of Figure 2 into an $\mathbf{L}^+(\mathbf{U})$ formula. This is possible since we only have to encode its behaviour *until it reaches* E.

We assume that the KS of Figure 2 is (Q, Q_0, l, R), and we keep the notations introduced in the proof of Theorem 6, page 12.

We define

$$\phi_K \stackrel{\text{def}}{=} \Big(\bigvee_{q_0 \in Q_0} P_{q_0}\Big) \wedge \neg E \wedge (\psi_{\text{state}} \wedge \psi_{\text{label}} \wedge \psi_{\text{trans}})\,\mathbf{U}\,E$$

By construction, $\phi_K \wedge \phi_{\text{horiz}} \wedge \phi_{\text{vert}}$ is satisfiable if, and only if, the instance of the tiling problem we considered has a solution.

Theorem 16. *Universal model checking is* PSPACE*-hard for* $\mathbf{L}^+(\mathbf{U})$.

Proof. This proof requires the second tiling problem: The input is the same, but the question is whether all correct tilings having c_0 as leftmost bottom color will eventually have c_1 as rightmost top color. We will write a formula expressing that each path either does not represent a correct tiling, or eventually reaches E. Thus we write

– Left-to-right tiling condition is not satisfied at some place:

$$\phi_{\text{horiz}} \stackrel{\text{def}}{=} \bigvee_{k=1}^{m-1} \bigvee_{d \in T} \top\,\mathbf{U}\,\Big(i = k \wedge d \wedge \\ \big(i = k\ \mathbf{U}\,(i = k+1 \wedge \bigwedge_{\substack{d' \in T \\ d'^{left} = d^{right}}} \neg d')\big)\Big)$$

– Bottom-up tiling condition is not fulfilled at some place:

$$\phi_{\text{vert}} \stackrel{\text{def}}{=} \bigvee_{k=1}^{m} \bigvee_{d \in T} \top\,\mathbf{U}\,\Big(i = k\ \wedge d \wedge \Big(i = k\ \mathbf{U}\,\big(\neg i = k \wedge \\ \big(\neg i = k\ \mathbf{U}\,(i = k\ \wedge \bigwedge_{\substack{d' \in T \\ d'^{down} = d^{up}}} \neg d')\big)\big)\Big)\Big)$$

A path satisfying those properties does not correspond to a correct tiling. Thus checking that all the paths satisfy $\phi_{\text{horiz}} \vee \phi_{\text{vert}} \vee \top\,\mathbf{U}\,E$ amounts to solving our tiling problem.

3.2 PSPACE*-hardness for* $\mathbf{L}^+(\mathbf{F}, \mathbf{X})$, $\mathbf{L}^+(\mathbf{G}, \mathbf{X})$, $\mathbf{L}_s^+(\mathbf{F}, \mathbf{X}^{-1})$ *and* $\mathbf{L}_s^+(\mathbf{G}, \mathbf{X}^{-1})$

Satisfiability and existential model checking for $\mathbf{L}^+(\mathbf{F}, \mathbf{X})$ are NP-complete [32]. We show here that universal model checking is harder for that fragment.

Theorem 17. *The universal model checking problem for* $\mathbf{L}^+(\mathbf{F}, \mathbf{X})$ *is* PSPACE*-hard.*

Proof. The reduction is similar to the previous one, and formulas are even easier to write: We simply have to write the formulas expressing that a path does not correspond to a correct tiling:

– Left-to-right tiling condition is not satisfied at some place:

$$\phi_{\text{horiz}} \stackrel{\text{def}}{=} \bigvee_{k=1}^{m-1} \bigvee_{d \in T} \mathbf{F}\Big(i = k \wedge d \wedge \bigwedge_{\substack{d' \in T \\ d'^{left} = d^{right}}} \mathbf{X} \neg d'\Big)$$

– Bottom-up tiling condition is not fulfilled at some place:

$$\phi_{\text{vert}} \stackrel{\text{def}}{=} \bigvee_{k=1}^{m} \bigvee_{d \in T} \mathbf{F}\Big(i = k \wedge d \wedge \bigwedge_{\substack{d' \in T \\ d'^{down} = d^{up}}} \mathbf{X}^{n} \neg d'\Big)$$

By duality, we get

Corollary 18. *Existential model checking and satisfiability problems are* PSPACE*-hard for* $\mathbf{L}^+(\mathbf{G}, \mathbf{X})$*.*

Proof. For existential model checking, the result comes by duality from the previous Theorem. The reduction from existential model checking to satisfiability for $\mathbf{L}(\mathbf{F}, \mathbf{X})$ [32] also applies to $\mathbf{L}^+(\mathbf{G}, \mathbf{X})$.

It is easy to adapt the proof of Theorem 17 to $\mathbf{L}^+(\mathbf{F}, \mathbf{X}^{-1})$. This entails the following Theorem:

Theorem 19. *The universal model checking problem for* $\mathbf{L}^+(\mathbf{F}, \mathbf{X}^{-1})$ *is* PSPACE*-hard.*

The following result also holds, but is not exactly dual with the previous one:

Theorem 20. *The existential model checking problem is* PSPACE*-hard for* $\mathbf{L}_s^+(\mathbf{G}, \mathbf{X}^{-1})$*.*

Proof. We consider the dual problem of the one we used for the proof of Theorem 16: Given the same input, the question is whether there exists a correct tiling that never satisfies the "final" condition. For this purpose, we simply have to express that a path satisfies the tiling conditions:

– Left-to-right tiling condition is satisfied:

$$\phi_{\text{horiz}} \stackrel{\text{def}}{=} \bigwedge_{k=2}^{m} \bigwedge_{d \in T} \mathbf{G}\left(i = k \wedge d \Rightarrow \bigvee_{\substack{d' \in T \\ d'^{right} = d^{left}}} \mathbf{X}^{-1}\, d'\right)$$

– Bottom-up tiling condition is satisfied:

$$\phi_{\text{vert}} \stackrel{\text{def}}{=} \bigwedge_{k=1}^{m} \bigwedge_{d \in T} \mathbf{G}\left(i = k\ \wedge d \Rightarrow \left(\mathbf{X}^{-1\,k} \bot \vee \bigvee_{\substack{d' \in T \\ d'^{up} = d^{down}}} \mathbf{X}^{-1\,n} d'\right)\right)$$

Checking that there exists a path satisfying $\phi_{\text{horiz}} \wedge \phi_{\text{vert}} \wedge \mathbf{G}\,\neg E$ amounts to solving the initial PSPACE-complete problem.

Even though satisfiability for $\mathbf{L}_s^+(\mathbf{G}\,, \mathbf{X}^{-1}\,)$ is NP-complete, we prove here that validity for $\mathbf{L}_s^+(\mathbf{F}\,, \mathbf{X}^{-1}\,)$ is PSPACE-complete, which emphasizes the fact that $\mathbf{X}^{-1}$ is not self-dual:

Lemma 21. *Validity is* PSPACE*-hard for* $\mathbf{L}_s^+(\mathbf{F}\,, \mathbf{X}^{-1}\,)$.

Proof. We reduce the same problem we used in the proof of Theorem 16: Do all correct tilings having c_0 as leftmost bottom color eventually have c_1 as rightmost top color?

We encode a slightly modified KS in order to be able to refer to the beginning of a path: The only initial state is labelled with *Init*, it has no incoming edge and has outgoing edges to the states where $i = 1$ and the d^{down}-color is c_0. Figure 3 illustrates this construction.

The reduction is achieved as follows: We write a formula stating that

– either the path does not belong to the modified structure,
– or it does not correspond to a correct tiling,
– or it eventually reaches E.

For the sake of simplicity, and since it is not ambiguous, we assume that our new KS is defined by $(Q, \{\mathit{Init}\}, R, l)$. We keep the notations introduced in the proof of Theorem 6. That a path does not belong

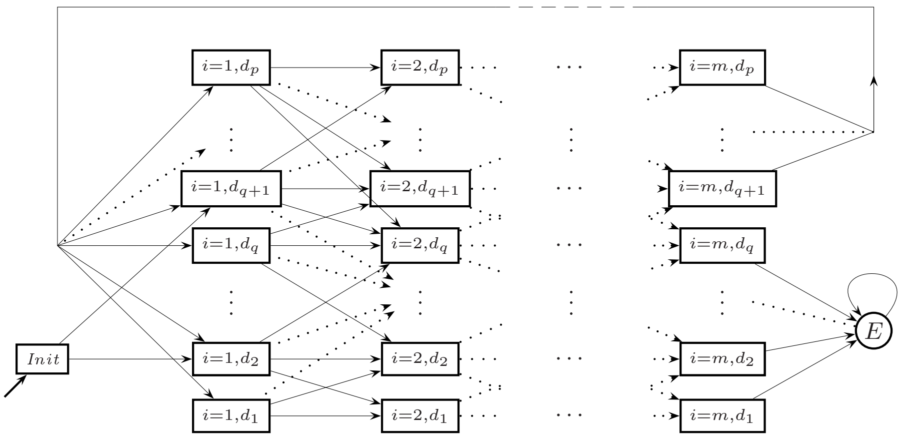


Figure 3 The modified Kripke structure (assuming that excatly d_2 and d_{q+1} have c_0 as d^{down}-color).

to the modified structure can be expressed through a disjunction of four subformulas:

$$\begin{aligned}
\phi_{\text{init}} &= \neg Init \\
\phi_{\text{state}} &= \mathbf{F}\,\neg\psi_{\text{state}} \\
\phi_{\text{label}} &= \mathbf{F}\,\neg\psi_{\text{label}} \\
\phi_{\text{trans}} &= \mathbf{F}\left[\Big(\neg Init \wedge \bigwedge_{(s,s')\in R} (\neg s' \vee \mathbf{X}^{-1}\,\neg s)\Big) \vee \Big(Init \wedge \mathbf{X}^{-1}\,\top\Big)\right]
\end{aligned}$$

A path satisfies the disjunction of these formulas iff it is not extracted from our Kripke structure.

Expressing that the path does not represent a correct tiling is done in the same way as before:

$$\phi_{\text{horiz}} = \bigvee_{d\in T} \mathbf{F}\left(\mathbf{X}^{-1}\,d \wedge \neg E \wedge \neg \bigvee_{\substack{d'\in T\\ d'^{left}=d^{right}}} d'\right)$$

$$\phi_{\text{vert}} = \bigvee_{d\in T} \mathbf{F}\left((\mathbf{X}^{-1}\,)^m d \wedge \neg E \wedge \neg \bigvee_{\substack{d'\in T\\ d'^{down}=d^{up}}} d'\right)$$

Let G be an infinite correct tiling having c_0 as leftmost bottom color, and π_G its associated path. Since the disjunction is valid, and since π_G correspond to a correct tiling, the validity of the disjunction above entails that π_G satisfies $\mathbf{F}\, E$, *i.e.* that the tiling eventually has c_1 as its rightmost top color.

Conversely, if all correct tilings having c_0 as leftmost bottom color eventually have c_1 as rightmost top color. Then, if a path belongs to the Kripke structure and correspond to a correct tiling, it will eventually reach state E.

3.3 PSPACE*-hardness for* $\mathbf{L}_s^+(\mathbf{F}, \mathbf{S})$ *and* $\mathbf{L}_s^+(\mathbf{G}, \mathbf{S})$

Theorem 22. *Existential model checking for* $\mathbf{L}_s^+(\mathbf{F}, \mathbf{S})$ *is* PSPACE-*hard.*

Proof. We still consider the structure of Figure 3. We express that

– the initial and final conditions are satisfied:

$$\phi_{\text{if}} \stackrel{\text{def}}{=} \mathbf{F}\, E$$

– horizontal sequences of dominoes form a correct tiling:

$$\phi_{\text{horiz}} \stackrel{\text{def}}{=} \mathbf{F}\, \Big(E \wedge \Big(\bigwedge_{k=2}^{m} \bigwedge_{d \in T} (i = k \wedge d) \Rightarrow \\ \big(i = k\ \mathbf{S}\, (i = k-1 \wedge \bigvee_{\substack{d' \in T \\ d'^{right} = d^{left}}} d')\big)\Big)\ \mathbf{S}\ \mathit{Init}\Big)$$

– vertical tiling conditions are fulfilled:

$$\phi_{\text{vert}} \stackrel{\text{def}}{=} \mathbf{F}\, \Big(E \wedge \Big(\bigwedge_{k=1}^{m} \bigwedge_{d \in T} (i = k \wedge d) \Rightarrow \big(i = k\, \mathbf{S}\, (\neg i = k \wedge \\ ((\neg i = k)\, \mathbf{S}\, (\mathit{Init} \vee (i = k \wedge \bigvee_{\substack{d' \in T \\ d'^{up} = d^{down}}} d'))))\Big)\ \mathbf{S}\ \mathit{Init}\Big)$$

A path satisfying all these conditions eventually reaches E, and corresponds to a correct tiling. Thus, existential model checking for $\mathbf{L}_s^+(\mathbf{F}, \mathbf{S})$ is PSPACE-complete.

Theorem 23. *Existential model checking for* $\mathbf{L}_s^+(\mathbf{G}, \mathbf{S})$ *is* PSPACE-*hard.*

Proof. This demonstration uses the same problem as in the demonstration of Theorem 20, but the reduction uses the structure of Fig. 3. Thus, we have to write two formulas stating that the sequence of states satisfies the horizontal and vertical tiling conditions:

– Left-to-right tiling condition is satisfied:

$$\phi_{\text{horiz}} \stackrel{\text{def}}{=} \bigwedge_{k=2}^{m} \bigwedge_{d \in T} \mathbf{G} \Big(i = k \wedge d \Rightarrow \bigvee_{\substack{d' \in T \\ d'^{right} = d^{left}}} (i = k \, \mathbf{S} \, (i = k-1 \wedge d') \Big)$$

– Bottom-up tiling condition is satisfied:

$$\phi_{\text{vert}} \stackrel{\text{def}}{=} \bigwedge_{k=1}^{m} \bigwedge_{d \in T} \mathbf{G} \Big(i = k \wedge d \Rightarrow \Big(i = k \;\; \mathbf{S} \;\; \Big(\neg i = k \wedge$$
$$(\neg i = k) \, \mathbf{S} \, \big(Init \vee (i = k \wedge \bigvee_{\substack{d' \in T \\ d'^{up} = d^{down}}} d')) \Big) \Big) \Big)$$

Checking that there exists a path satisfying $\phi_{\text{horiz}} \wedge \phi_{\text{vert}} \wedge \mathbf{G} \, \neg E$ amounts to solving the initial PSPACE-hard problem.

Theorem 24. *Satisfiability is* PSPACE*-hard for* $\mathbf{L}_s^+(\mathbf{F}\,,\,\mathbf{S}\,)$.

Proof. We simply have to encode the KS of Figure 3 into temporal logic formulas. We still use notations defined on page 12. We define the followong formulas:

$$\phi_{\text{init}} \stackrel{\text{def}}{=} Init \wedge \psi_{\text{state}} \wedge \psi_{\text{label}}$$
$$\phi_{\text{trans}} \stackrel{\text{def}}{=} \Big[\Big((i = 1 \;\wedge\; \neg(d^{down} = c_0)) \Rightarrow (i = 1 \;\; \mathbf{S} \;\; i = m) \Big) \wedge$$
$$\Big((i = 1 \;\wedge\; (d^{down} = c_0)) \Rightarrow (i = 1 \;\; \mathbf{S} \;\; (\phi_{\text{init}} \vee i = m)) \Big) \wedge$$
$$\Big(\bigwedge_{k=2}^{m} (i = k \Rightarrow (i = k \;\; \mathbf{S} \;\; (i = k-1))) \Big) \Big] \;\; \mathbf{S} \;\; \phi_{\text{init}}$$
$$\phi_{\text{path}} \stackrel{\text{def}}{=} \mathbf{F} \; \Big(E \wedge E \;\; \mathbf{S} \, (\neg E \;\wedge\; (\neg E \wedge \psi_{\text{state}} \wedge \psi_{\text{label}} \wedge \phi_{\text{trans}}) \;\; \mathbf{S} \;\; \phi_{\text{init}}) \Big)$$

This does not ensure that a path exactly encodes a run in the Kripke structure, since we cannot avoid going back to *Init* in the middle of the run. But the formula ensures that the part of the path between the first occurence of an E and the latest $Init$ before that E really encodes a run in the Kripke structure. The conjunction of this formula and formulas of the proof of Theorem 22 is satisfiable if, and only if, the tiling problem has a solution. Thus satisfiability is PSPACE-complete for $\mathbf{L}_s^+(\mathbf{F}\,,\,\mathbf{S}\,)$.

Theorem 25. *The universal model checking problem for* $\mathbf{L}_s^+(\mathbf{F}\,,\,\mathbf{S}\,)$ *is* PSPACE*-hard.*

Proof. The reduction is similar to the one of Theorem 16. We have to write formulas expressing that horizontal or vertical tiling conditions are not fulfilled. In $\mathbf{L}_s^+(\mathbf{F}, \mathbf{S})$, this cas be written as follows:

– Left-to-right tiling condition is not satisfied at some position:

$$\psi_{\text{horiz}(k,d)} \stackrel{\text{def}}{=} \Big((i=k \wedge d) \wedge (i=k\ \mathbf{S}\,(i=k-1 \wedge \bigwedge_{\substack{d'\in T\\ d'^{right}=d^{left}}} \neg d'))\Big)$$

$$\phi_{\text{horiz}} \stackrel{\text{def}}{=} \bigvee_{k=2}^{m} \bigvee_{d\in T} \mathbf{F}\ \psi_{\text{horiz}(k,d)}$$

– Bottom-up tiling condition is not fulfilled at some position:

$$\psi_{\text{vert}(k,d)} \stackrel{\text{def}}{=} \Big((i=k\ \wedge d) \wedge (i=k\ \mathbf{S}\,(\neg i=k\ \wedge (\neg i=k\ \mathbf{S}\ (i=k\ \wedge \bigwedge_{\substack{d'\in T\\ d'^{up}=d^{down}}} \neg d'))))\Big)$$

$$\phi_{\text{vert}} \stackrel{\text{def}}{=} \bigvee_{k=1}^{m} \bigvee_{d\in T} \mathbf{F}\ \psi_{\text{vert}(k,d)}$$

All path satisfy $\mathbf{F}\,E \vee \phi_{\text{horiz}} \vee \phi_{\text{vert}}$ if, and only if, all correct tiling eventually have c_1 as rightmost top color.

Theorem 26. *The universal model checking problem for $\mathbf{L}_s^+(\mathbf{G}, \mathbf{S})$ is* PSPACE*-hard.*

Proof. We reduce the dual problem of our initial tiling problem: Given the same input, the question is whether there exists no correct tiling meeting both initial and final conditions. This is achieved by writing that, for all path in the Kripke structure of Figure 3, if state E is eventually reached, then the path does not correspond to a correct tiling. Thus we write:

$$\mathbf{G}\,\Big(E \Rightarrow \top\,\mathbf{S}\,\Big((\bigvee_{k=2}^{m}\bigvee_{d\in T}\psi_{\text{horiz}(k,d)}) \vee (\bigvee_{k=1}^{m}\bigvee_{d\in T}\psi_{\text{vert}(k,d)})\Big)\Big)$$

Theorem 27. *Validity for $\mathbf{L}_s^+(\mathbf{G}, \mathbf{S})$ is* PSPACE*-hard.*

Proof. We keep the notations of the previous proof, and write the following formula:

$$\begin{aligned}\phi \stackrel{\text{def}}{=} \mathit{Init} \Rightarrow \mathbf{G}\Big[E \Rightarrow \\ \mathbf{F}^{-1}\big((\bigvee_{k=2}^{m}\bigvee_{d\in T}\psi_{\text{horiz}(k,d)}) \vee (\bigvee_{k=1}^{m}\bigvee_{d\in T}\psi_{\text{vert}(k,d)}) \vee \\ \neg\psi_{\text{state}} \vee \neg\psi_{\text{label}} \vee (\bigwedge_{(s,s')\in R} \neg s' \vee s'\,\mathbf{S}\,\neg s)\big)\Big]\end{aligned}$$

Clearly, if ϕ is valid, then the path corresponding to a correct tiling having c_0 as leftmost bottom color cannot reach E, *i.e.* the tiling never has c_1 as rightmost top color. Conversely, if no correct tiling ever meets both initial and final requirements, then any path not satisfying ϕ will start in state *Init*, reach state E, and correspond to a correct tiling, which is impossible.

Theorem 28. *Model-checking and satisfiability for fragments* $\mathbf{L}^+(\mathbf{U})$, $\mathbf{L}^+(\mathbf{G}, \mathbf{X})$, $\mathbf{L}^+_s(\mathbf{G}, \mathbf{X}^{-1})$, $\mathbf{L}^+_s(\mathbf{F}, \mathbf{S})$, *and for fragments of* PLTL *containing one of them, are* PSPACE*-complete. Model-checking is* PSPACE*-complete for* $\mathbf{L}^+_s(\mathbf{G}, \mathbf{S})$.

Proof. This is a direct consequence of [32, Theorem 4.1], and of Theorems in this section.

4 Concluding remarks

The results we got are sufficient to completely classify all the considered fragments of PLTL w.r.t. the complexity of (existential and universal) model-checking and satisfiability problems.

This exhaustive case study led to several surprising results. We showed that existential and universal model checking might have the different complexity for positive fragments (NP vs. PSPACE). We found only one case where existential model checking and satisfiability have different theoretical complexity. On the other hand, we observe that using the symmetric past-time modalities of the allowed future modalities does not increase the complexity of verification problems. The same remark holds for the use of future modalities in the scope of past-time modalities. This all boils down to the conclusion that past is really cheap.

After this study on the effect of adding past into *fragments* of LTL, it would be interesting to look into when "past is for free" for *extensions* of that logic, such as CTL* (as far as we know, the complexity of model checking for CTL* with linear past is still open [18]) or timed temporal logics ([2] proves that, for the validity problem over timed state sequences, past can be added for free in the Metric Temporal Logic from [17], but not in the Timed Propositional Temporal Logic of [1]).

References

1. R. Alur and T. A. Henzinger. A really temporal logic. In *Proc. 30th IEEE Symp. Foundations of Computer Science (FOCS'89), Research Triangle Park, NC, USA, Oct. 1989*, pages 164–169, 1989.
2. R. Alur and T. A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.
3. Marco Benedetti and Alessandro Cimati. Bounded model checking for past LTL. In *Proc. 9th Int. Conf. Tools and Algorithms for Construction and Analysis of Systems (TACAS'2003) Warsaw, Poland, Apr. 2003*, volume 2619 of *Lecture Notes in Computer Science*, pages 18–33. Springer, 2003.
4. Samuel R. Buss. Algorithms for boolean formula evaluation and for tree contraction. In Peter Clote and Jan Krajícek, editors, *Arithmetic, Proof Theory and Computational Complexity*, pages 95–115. Oxford University Press, 1993.
5. E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. Logics of Programs Workshop, Yorktown Heights, New York, May 1981*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.
6. E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.
7. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking.* MIT Press, 1999.
8. S. Demri and Ph. Schnoebelen. The complexity of propositional linear temporal logics in simple cases. *Information and Computation*, 174(1):84–103, 2002.
9. E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, vol. B*, chapter 16, pages 995–1072. Elsevier Science, 1990.
10. K. Etessami, M. Y. Vardi, and T. Wilke. First order logic with two variables and unary temporal logic. In *Proc. 12th IEEE Symp. Logic in Computer Science (LICS'97), Warsaw, Poland, June–July 1997*, pages 228–235. IEEE Comp. Soc. Press, 1997.
11. D. M. Gabbay. The declarative past and imperative future: Executable temporal logic for interactive systems. In *Proc. Workshop Temporal Logic in Specification, Altrincham, UK, Apr. 1987*, volume 398 of *Lecture Notes in Computer Science*, pages 409–448. Springer, 1989.
12. D. M. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proc. 7th ACM Symp. Principles of Programming Languages (POPL'80), Las Vegas, NV, USA, Jan. 1980*, pages 163–173, 1980.
13. Paul Gastin and Denis Oddoux. LTL with past and two-way very-weak alternating automata. Technical report, LIAFA, Univ. Paris 7, Paris, France, 2003.

14. D. Harel. Recurring dominos: Making the highly undecidable highly understandable. *Annals of Discrete Mathematics*, 24:51–72, 1985.
15. J. A. W. Kamp. *Tense Logic and the Theory of Linear Order.* PhD thesis, UCLA, Los Angeles, CA, USA, 1968.
16. Y. Kesten, Z. Manna, H. McGuire, and A. Pnueli. A decision algorithm for full propositional temporal logic. In *Proc. 5th Int. Conf. Computer Aided Verification (CAV'93), Elounda, Greece, June 1993*, volume 697 of *Lecture Notes in Computer Science*, pages 97–109. Springer, 1993.
17. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.
18. O. Kupferman and A. Pnueli. Once and for all. In *Proc. 10th IEEE Symp. Logic in Computer Science (LICS'95), San Diego, CA, USA, June 1995*, pages 25–35. IEEE Comp. Soc. Press, 1995.
19. F. Laroussinie, N. Markey, and Ph. Schnoebelen. Model checking CTL^+ and FCTL is hard. In *Proc. 4th Int. Conf. Foundations of Software Science and Computation Structures (FOSSACS'2001), Genoa, Italy, Apr. 2001*, volume 2030 of *Lecture Notes in Computer Science*, pages 318–331. Springer, 2001.
20. F. Laroussinie, N. Markey, and Ph. Schnoebelen. On model checking durational Kripke structures (extended abstract). In *Proc. 5th Int. Conf. Foundations of Software Science and Computation Structures (FOSSACS'2002), Grenoble, France, Apr. 2002*, volume 2303 of *Lecture Notes in Computer Science*, pages 264–279. Springer, 2002.
21. F. Laroussinie, N. Markey, and Ph. Schnoebelen. Temporal logic with forgettable past. In *Proc. 17th IEEE Symp. Logic in Computer Science (LICS'2002), Copenhagen, Denmark, July 2002*, pages 383–392. IEEE Comp. Soc. Press, 2002.
22. F. Laroussinie and Ph. Schnoebelen. A hierarchy of temporal logics with past. *Theoretical Computer Science*, 148(2):303–324, 1995.
23. O. Lichtenstein, A. Pnueli, and L. D. Zuck. The glory of the past. In *Proc. Logics of Programs Workshop, Brooklyn, NY, USA, June 1985*, volume 193 of *Lecture Notes in Computer Science*, pages 196–218. Springer, 1985.
24. Z. Manna and A. Pnueli. Completing the temporal picture. *Theoretical Computer Science*, 83:97–130, 1991.
25. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer, 1992.
26. Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety.* Springer, 1995.
27. N. Markey and Ph. Schnoebelen. Model checking a path (preliminary report). In R. Amadio and D. Lugiez, editors, *Proceedings of the 14th International Conference on Concurrency Theory (CONCUR 2003)*, volume 2761 of *Lecture Notes in Computer Science*, pages 251–265. Springer, 2003.
28. Nicolas Markey. Temporal logic with past is exponentially more succinct. *EATCS Bull.*, 79:122–128, 2003.
29. A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symp. Foundations of Computer Science (FOCS'77), Providence, RI, USA, Oct.-Nov. 1977*, pages 46–57, 1977.
30. Y. S. Ramakrishna, L. E. Moser, L. K. Dillon, P. M. Melliar-Smith, and G. Kutty. An automata-theoretic decision procedure for propositional temporal logic with Since and Until. *Fundamenta Informaticae*, 17(3):271–282, 1992.

31. Philippe Schnoebelen. Oracle circuits for branching-time model checking. In *Proc. 30th Int. Coll. Automata, Languages, and Programming (ICALP'2003), Eindhoven, NL, July 2003*, volume 2719 of *Lecture Notes in Computer Science*. Springer, 2001.
32. A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.
33. M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.