Journal of Artificial Intelligence Research 63 (2018) 557-623 Submitted 02/18; published 11/18

LTL on Finite and Process Traces: Complexity Results and a Practical Reasoner

Valeria Fionda FIONDA@MAT.UNICAL.IT
Gianluigi Greco GRECO@MAT.UNICAL.IT
DeMaCS, University of Calabria, Italy

Abstract

Linear temporal logic (LTL) is a modal logic where formulas are built over temporal operators relating events happening in different time instants. According to the standard semantics, LTL formulas are interpreted on traces spanning over an infinite timeline. However, applications related to the specification and verification of business processes have recently pointed out the need for defining and reasoning about a variant of LTL, which we name LTL_p, whose semantics is defined over *process traces*, that is, over finite traces such that, at each time instant, precisely one propositional variable (standing for the execution of some given activity) evaluates true.

The paper investigates the theoretical underpinnings of LTL_p and of a related logic formalism, named LTL_f, which had already attracted attention in the literature and where formulas have the same syntax as in LTL_p and are evaluated over *finite traces*, but without any constraint on the number of variables simultaneously evaluating true. The two formalisms are comparatively analyzed, by pointing out similarities and differences. In addition, a thorough complexity analysis has been conducted for reasoning problems about LTL_p and LTL_f, by considering arbitrary formulas as well as classes of formulas defined in terms of restrictions on the temporal operators that are allowed. Finally, based on the theoretical findings of the paper, a practical reasoner specifically tailored for LTL_p and LTL_f has been developed by leveraging state-of-the-art SAT solvers. The behavior of the reasoner has been experimentally compared with other systems available in the literature.

1. Introduction

Linear temporal logic, denoted by LTL, is a modal logic introduced in the seventies as a formal tool for verifying the correctness of computer programs and reactive systems (Pnueli, 1977, 1981). Since then, it found applications in several fields of artificial intelligence and computer science, including planning (Bacchus & Kabanza, 1998; Calvanese, De Giacomo, & Vardi, 2002; De Giacomo & Vardi, 1999; Patrizi, Lipovetzky, De Giacomo, & Geffner, 2011; Bacchus & Kabanza, 2000; Baier & McIlraith, 2006; Bienvenu, Fritz, & McIlraith, 2006, 2011; Sohrabi, Baier, & McIlraith, 2011), robotics and control theory (Bobadilla, Sanchez, Czarnowski, Gossman, & LaValle, 2011; Fagin, Halpern, Moses, & Vardi, 1995; Ding, Smith, Belta, & Rus, 2014; Kloetzer & Belta, 2008; Kwon & Agha, 2008), and business process management (Maggi, Montali, Westergaard, & van der Aalst, 2011a; Maggi, Westergaard, Montali, & van der Aalst, 2011c). In addition, there are other, closely related linear logics—see the work of Rozier (2011) for a survey—and several extensions of LTL have been proposed and studied to enhance its basic expressiveness, among which Metric Temporal Logic (Alur & Henzinger, 1990) is a noticeable example where modalities can be annotated with real intervals representing timing constraints.

LTL formulas are built on top of propositional logic by allowing the use of temporal operators, such as *next* (X), *always* (G), *eventually* (F), *until* (U), and *release* (R), in order to relate events

©2018 AI Access Foundation. All rights reserved.

happening in different time instants. In particular, formulas are interpreted over *infinite traces*,[1] that is, sequences of time instants each one describing the events, modeled as propositional variables, occurring there.

Example 1 Assume that $\{a, b, c\}$ is a set of propositional variables, and consider the LTL formula $\varphi = \mathsf{F}(c) \wedge \mathsf{G}(\neg a \vee \neg b) \wedge \mathsf{G}(\neg a \vee \mathsf{X}(b)) \wedge \mathsf{G}(\neg b \vee \mathsf{X}(a))$. The formula prescribes the existence of some time instant where c holds (because of the conjunct $\mathsf{F}(c)$). In addition, at each time instant, it cannot happen that a and b are both true (because of $\mathsf{G}(\neg a \vee \neg b)$). And, finally, whenever a (respectively, b) holds at some given time instant, then b (respectively, a) actually holds at the subsequent time instant (because of $\mathsf{G}(\neg a \vee \mathsf{X}(b)) \wedge \mathsf{G}(\neg b \vee \mathsf{X}(a))$). Consider then the trace $\pi = \pi_0, \pi_1, \pi_2, ...$, where $\pi_i = \{a, c\}$ (respectively, $\pi_i = \{b, c\}$) is the state associated with each odd (respectively, even) time instant $i \geq 0$. It can be checked that π satisfies the requirements prescribed by φ, and we say that π is a *model* of φ. Another model of φ is the trace $\pi' = \{c\}, \{c\}, \{c\}, ...$, consisting of the state $\{c\}$ being repeated infinitely often. $\lhd$

Despite the wide spectrum of applicability of LTL, there are certain applications where its semantics interpreted over infinite traces is not appropriate. For instance, in the context of reasoning problems related to the specification and verification of business processes, in particular when dealing with *declarative constraint-based workflow management systems* (Pesic, Bosnacki, & van der Aalst, 2010; Pesic, Schonenberg, & van der Aalst, 2007; van derAalst, Pesic, & Schonenberg, 2009), it has been observed that a more natural choice is to focus on interpretations over *finite* traces. Indeed, a temporal logic having the same syntax as LTL and equipped with this semantics, denoted by LTL$_f$, has been recently proposed and studied in the literature (De Giacomo & Vardi, 2013). Furthermore, a reasoner for LTL$_f$ formulas, which takes advantage of the finite-trace semantics by simplifying classical decision procedures for LTL, is available (Li, Zhang, Pu, Vardi, & He, 2014a).

Example 2 Consider the LTL$_f$ formula φ' whose syntax is the same as the LTL formula φ discussed in Example 1, that is, $\varphi' = \mathsf{F}(c) \wedge \mathsf{G}(\neg a \vee \neg b) \wedge \mathsf{G}(\neg a \vee \mathsf{X}(b)) \wedge \mathsf{G}(\neg b \vee \mathsf{X}(a))$.

Being an LTL$_f$ formula, φ' is interpreted over finite traces. In particular, any finite trace obtained by truncating the sequence $\pi' = \{c\}, \{c\}, \{c\}, ...$ at some time instant is a finite model of φ'. By contrast, consider the trace $\pi = \pi_0, \pi_1, \pi_2, ...$, where $\pi_i = \{a, c\}$ (respectively, $\pi_i = \{b, c\}$) is the state associated with each odd (respectively, even) time instant $i \geq 0$. Then, no finite prefix of π satisfies the formula φ', due to the LTL$_f$ subformula $\psi' = \mathsf{G}(\neg a \vee \mathsf{X}(b)) \wedge \mathsf{G}(\neg b \vee \mathsf{X}(a))$, which always calls for a subsequent time instant. In fact, note that the LTL$_f$ formula ψ' is unsatisfiable, while the LTL formula having the same syntax as ψ' is satisfiable (for instance, by the trace π).

Eventually, to gain some more insight about the different behavior of the two formalisms, note that while negation-free formulas are always satisfiable in LTL, this is not the case for LTL$_f$. An example of this kind is given by $a \wedge \mathsf{G}(\mathsf{X}(a))$: Interpreted over finite traces (hence, as an LTL$_f$ formula) $a \wedge \mathsf{G}(\mathsf{X}(a))$ is not satisfiable, while interpreted over infinite traces (hence, as an LTL formula) it is satisfiable by the trace where the state $\{a\}$ is repeated infinitely often. $\lhd$

Actually, in applications related to the specification and verification of business processes, linear temporal logic is very often interpreted over traces that are not only finite, but for which exactly one

1. The term "computation" is frequently adopted in classical studies about LTL instead of the term "trace" we shall use hereinafter. Indeed, "computation" is less standard than "trace" in the specific literature related to this paper—in particular, when the logic formalism is used in the context of specification and verification of business processes.

propositional variable holds at each time instant (standing for the execution of some given activity). Traces of this kind are hereinafter called *process traces*. As an example, the use of process traces is rather popular in the context of *process mining* applications (IEEE Task Force on Process Mining, 2011), whose goal is to automatically derive a process schema that can explain all the episodes recorded in an event log related to the execution of the activities of an underlying process. Indeed, process mining algorithms often deal with an abstract view of a log, by just focusing on the order of the execution of the various activities registered by a transactional information system.

Example 3 The finite trace $\pi'' = \pi''_0$ defined as a sequence consisting of one state only such that $\pi''_0 = \{c\}$ is a process trace, because the cardinality $|\pi''_0|$ of the state π''_0 is such that $|\pi''_0| = 1$.

Note that π'' is a finite model of the formula φ' in Example 2. However, there are LTL$_f$ formulas that can be satisfied by some finite trace, but for which no process trace is a model. This is, for instance, the case for the formula $a \wedge b$. $\lhd$

Hereinafter, we denote by LTL$_p$ the temporal logic whose syntax is precisely the same as LTL$_f$, and which is however interpreted over process traces, that is, over finite traces under the additional constraint that exactly one propositional variable holds at each time instant. Rather surprisingly, despite the above mentioned applications (which will be further discussed in Section 2.2) and differently from LTL$_f$, the logic LTL$_p$ has not been systematically studied in the literature, so far.

1.1 Complexity of LTL, LTL$_f$, and LTL$_p$ Fragments

The study of the complexity of LTL has been a central topic of research since the seminal paper by Sistla and Clarke (1985). In particular, they showed that the *satisfiability* problem for LTL formulas, i.e., deciding whether they admit some model, is PSPACE-complete in general, while more favorable complexity results (NP-completeness results) can be obtained by avoiding the interplay of certain temporal operators.

Moving from these foundational results, the complexity of LTL fragments has been studied by many researchers. In particular, further NP-complete fragments defined by restricting the set of temporal operators that are allowed have been singled out by Schobbens and Raskin (1999) and by Ono and Nakamura (1980). Markey (2004) analyzed the complexity as a function of the use of negation, of the temporal operators, and of their nesting, by identifying NP- or PSPACE-complete fragments. Chen and Lin (1993) showed that the complexity of deciding satisfiability remains unchanged if the input is restricted to propositional temporal Horn formulas. Demri and Schnoebelen (2002) studied the complexity of fragments built according to three parameters (the temporal operators allowed, the number of nested temporal operators, and the number of propositional variables) by obtaining tractable, NP-complete, and PSPACE-complete fragments. Dixon, Fisher, and Konev (2007) identified tractable classes of LTL formulas by combining XOR fragments. Artale, Kontchakov, Ryzhikov, and Zakharyaschev (2013) defined a number of fragments of LTL in terms of the available temporal operators and the structure of the clausal normal form, by determining when deciding satisfiability is tractable or not. Hemaspaandra (2001) proved that modal satisfiability becomes tractable if only conjunction, atomic negation, F, and G are considered. In addition, restrictions over the Boolean connectives derived from the Post's lattice of closed sets of Boolean functions (Post, 1941) combined with restrictions on the temporal operators have been studied, too (Bauland, Schneider, Schnoor, Schnoor, & Vollmer, 2009). Further related results have been derived by Halpern (1995) and Bauland, Hemaspaandra, Schnoor, and Schnoor (2006). A thorough analysis for LTL combined with spatial logics has been conducted by Gabelaia, Kontchakov, Kurucz, Wolter, and Zakharyaschev (2005), discussing in particular the effect on the complexity and on the expressiveness of the resulting formalism, by varying the temporal operators that are allowed.

Finally, note that in this classical setting where traces are infinite, a logic that is a combination of the standard linear time temporal logic with cardinality constraints restricting the numbers of literals that can be satisfied at any time instant has been studied by Dixon, Konev, Fisher, and Nietiadi (2013). From the results derived by these authors, it follows that LTL satisfiability is feasible in polynomial time when precisely one variable is required to hold at each time instant and when LTL formulas are given in the so-called *separated normal form* (Fisher, 1991). In fact, the algorithm proposed by Dixon et al. (2013) exhibits an exponential behavior with respect to the number of unconstrained variables, and translating an arbitrary LTL formula into an equivalent one in separated normal form requires the introduction of linearly-many fresh, unconstrained variables.

While several deep and useful results have been derived for LTL, less effort has been spent to analyze the complexity of LTL$_f$. For this logic, satisfiability is known to remain PSPACE-complete (De Giacomo & Vardi, 2013) and, of course, the basic NP-hard "lower bound" (deriving from the NP-hardness of deciding the satisfiability of a propositional formula, i.e., without temporal operators) still holds. However, no systematic study has been conducted so far over fragments defined with respect to the temporal operators occurring in the formulas. Furthermore, no analysis has been proposed in the literature to assess whether islands of tractability can be defined by possibly restricting the Boolean connectives that are allowed.

Similarly, no study of the computational complexity of LTL$_p$ has been conducted in the literature, so far. In fact, while for formulas in separated normal form a polynomial-time algorithm for satisfiability could be defined by adapting the approach by Dixon et al. (2013) to finite traces, the picture remains unclear for general LTL$_p$ formulas; indeed, as pointed out above, the satisfiability algorithm of Dixon et al. exhibits an exponential dependence with respect to the linearly-many auxiliary variables that are needed, in general, for transforming an arbitrary formula into one in separated normal form. Moreover, in addition to the lack of knowledge for specific syntactic fragments, results in the literature do not even clarify whether satisfiability remains PSPACE-complete for LTL$_p$ without syntactic restrictions. In particular, the computational properties of LTL$_p$ are different from those of LTL$_f$ and it is not always possible to generalize to LTL$_p$ known computational results about LTL$_f$. As an example, it is easy to observe that, in absence of temporal operators, satisfiability for LTL$_p$ is no longer NP-hard; indeed, we are still given a propositional formula, but now the goal is to check the existence of a model where precisely one variable evaluates true—which is clearly feasible in polynomial time, by testing all the possible candidate models.

1.2 Contributions

The main goal of this paper is to study the logics LTL$_p$ and LTL$_f$, by characterizing their distinguishing features and their relationships. Towards a fine-grained analysis, the study is conducted in a way that is parametric with respect to the temporal operators that are allowed.

In more detail, for any given set T of temporal operators, we define $\langle T\rangle$-LTL$_p$ (respectively, $\langle T\rangle$-LTL$_f$) as the class of all LTL$_p$ (respectively, LTL$_f$) formulas where only operators in T can be used and where negation is allowed in atomic form only.[2] Then, we analyze all such classes built

2. The requirement is meant to avoid that operators outside T can be trivially simulated by syntactic manipulations—this requirement frequently occurs when the focus is on analyzing syntactic fragments (Sistla & Clarke, 1985).

over the classical LTL operators X, G, F, U, and R, plus the *weak next* $\mathsf{X_w}$ operator, which is specific for LTL over finite traces (De Giacomo & Vardi, 2013), and we provide the following contributions:

- ▶ We identify those classes of formulas for which satisfiability is always witnessed by models whose length is linear in the size of the formula—we say that these classes enjoy the *linear-length model property*. In particular, we show that for each of the classes we consider, either the linear-length model property holds, or satisfiable formulas can be exhibited for which no model exists whose length is polynomially bounded. From the technical viewpoint, in order to establish some of these results on LTL$_p$, we first exhibit a general method that, over certain fragments, can be used to reformulate LTL$_f$ formulas into "equivalent" (more formally, satisfiability preserving) LTL$_p$ ones. Then, we provide the corresponding results over LTL$_f$.
- ▶ We perform a systematic study of the complexity of $\langle T\rangle$-LTL$_p$ and $\langle T\rangle$-LTL$_f$. For each possible fragment, satisfiability emerged to be either PSPACE-complete, or NP-complete, or feasible in polynomial time. Technical efforts have been spent, in particular, to derive the results related to those fragments for which the reformulation mentioned above cannot be applied. In order to single out further islands of tractability, the picture is completed by studying fragments defined by also considering restrictions on the Boolean connectives that are allowed. Notably, all tractability results are established via constructive polynomial-time algorithms that either compute a model or check that the given formula is not satisfiable.

In addition to such theoretical results, we provide the following more practical contributions:

- ▶ We present a reasoner, called LTL2SAT, for LTL$_p$ and LTL$_f$ formulas.[3] For the classes on which satisfiability is feasible in polynomial time, LTL2SAT implements ad-hoc efficient solution approaches. For the other classes, LTL2SAT encodes the temporal formula into a propositional one and uses a state-of-the-art *SAT solver* to compute a model. The encoding is parametric with respect to the length of the models, which we derive from our theoretical analysis. In particular, for the classes of formulas that enjoy the linear-length model property, we can guarantee that the size of the resulting propositional formula is polynomially bounded. Hence, the approach implemented by LTL2SAT can be viewed as combining ideas taken from bounded LTL model checking methods (e.g., Biere, Heljanko, Junttila, Latvala, & Schuppan, 2006; Cimatti, Clarke, Giunchiglia, Giunchiglia, Pistore, Roveri, Sebastiani, & Tacchella, 2002) with some early termination strategies and with methods that are specific to deal with process traces.
- ▶ We discuss results of an experimental evaluation we conducted over LTL2SAT and of its comparison with *Aalta* (Li et al., 2014a), which was the only reasoner so far available for LTL$_f$ formulas and which has been shown to outperform existing LTL reasoners adapted to deal with finite traces (Edelkamp, 2006; Gerevini, Haslum, Long, Saetti, & Dimopoulos, 2009; Pesic & van der Aalst, 2006a), and with NuSMV 2.6 (Cimatti et al., 2002), a state-of-the-art bounded model checker for LTL. Experiments are conducted over standard benchmarks available in the literature. Notably, many of the syntactic fragments studied in the paper (isolating NP or even tractable classes of formulas) frequently occur in them, thereby confirming the practical relevance of our theoretical complexity classification.

3. The reasoner is available at `http://ltl2sat.wordpress.com/`

We point out that some of the results discussed above appeared, in a preliminary form, in the proceedings of the AAAI'16 conference (Fionda & Greco, 2016). In particular, it is worthwhile noticing that Fionda and Greco only discussed proof sketches for some results over X, G, and F and illustrated some exploratory experimental evaluation. The analysis of the operators $\mathsf{X_w}$, U, and R (and of their interplay with X, F, and G) are entirely novel contributions of this paper.

Organization. The rest of the paper is organized as follows. The syntax and the semantics of LTL$_p$ and LTL$_f$ are formalized in Section 2, while in Section 3 some relationships existing between LTL$_p$ and LTL$_f$ are disclosed. The linear-length model property is discussed in Section 4. Complexity results are illustrated in Section 5, and islands of tractability for the satisfiability problem are singled out in Section 6. Implementation issues and experimental results are discussed in Section 7. A few final remarks and avenues for further research are illustrated in Section 8.

2. LTL on Finite and Process Traces

In this section, we present the syntax and the semantics of LTL$_f$ and LTL$_p$, by also discussing application scenarios where focusing on process traces is more appropriate than just focusing on general finite traces. The section eventually presents some basic properties of LTL$_p$ (and LTL$_f$), which will be used in the rest of the paper.

2.1 Preliminaries on LTL on Finite Traces

Syntax. Assume that a universe $\mathcal{V}$ of propositional variables is given. An LTL$_f$ formula φ is built over the propositional variables in $\mathcal{V}$, by using the Boolean connectives "$\wedge$", "$\vee$", and "$\neg$", plus a number of temporal operators. In the paper, we focus on the temporal operators "X" (next), "$\mathsf{X_w}$" (weak next), "G" (always), "F" (eventually), "U" (until), and "R" (release). We assume that negation in φ is *atomic* in the sense that it is directly applied to propositional variables only. More formally, φ is built according to the following grammar:

$$\varphi ::= x \mid \neg x \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid \mathsf{X}(\varphi) \mid \mathsf{X_w}(\varphi) \mid \mathsf{G}(\varphi) \mid \mathsf{F}(\varphi) \mid (\varphi\, \mathsf{U}\, \varphi) \mid (\varphi\, \mathsf{R}\, \varphi),$$

where x is any variable in $\mathcal{V}$. The formula φ is said to be *negation-free* if no negation occurs in it. More generally, note that we are dealing with formulas φ given in the so-called *negation normal form*. In fact, all results derived in the paper apply to arbitrary formulas, provided they are rewritten beforehand in negation normal form. Later in the section, we shall observe that this rewriting can be always performed by pushing negation to propositional variables. Similarly, observe that the above grammar does not include the classical implication and equivalence Boolean connectives, which can be simulated as usual via "$\wedge$", "$\vee$", and "$\neg$" (first using negation that is not atomic and then pushing such negation to propositional variables).

The temporal height[4] of a formula φ with respect to any operator $\mathsf{O} \in \{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}, \mathsf{R}\}$, denoted by $th(\varphi, \mathsf{O})$, is the maximum number of nested operators O in φ. The temporal size with respect to O, denoted by $ts(\varphi, \mathsf{O})$, is the overall number of occurrences of O in φ, The size of φ, denoted by $||\varphi||$, is the total number of symbols (temporal operators, Boolean connectives and propositional variables) it contains. The set of all the variables occurring in φ is denoted by $\mathcal{V}_\varphi$.

4. The notion of *temporal height* we use in the paper does not coincide with the (standard) notion of *height* of a formula that deals with the nesting of the operators independently of their types. In fact, the results we shall establish in terms of the temporal height still holds—but would be weaker if reformulated—over that more standard notion.

Example 4 For the LTL$_f$ formula $\varphi = ((a \wedge \neg b) \wedge (\mathsf{F}((c \wedge \mathsf{G}(a))) \wedge \mathsf{X}(b)))$, we have: $\mathcal{V}_\varphi = \{a, b, c\}$; $th(\varphi, \mathsf{X}) = th(\varphi, \mathsf{F}) = th(\varphi, \mathsf{G}) = 1$; $ts(\varphi, \mathsf{X}) = ts(\varphi, \mathsf{F}) = ts(\varphi, \mathsf{G}) = 1$; and $||\varphi|| = 13$. $\lhd$

Throughout the paper, we consider classes of LTL$_f$ formulas defined by imposing syntactic restrictions on the allowed operators. Formally, for any set $T \subseteq \{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}, \mathsf{R}\}$, we define $\langle T \rangle$-LTL$_f$ as the class of all LTL$_f$ formulas where only temporal operators in T can be used.

Semantics. A *finite trace* over variables in $\mathcal{V}$ is a sequence $\pi = \pi_0, \pi_1, ..., \pi_{n-1}$ associating to each $i \in \{0, ..., n-1\}$ a *state* $\pi_i \subseteq \mathcal{V}$, consisting of the set of all propositional variables that are assumed to hold at the *instant* i. The number of instants over which π is defined is its *length*, and is denoted by $len(\pi)$.

Given a finite trace π, we define when an LTL$_f$ formula φ is true in π at the instant $i \in \{0, ..., len(\pi)\text{-}1\}$, denoted by $\pi, i \models \varphi$, by inductively considering *subformulas* as follows:

$\pi, i \models x$	iff	$x \in \pi_i$;
$\pi, i \models \neg x$	iff	$x \notin \pi_i$;
$\pi, i \models (\varphi_1 \wedge \varphi_2)$	iff	$\pi, i \models \varphi_1$ and $\pi, i \models \varphi_2$;
$\pi, i \models (\varphi_1 \vee \varphi_2)$	iff	$\pi, i \models \varphi_1$ or $\pi, i \models \varphi_2$;
$\pi, i \models \mathsf{X}(\varphi')$	iff	$i < len(\pi)\text{-}1$ and $\pi, i+1 \models \varphi'$;
$\pi, i \models \mathsf{X_w}(\varphi')$	iff	$i < len(\pi)\text{-}1$ and $\pi, i+1 \models \varphi'$, or $i = len(\pi)\text{-}1$;
$\pi, i \models \mathsf{G}(\varphi')$	iff	$\forall j$ with $i \leq j < len(\pi)$ it holds that $\pi, j \models \varphi'$;
$\pi, i \models \mathsf{F}(\varphi')$	iff	$\exists j$ with $i \leq j < len(\pi)$ such that $\pi, j \models \varphi'$;
$\pi, i \models (\varphi_1 \mathbin{\mathsf{U}} \varphi_2)$	iff	$\exists j$ with $i \leq j < len(\pi)$ such that $\pi, j \models \varphi_2$ and $\forall k$ with $i \leq k < j$ it holds that $\pi, k \models \varphi_1$;
$\pi, i \models (\varphi_1 \mathbin{\mathsf{R}} \varphi_2)$	iff	$\forall i \leq j < len(\pi)$ it holds that $\pi, j \models \varphi_2$; or $\exists j$ with $i \leq j < len(\pi)$ such that $\pi, j \models \varphi_1$ and $\forall k$ with $i \leq k \leq j$ it holds that $\pi, k \models \varphi_2$;

Whenever $\pi, 0 \models \varphi$ holds, we say that π is a *model* of φ. In this case, the formula φ is said to be satisfiable. Note that if φ does not contain temporal operators, then π is a *propositional formula* and $\pi, 0 \models \varphi$ reduces to the notion of satisfiability for propositional logic. Moreover, note that, by inspecting the above definition, it follows that every satisfiable LTL$_f$ formula φ admits a model π such that $\pi_i \subseteq \mathcal{V}_\varphi$, for each $i \in \{0, ..., len(\pi)\text{-}1\}$.

Example 5 Consider the formula φ of Example 4 and the model π of φ with $len(\pi) = 6$ and such that π_0=$\{a\}$, π_1=$\{b\}$, π_2=$\{a, c\}$, π_3=$\{a\}$, π_4=$\{a, c\}$, π_5=$\{a\}$, and π_6=$\{a\}$. Figure 1 shows the subformulas required to hold based on the inductive definition of satisfiability.

At the top-most level, we have $\pi, 0 \models (a \wedge \neg b)$ and $\pi, 0 \models (\mathsf{F}((c \wedge \mathsf{G}(a))) \wedge \mathsf{X}(b))$. From the former condition, we derive $a \in \pi_0$ and $b \notin \pi_0$. From the latter, we get that $\pi, 0 \models \mathsf{F}((c \wedge \mathsf{G}(a)))$ and $\pi, 0 \models \mathsf{X}(b)$. In its turn, $\pi, 0 \models \mathsf{F}((c \wedge \mathsf{G}(a)))$ implies the existence of a time instant $j \in \{0, ..., 6\}$ such that $\pi, j \models (c \wedge \mathsf{G}(a))$. It can be checked that the property holds for $j = 2$ and $j = 4$: The figure illustrates the case $\pi, 4 \models (c \wedge \mathsf{G}(a))$, which implies $\pi, 4 \models c$, i.e., $c \in \pi_4$, and $\pi, 4 \models \mathsf{G}(a)$, i.e., $a \in \pi_j$ for all $j \in \{4, 5, 6\}$. Finally, $\pi, 0 \models \mathsf{X}(b)$ implies $\pi, 1 \models b$ and, therefore, $b \in \pi_1$. $\lhd$

To keep notation simple, in some cases, parenthesis will be omitted when this does not originate ambiguities in the application of the definition of satisfiability. In particular, we shall assume as usual that, in the evaluation of Boolean expressions, conjunction has precedence over disjunction. Furthermore, we often just write $\pi \models \varphi$ whenever $\pi, 0 \models \varphi$ holds.

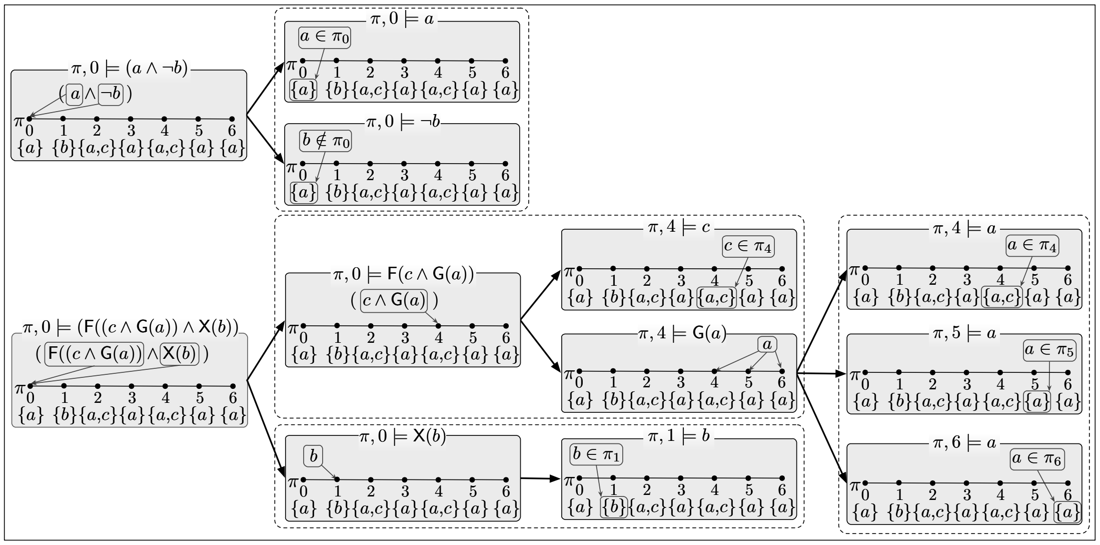

$\pi, 0 \models (a \wedge \neg b)$
$\pi, 0 \models a$
$a \in \pi_0$
$\pi, 0 \models \neg b$
$b \notin \pi_0$
$\pi, 0 \models (\mathsf{F}((c \wedge \mathsf{G}(a)) \wedge \mathsf{X}(b))$
$\pi, 0 \models \mathsf{F}(c \wedge \mathsf{G}(a))$
$\pi, 4 \models c$
$c \in \pi_4$
$\pi, 4 \models \mathsf{G}(a)$
$\pi, 4 \models a$
$a \in \pi_4$
$\pi, 5 \models a$
$a \in \pi_5$
$\pi, 6 \models a$
$a \in \pi_6$
$\pi, 0 \models \mathsf{X}(b)$
$\pi, 1 \models b$
$b \in \pi_1$
$\{a\}$ $\{b\}\{a,c\}\{a\}\{a,c\}\{a\}$ $\{a\}$

Figure 1: An LTL$_f$ formula φ, a trace π, and the graphical representation of the application of the inductive rules of the semantics for $\pi, 0 \models \varphi$.

2.2 From Finite Traces to Process Traces

LTL on Process Traces. A finite trace π is said to be a *process trace* if, for each time instant $i \in \{0, ..., len(\pi)\text{-}1\}$, π_i is a set consisting of one element only, i.e., $|\pi_i| = 1$.

In addition to LTL$_f$ our goal is to study the logic LTL$_p$, whose syntax coincides with the syntax of LTL$_f$ and whose semantics differs only in the fact that satisfiability (denoted by $\models_p$) is defined with respect to process traces only. In the following, a process trace that is a model of an LTL$_p$ formula is called a PT-model. For each $T \subseteq \{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}, \mathsf{R}\}$, the class $\langle T \rangle$-LTL$_p$ is defined analogously to the class $\langle T \rangle$-LTL$_f$.

Example 6 The model π of the formula φ in Example 5 is not a process trace. In fact, note that the LTL$_p$ formula $c \wedge \mathsf{G}(a)$ admits no PT-model, because it prescribes that, at the initial time instant, a and c must simultaneously hold. $\triangleleft$

An important application scenario for LTL$_p$ is in the context of the declarative specification and verification of business processes. Indeed, nowadays, process management systems (PMSs) represent a key technological infrastructure for developing advanced applications, where actors interact with each other while carrying out a number of activities according to certain business rules. PMSs provide support for designing the static aspects of the process, such as specifying precedence relationships among activities, and controlling their actual execution. However, to benefit of this support, designers are traditionally asked to define the process via some procedural language, where all scenarios and paths of execution have to be explicitly considered. Clearly, this is time consuming and might be even impossible when the process knowledge is incomplete or when the underlying schema is loosely structured, as it happens in knowledge-intensive applications (Di Ciccio, Marrella, & Russo, 2015). Moving from the above observation, recent research focused on developing declarative platforms for process management, which are more flexible in that any pos-

LTL$_p$ formula	name of the DECLARE template
$\varphi_1 = \mathsf{F}(a) \wedge \mathsf{G}(\neg a \vee \mathsf{X_w}(\mathsf{G}(\neg a)))$	$exactly1(a)$
$\varphi_2 = (\neg s \,\mathsf{U}\, a) \vee \mathsf{G}(\neg s)$	$precedence(a, s)$
$\varphi_3 = (\neg l \,\mathsf{U}\, s) \vee \mathsf{G}(\neg l)$	$precedence(s, l)$
$\varphi_4 = (\neg o \,\mathsf{U}\, s) \vee \mathsf{G}(\neg o)$	$precedence(s, o)$
$\varphi_5 = (\mathsf{F}(l) \vee \mathsf{F}(o)) \wedge (\mathsf{G}(\neg l) \vee \mathsf{G}(\neg o))$	$exclusive\text{-}choice(l, o)$
$\varphi_6 = \mathsf{G}(\neg l \vee \mathsf{F}(n))$	$response(l, n)$
$\varphi_7 = \mathsf{G}(\neg o \vee \mathsf{F}(n))$	$response(o, n)$

Table 1: The LTL$_p$ formulas associated with the constraints discussed in Example 7.

sible enactment is allowed unless a constraint, expressed in some formal logic, is known to hold and explicitly forbids it.

A noticeable example of a logic-based language used to declaratively specify and reason about business processes is DECLARE (Pesic & van der Aalst, 2006b). The language is based on a set of *templates*, each prescribing a constraint among the activities over which it is defined. Each constraint is formalized through an LTL formula interpreted on finite traces. In particular, DECLARE adopts an abstraction, which is rather popular in the business process and process mining communities, and according to which logs produced by transactional systems are just viewed as sequences of activities. Accordingly, DECLARE assumes that, at each point in time, one and only one task is executed, so that a process specification can be viewed as a set of LTL$_p$ formulas prescribing the temporal properties that are required to hold over the allowed process traces.

Example 7 Consider the process for gastric cancer surgical treatments discussed by Rovani, Maggi, de Leoni, and van der Aalst (2015). The process involves 5 activities, shortly denoted as $a, s, l, o,$ and n, on which the following constraints are defined: *(1)* the activity "First Hospital Admission" (a), consisting of the registration of the data of the patient, must be performed exactly once; *(2)* a is a prerequisite to subsequently proceed with the "Preoperative Screening" (s); *(3 and 4)* in its turn, s must be performed before any surgical treatment, either "Laparoscopic Gastrectomy" (l) or "Open Gastrectomy" (o); *(5)* only one of these two surgery treatments can be executed, and at least one of them is required; *(6 and 7)* after a treatment is performed, a "Nursing" (n) period will be eventually needed in order to monitor the patient.

Each of the above constraints, say $i \in \{1, ..., 7\}$, is equipped in DECLARE with an LTL$_p$ formula φ_i defining its formal semantics, and a process trace complies with the whole declarative specification if, and only if, it is a model of the conjunction $\Phi = \bigwedge_{i=1}^{7} \varphi_i$. For the sake of completeness, the formulas associated with the constraints discussed above are reported in Table 1, together with the name of the corresponding DECLARE template. $\lhd$

An important advantage of DECLARE compared to classical procedural languages for business processes is that, by reasoning on top of the logic specification, analysts can gain some useful insight about important properties of the specifications. For instance, the problem of verifying whether a given set of DECLARE constraints is *consistent*, i.e., whether there exists a process trace satisfying all the given constraints, just reduces to the satisfiability problem for LTL$_p$.

Example 8 Consider again the specification discussed in Example 7 and its associated formula $\Phi = \bigwedge_{i=1}^{7} \varphi_i$. Observe that the specification is consistent. This is witnessed, for instance, by the process trace $\{a\}, \{s\}, \{l\}, \{n\}$ which is a PT-model of Φ. Another process trace satisfying Φ is $\{a\}, \{s\}, \{l\}, \{s\}, \{l\}, \{n\}$; in particular, note that in this case activities s and l occur more than once. Similarly, $\{a\}, \{s\}, \{o\}, \{n\}, \{o\}, \{n\}$ and $\{a\}, \{s\}, \{o\}, \{o\}, \{n\}$ are two PT-models of Φ. On the other hand, the trace $\{a\}, \{s\}, \{l\}, \{o\}, \{n\}$ is not a PT-model of Φ, since it violates the constraint defined by φ_5 prescribing that l and o are mutually exclusive. $\lhd$

It is worthwhile noticing that DECLARE constraints are designed in a way that focusing on PT-models is crucial to preserve the intended semantics of the process specification. This clearly emerges by looking at the above exemplification: If we interpret the formula Φ as an LTL$_f$ formula (rather than as an LTL$_p$ formula), then the trace π with $len(\pi) = 1$ and $\pi_0 = \{a, s, l, n\}$ would be allowed as a model, despite it completely flattens the temporal precedences that are arise in the application at hand. It might even happen that a logic formula associated to a given specification is satisfiable when interpreted over finite traces, but does not admit any model according to the semantics of DECLARE. This is likely to occur when using DECLARE templates that involve the next or the weak next operators, as illustrated below.

Example 9 Recall from the work of Pesic and van der Aalst (2006b) that $chain\text{-}response(\alpha, \beta)$ is a DECLARE template prescribing that, whenever the activity α is executed, then the activity β must be executed next. Its formal semantics can be given by the formula $\mathsf{G}(\neg\alpha \vee \mathsf{X}(\beta))$.

Consider now a DECLARE specification consisting of the constraints $exactly1(a)$, $chain\text{-}response(a, b)$ and $chain\text{-}response(a, c)$—the formula associated with the former constraint is reported in Table 1. This specification is inconsistent according to DECLARE, since a is required to hold at some time instant, and then b and c are simultaneously required to hold in the subsequent time instant. On the other hand, a finite model satisfying the specification is just given by the trace $\pi = \pi_0, \pi_1$ such that $\pi_0 = \{a\}$ and $\pi_1 = \{b, c\}$. $\lhd$

As it emerged from the above discussion, investigating the theoretical underpinnings of LTL$_p$ and developing reasoning engines specifically tailored for that logic (rather than in general for LTL$_f$) would find prompt applications in the context of the growing body of literature related to the declarative specification of business processes. In the following subsection, we start our study of LTL$_p$ by focusing on some simple, yet useful properties.

2.3 Basic Properties of LTL$_p$

This section analyses some basic properties of LTL$_p$ that will be exploited in some of the results reported later in the paper.

2.3.1 NEGATION-FREE LTL$_p$ FORMULAS

An important peculiarity of LTL$_p$ is that, by focusing on PT-models, it might happen that we need to consider variables outside the "active domain" $\mathcal{V}_\varphi$ when we look for the satisfiability of a formula φ. For instance, if π is a PT-model of the formula $\neg a$, then we must have $|\pi_0| = 1$ and $a \notin \pi_0$. However, the following result shows that if an LTL$_p$ formula φ is satisfiable, then we are guaranteed about the existence of a PT-model π' of φ such that the set of propositional variables it contains is either a subset of $\mathcal{V}_\varphi$ or it includes at most one variable, say p, outside $\mathcal{V}_\varphi$; that is, $(\bigcup_{i=0}^{len(\pi')-1} \pi'_i \setminus \mathcal{V}_\varphi) \subseteq \{p\}$ always holds. To establish the result, we need a technical lemma which is, however, interesting on its own, as it shows that negation does not increase the expressiveness of

LTL$_p$ formulas. Formally, we exhibit a polynomial-time satisfiability preserving translation ξ_p from arbitrary LTL$_p$ formulas to negation-free ones.

Lemma 10 *Let φ be an LTL$_p$ formula and let $\hat{\mathcal{V}} = \mathcal{V}_\varphi \cup \{p\}$ be a set of variables where $p \in \mathcal{V} \setminus \mathcal{V}_\varphi$. For each variable $x \in \mathcal{V}_\varphi$, consider the formula ϕ_x consisting of the disjunction of all the variables in $\hat{\mathcal{V}}$ excluding x, that is, $\phi_x = \bigvee_{y \in \hat{\mathcal{V}} \setminus \{x\}} y$. Let $\xi_p(\varphi)$ be the negation-free LTL$_p$ formula obtained from φ by applying the polynomial-time translation function ξ_p inductively defined as follows over subformulas ψ of φ:*

- *If $\psi = x$ for some variable $x \in \mathcal{V}_\varphi$, then $\xi_p(\psi) = x$;*
- *If $\psi = \neg x$ for some variable $x \in \mathcal{V}_\varphi$, then $\xi_p(\psi) = \phi_x$;*
- *If $\psi = (\psi \vee \psi'')$, then $\xi_p(\psi) = (\xi_p(\psi') \vee \xi_p(\psi''))$;*
- *If $\psi = (\psi \wedge \psi'')$, then $\xi_p(\psi) = (\xi_p(\psi') \wedge \xi_p(\psi''))$;*
- *If $\psi = \mathsf{X}(\psi')$, then $\xi_p(\psi) = \mathsf{X}(\xi_p(\psi'))$;*
- *If $\psi = \mathsf{X_w}(\psi')$, then $\xi_p(\psi) = \mathsf{X_w}(\xi_p(\psi'))$;*
- *If $\psi = \mathsf{F}(\psi')$, then $\xi_p(\psi) = \mathsf{F}(\xi_p(\psi'))$;*
- *If $\psi = \mathsf{G}(\psi')$, then $\xi_p(\psi) = \mathsf{G}(\xi_p(\psi'))$.*
- *If $\psi = (\psi' \,\mathsf{U}\, \psi'')$, then $\xi_p(\psi) = (\xi_p(\psi') \,\mathsf{U}\, \xi_p(\psi''))$.*
- *If $\psi = (\psi' \,\mathsf{R}\, \psi'')$, then $\xi_p(\psi) = (\xi_p(\psi') \,\mathsf{R}\, \xi_p(\psi''))$.*

Then, the following properties hold:

(1) Assume that π is a PT-model of φ. Then, a PT-model π' of $\xi_p(\varphi)$ can be built from π in polynomial time such that $len(\pi') = len(\pi)$ and $\bigcup_{i=0}^{len(\pi')\text{-}1} \pi'_i \subseteq \hat{\mathcal{V}}$. That is, π' is defined over the variables in $\mathcal{V}_\varphi$ possibly including the additional variable p.

(2) Assume that π'' is a PT-model of $\xi_p(\varphi)$. Then, π'' is also a PT-model of φ.

Proof. We start by proving (1). The statement is not trivial when φ is satisfiable, so that we can take some PT-model π of φ. Based on π, let us build the finite process trace π', with $len(\pi) = len(\pi')$, as follows: For each $i \in \{0, ..., len(\pi)\text{-}1\}$, we set $\pi'_i = \pi_i$ if $\pi_i \subseteq \mathcal{V}_\varphi$, while we set $\pi'_i = \{p\}$ if $\pi_i \subseteq \mathcal{V} \setminus \mathcal{V}_\varphi$. Note that π' is well-defined, since $|\pi_i| = 1$ holds, for each $i \in \{0, ..., len(\pi)\text{-}1\}$. An example of this construction is reported in Figure 2.

We now show that π' is a PT-model of $\xi_p(\varphi)$. To this end, consider any subformula ψ of φ and any time instant $i \in \{0, ..., len(\pi)\text{-}1\}$ such that $\pi, i \models_p \psi$. We show that $\pi', i \models_p \xi_p(\psi)$ holds, too. This will entail that $\pi', 0 \models_p \xi_p(\varphi)$. We proceed by structural induction on the subformulas ψ.

Base case. If $\psi = x$ for some variable $x \in \mathcal{V}_\varphi$, then $\pi_i = \{x\}$ must hold. By construction, we have that $\pi'_i = \pi_i = \{x\}$ and $\xi_p(\psi) = x$. It follows that $\pi', i \models_p \xi_p(\psi)$.

On the other hand, if $\psi = \neg x$, then x does not belong to π_i and $\xi_p(\psi) = \phi_x = \bigvee_{y \in \hat{\mathcal{V}} \setminus \{x\}} y$. We distinguish two cases. If $\pi_i \subseteq \mathcal{V}_\varphi$, then we have $\pi'_i = \pi_i = \{y\}$ for some $y \in \mathcal{V}_\varphi \setminus \{x\}$. If $\pi_i \not\subseteq \mathcal{V}_\varphi$, then we have $\pi_i = \{p\} \subseteq \hat{\mathcal{V}} \setminus \mathcal{V}_\varphi$. In both cases, we conclude that $\pi', i \models_p \xi_p(\psi)$ holds.

Inductive step. Assume that the property holds over each subformula of ψ and assume that i is a time instant such that $\pi, i \models_p \psi$. We show that $\pi', i \models_p \xi_p(\psi)$ holds, too.

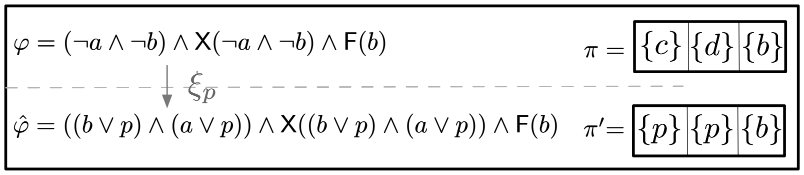

$\varphi = (\neg a \wedge \neg b) \wedge \mathsf{X}(\neg a \wedge \neg b) \wedge \mathsf{F}(b)$
$\pi = \{c\}\ \{d\}\ \{b\}$
ξ_p
$\hat{\varphi} = ((b \vee p) \wedge (a \vee p)) \wedge \mathsf{X}((b \vee p) \wedge (a \vee p)) \wedge \mathsf{F}(b)$
$\pi' = \{p\}\ \{p\}\ \{b\}$

Figure 2: An LTL$_p$ formula φ, a model π, the LTL$_p$ formula $\xi_p(\varphi)$, and the model π' built according to the construction in the proof of Lemma 10. Note that $\mathcal{V}_\varphi = \{a, b\}$, $\pi_0 = \{c\} \subseteq \mathcal{V} \setminus \mathcal{V}_\varphi$ and $\pi_1 = \{d\} \subseteq \mathcal{V} \setminus \mathcal{V}_\varphi$, so that $\pi'_0 = \pi'_1 = \{p\}$ holds.

- If $\psi = (\psi' \vee \psi'')$, then $\xi_p(\psi) = (\xi_p(\psi') \vee \xi_p(\psi''))$. By the inductive hypothesis on the subformulas ψ' and ψ'', we know that $\pi', i \models_p \xi_p(\psi')$ holds whenever $\pi, i \models_p \psi'$; and that $\pi', i \models_p \xi_p(\psi'')$ holds whenever $\pi, i \models_p \psi''$. Hence, we can derive $\pi', i \models_p \xi_p(\psi)$. The same line of reasoning applies to the case where $\psi = (\psi' \wedge \psi'')$, after noticing that $\xi_p(\psi) = (\xi_p(\psi') \wedge \xi_p(\psi''))$ holds in that case.

- If $\psi = \mathsf{X}(\psi')$, then $\pi, i \models_p \psi$ means that $\pi, i+1 \models_p \psi'$. By the inductive hypothesis on the subformula ψ', we know that $\pi', (i{+}1) \models_p \xi_p(\psi')$. Hence, we derive that $\pi', i \models_p \mathsf{X}(\xi_p(\psi'))$. By definition of ξ_p, this entails that $\pi', i \models_p \xi_p(\mathsf{X}(\psi'))$.

- If $\psi = \mathsf{X_w}(\psi')$, then $\pi, i \models \psi$ means that either $\pi, i+1 \models_p \psi'$ and $i < len(\pi)$-1, or $i = len(\pi)$-1. The first case is similar to that of X. As for the second one, the result follows directly by the observation that $len(\pi') = len(\pi)$.

- If $\psi = \mathsf{F}(\psi')$, then $\pi, i \models_p \psi$ implies the existence of a time instant $i' \in \{i, ..., len(\pi)\text{-}1\}$ such that $\pi, i' \models \psi'$. By the inductive hypothesis, we know that $\pi', i' \models_p \xi_p(\psi')$. Hence, $\pi', i \models_p \mathsf{F}(\xi_p(\psi'))$. By definition of ξ_p, this entails that $\pi', i \models_p \xi_p(\mathsf{F}(\psi'))$.

- If $\psi = \mathsf{G}(\psi')$, then $\pi, i \models_p \psi$ implies that for each time instant $i' \in \{i, ..., len(\pi)\text{-}1\}$, we have that $\pi, i' \models \psi'$. By the inductive hypothesis, we know that $\pi', i' \models_p \xi_p(\psi')$ holds, and we conclude that $\pi', i \models_p \mathsf{G}(\xi_p(\psi'))$. By definition of ξ_p, this entails that $\pi', i \models_p \xi_p(\mathsf{G}(\psi'))$.

- If $\psi = (\psi' \,\mathsf{U}\, \psi'')$, then $\pi, i \models_p \psi$ implies that there is a time instant $i' \geq i$ such that $\pi, i' \models \psi''$ and for each time instant $i'' \in \{i, ..., i'\text{-}1\}$ we have that $\pi, i'' \models \psi'$. By the inductive hypothesis, we know that $\pi', i' \models_p \xi_p(\psi'')$ and $\pi', i'' \models_p \xi_p(\psi')$, for each $i'' \in \{i, ..., i'\text{-}1\}$, hold. Hence, we conclude that $\pi', i \models_p (\xi_p(\psi') \,\mathsf{U}\, \xi_p(\psi''))$ holds. By definition of ξ_p, this entails that $\pi', i \models_p \xi_p(\psi' \,\mathsf{U}\, \psi'')$.

- If $\psi = (\psi' \,\mathsf{R}\, \psi'')$, then $\pi, i \models_p \psi$ implies either that $\pi, i' \models_p \psi''$ for all $i' \geq i$, or there is a time instant $i''' \geq i$ such that $\pi, i''' \models_p \psi'$ and for each time instant $i'' \in \{i, ..., i'''\}$ we have that $\pi, i'' \models_p \psi''$. In the first case, by the inductive hypothesis, we know that $\pi', i' \models_p \xi_p(\psi'')$ holds for each $i' \geq i$ and, hence, we conclude that $\pi', i \models_p (\xi_p(\psi') \,\mathsf{R}\, \xi_p(\psi''))$ holds. In the second case, by the inductive hypothesis, we know that $\pi', i''' \models_p \xi_p(\psi')$ and $\pi', i'' \models_p \xi_p(\psi'')$, for each $i'' \in \{i, ..., i'''\}$, hold. Hence, again, we conclude that $\pi', i \models_p (\xi_p(\psi') \,\mathsf{R}\, \xi_p(\psi''))$ holds. By definition of ξ_p, this entails that $\pi', i \models_p \xi_p(\psi' \,\mathsf{R}\, \psi'')$.

Let us now move to show (2). Note that combined with (1), the result will imply that φ is satisfiable if, and only if, $\xi_p(\varphi)$ is satisfiable. Assume that π'' is a PT-model of $\xi_p(\varphi)$. We have to show that π'' is also a PT-model of φ. We proceed again by structural induction by considering, this time, every subformula $\xi_p(\psi)$ of $\xi_p(\varphi)$ and by showing that, for each time instant $i \in \{0, ..., len(\pi'')\text{-}1\}$ such that $\pi'', i \models_p \xi_p(\psi)$, we have that $\pi'', i \models_p \psi$. Eventually, this will entail that $\pi'', 0 \models_p \varphi$, as we know that $\pi'', 0 \models_p \xi_p(\varphi)$.

Base case. If $\xi_p(\psi) = x$ for some variable $x \in \mathcal{V}_\varphi$, then $\pi''_i = \{x\}$ and $\psi = x$ must hold. It follows that $\pi'', i \models_p \psi$. On the other hand, if $\xi_p(\psi) = \phi_x = \bigvee_{y \in \hat{\mathcal{V}} \setminus \{x\}} y$, then either $\pi''_i = \{y\}$, for some variable $y \in \mathcal{V}_\varphi \setminus \{x\}$ or $\pi''_i = \{p\}$; indeed, note that given our focus on process traces, the whole disjunction ϕ_x actually behaves as an exclusive choice among the variables on which it is defined. Since $\psi = \neg x$, then $\pi'', i \models_p \psi$ holds.

Inductive step. Assume that the property holds over each subformula of $\xi_p(\psi)$ and assume that i is a time instant such that $\pi'', i \models_p \xi_p(\psi)$. We show that $\pi'', i \models_p \psi$ holds, too.

- If $\xi_p(\psi) = \xi_p(\psi' \vee \psi'')$, then we have that $\xi_p(\psi) = (\xi_p(\psi') \vee \xi_p(\psi''))$, by definition of ξ_p. Hence, $\pi'', i \models_p \xi_p(\psi)$ directly entails, by induction on ψ' and ψ'', that $\pi'', i \models_p \psi' \vee \psi''$. The same reasoning applies to the case where $\xi_p(\psi) = \xi_p(\psi' \wedge \psi'')$.

- If $\xi_p(\psi) = \xi_p(\mathsf{X}(\psi'))$, then we have that $\xi_p(\psi) = \mathsf{X}(\xi_p(\psi'))$, by definition of ξ_p. Hence, $\pi'', i \models_p \xi_p(\psi)$ implies that $\pi'', (i+1) \models_p \xi_p(\psi')$. By the inductive hypothesis, we know that $\pi'', (i+1) \models_p \psi'$. Hence, we derive that $\pi'', i \models_p \mathsf{X}(\psi')$.

- If $\xi_p(\psi) = \xi_p(\mathsf{X_w}(\psi'))$, then we have that $\xi_p(\psi) = \mathsf{X_w}(\xi_p(\psi'))$, by definition of ξ_p. Hence, $\pi'', i \models_p \xi_p(\psi)$ means that either $\pi'', (i+1) \models_p \xi_p(\psi')$ and $i < len(\pi'')\text{-}1$, or $i = len(\pi'')\text{-}1$. In the latter case, the result is immediate. In the former case, the result can be proven with the same line of reasoning used when analyzing X.

- If $\xi_p(\psi) = \xi_p(\mathsf{F}(\psi'))$, then we have that $\xi_p(\psi) = \mathsf{F}(\xi_p(\psi'))$, by definition of ξ_p. Hence, $\pi'', i \models_p \xi_p(\psi)$ implies the existence of a time instant $i' \in \{i, ..., len(\pi'')\text{-}1\}$ such that $\pi'', i' \models_p \xi_p(\psi')$. By the inductive hypothesis, we know that $\pi'', i' \models_p \psi'$. Hence, we derive that $\pi'', i \models_p \mathsf{F}(\psi')$.

- If $\xi_p(\psi) = \xi_p(\mathsf{G}(\psi'))$, then we have that $\xi_p(\psi) = \mathsf{G}(\xi_p(\psi'))$, by definition of ξ_p. Hence, $\pi'', i \models_p \xi_p(\psi)$ implies that for each time instant $i' \in \{i, ..., len(\pi'')\text{-}1\}$, we have that $\pi'', i' \models_p \xi_p(\psi')$. By the inductive hypothesis, we know that $\pi'', i' \models_p \psi'$ holds and, we conclude that $\pi'', i \models_p \mathsf{G}(\psi')$.

- If $\xi_p(\psi) = \xi_p(\psi' \,\mathsf{U}\, \psi'')$, then we actually have that $\xi_p(\psi) = (\xi_p(\psi') \,\mathsf{U}\, \xi_p(\psi''))$, by definition of ξ_p. Hence, $\pi'', i \models_p \xi_p(\psi)$ implies that there is a time instant $i' \geq i$ such that $\pi'', i' \models_p \xi_p(\psi'')$ and, for each time instant $i'' \in \{i, ..., i'\text{-}1\}$, we have that $\pi'', i'' \models_p \xi_p(\psi')$. By the inductive hypothesis, we know that $\pi'', i' \models_p \psi''$ and $\pi'', i'' \models_p \psi'$, for each $i'' \in \{i, ..., i'\text{-}1\}$, hold. Hence, we conclude that $\pi'', i \models_p (\psi' \,\mathsf{U}\, \psi'')$ holds.

- If $\xi_p(\psi) = \xi_p(\psi' \,\mathsf{R}\, \psi'')$, then we actually have that $\xi_p(\psi) = (\xi_p(\psi') \,\mathsf{R}\, \xi_p(\psi''))$, by definition of ξ_p. Hence, $\pi'', i \models_p \xi_p(\psi)$ implies either that that $\pi'', i' \models_p \xi_p(\psi'')$ for each time instant $i' \geq i$, or that there is a time instant $i''' \geq i$ such that $\pi'', i''' \models_p \xi_p(\psi')$ and, for each time instant $i'' \in \{i, ..., i'''\}$, we have that $\pi'', i'' \models_p \xi_p(\psi'')$. In the first case, by the inductive hypothesis, we know that $\pi'', i' \models_p \psi''$ for each $i' \geq i$, holds. In the second case, by the inductive hypothesis, we know that $\pi'', i''' \models_p \psi'$ and $\pi'', i'' \models_p \psi''$, for each $i'' \in \{i, ..., i'''\}$, hold. Hence, we conclude that $\pi'', i \models_p (\psi' \mathsf{R}\, \psi'')$ holds. □

We have already noticed in the proof of the above result that φ is satisfiable if, and only if, $\xi_p(\varphi)$ is satisfiable. In fact, by combining points *(1)* and *(2)* in Lemma 10, it directly follows the desired characterization on the PT-models of LTL$_p$ formulas.

Corollary 11 *Let φ be an* LTL$_p$ *formula and let $\hat{\mathcal{V}} = \mathcal{V}_\varphi \cup \{p\}$ be a set of variables where $p \in \mathcal{V} \backslash \mathcal{V}_\varphi$. If φ is satisfiable, then there exists a* PT*-model π' of φ such that $\bigcup_{i=0}^{len(\pi')\text{-}1} \pi'_i \subseteq \hat{\mathcal{V}}$.*

2.3.2 LTL FORMULAS AND NEGATION NORMAL FORM

We leave the section by recalling that, throughout the paper, we shall consider (LTL$_f$ and LTL$_p$) formulas φ in negation normal form. This is done without loss of generality. Indeed, assume that a formula φ is given where subformulas having the form $\neg\varphi'$ are allowed, with the intended meaning that $\pi, i \models \neg\varphi'$ holds if, and only if, $\pi, i \models \varphi'$ does not hold. Then, the following properties—that directly follow by the semantics—can be used to transform φ into an equivalent formula in negation normal form:

$\pi, i \models \neg(\varphi_1 \wedge \varphi_2)$	iff	$\pi, i \models (\neg\varphi_1) \vee (\neg\varphi_2)$;
$\pi, i \models \neg(\varphi_1 \vee \varphi_2)$	iff	$\pi, i \models (\neg\varphi_1) \wedge (\neg\varphi_2)$;
$\pi, i \models \neg(\mathsf{X}(\varphi'))$	iff	$\pi, i \models \mathsf{X}_\mathsf{w}(\neg\varphi')$;
$\pi, i \models \neg(\mathsf{X}_\mathsf{w}(\varphi'))$	iff	$\pi, i \models \mathsf{X}(\neg\varphi')$;
$\pi, i \models \neg(\mathsf{G}(\varphi'))$	iff	$\pi, i \models \mathsf{F}(\neg\varphi')$;
$\pi, i \models \neg(\mathsf{F}(\varphi'))$	iff	$\pi, i \models \mathsf{G}(\neg\varphi')$;
$\pi, i \models \neg(\varphi_1 \,\mathsf{U}\, \varphi_2)$	iff	$\pi, i \models (\neg\varphi_1)\mathsf{R}(\neg\varphi_2)$;
$\pi, i \models \neg(\varphi_1 \,\mathsf{R}\, \varphi_2)$	iff	$\pi, i \models (\neg\varphi_1)\mathsf{U}(\neg\varphi_2)$;

3. Useful Relationships between LTL$_p$ and LTL$_f$

In this section, we study mechanisms to "reformulate" certain LTL$_p$ fragments in terms of LTL$_f$ formulas, and vice versa. More formally, all reformulations are actually polynomial-time satisfiability preserving translations. The results are of interest in their own, as they shed lights on relationships that exist between LTL$_p$ and LTL$_f$. Furthermore, they will play a key role to simplify our subsequent elaborations and, in particular, the complexity analysis of LTL$_p$.

3.1 From LTL$_p$ to LTL$_f$

Let us first address the question of whether there exists a polynomial-time satisfiability preserving translation from LTL$_p$ to LTL$_f$. With this respect, for any *negation-free* LTL$_p$ formula φ, the reader might immediately notice that the following LTL$_f$ formula

$$\varphi \wedge \mathsf{G}\Big(\bigvee_{y \in \mathcal{V}_\varphi} (y \wedge \bigwedge_{y' \in \mathcal{V}_\varphi \backslash \{y\}} \neg y')\Big)$$

precisely encodes the semantics of process traces.

Dealing with an arbitrary LTL$_p$ formula φ is slightly more complex. Indeed, we have to exploit the properties pointed out in Section 2.3 and, in particular, Corollary 11, guaranteeing that, as far as the satisfiability is concerned, we have to additionally consider one variable, say p, not contained in $\mathcal{V}_\varphi$. The result below incorporates this observation and exhibits a polynomial-time satisfiability preserving translation ξ_{pf}, which moreover is not based on the use of the temporal operator G.

Theorem 12 *Let φ be an* LTL$_p$ *formula and let $\hat{\mathcal{V}} = \mathcal{V}_\varphi \cup \{p\}$ be a set of variables where $p \in \mathcal{V} \setminus \mathcal{V}_\varphi$. For each literal $\ell \in \{x, \neg x\}$, where x is any variable in $\mathcal{V}_\varphi$, consider the following formula:*

$$\phi_\ell = \ell \wedge \left(\bigvee_{y \in \hat{\mathcal{V}}} (y \wedge \bigwedge_{y' \in \hat{\mathcal{V}} \setminus \{y\}} \neg y') \right).$$

Let $\xi_{pf}(\varphi)$ be the LTL$_f$ *formula obtained from φ by applying the polynomial-time translation function ξ_{pf} inductively defined as follows over subformulas ψ of φ:*

- *If $\psi = x$ for some variable $x \in \mathcal{V}_\varphi$, then $\xi_{pf}(\psi) = \phi_x$;*
- *If $\psi = \neg x$ for some variable $x \in \mathcal{V}_\varphi$, then $\xi_{pf}(\psi) = \phi_{\neg x}$;*
- *If $\psi = (\psi \vee \psi'')$, then $\xi_{pf}(\psi) = (\xi_{pf}(\psi') \vee \xi_{pf}(\psi''))$;*
- *If $\psi = (\psi \wedge \psi'')$, then $\xi_{pf}(\psi) = (\xi_{pf}(\psi') \wedge \xi_{pf}(\psi''))$;*
- *If $\psi = \mathsf{X}(\psi')$, then $\xi_{pf}(\psi) = \mathsf{X}(\xi_{pf}(\psi'))$;*
- *If $\psi = \mathsf{X_w}(\psi')$, then $\xi_{pf}(\psi) = \mathsf{X_w}(\xi_{pf}(\psi'))$;*
- *If $\psi = \mathsf{F}(\psi')$, then $\xi_{pf}(\psi) = \mathsf{F}(\xi_{pf}(\psi'))$;*
- *If $\psi = \mathsf{G}(\psi')$, then $\xi_{pf}(\psi) = \mathsf{G}(\xi_{pf}(\psi'))$.*
- *If $\psi = (\psi' \,\mathsf{U}\, \psi'')$, then $\xi_{pf}(\psi) = (\xi_{pf}(\psi') \,\mathsf{U}\, \xi_{pf}(\psi''))$.*
- *If $\psi = (\psi' \,\mathsf{R}\, \psi'')$, then $\xi_{pf}(\psi) = (\xi_{pf}(\psi') \,\mathsf{R}\, \xi_{pf}(\psi''))$.*

Then, the following properties hold:

(1) Assume that π' is a PT*-model of φ (by Corollary 11, without loss of generality) such that $\bigcup_{i=0}^{len(\pi')\text{-}1} \pi'_i \subseteq \hat{\mathcal{V}}$. Then, π' is a model of $\xi_{pf}(\varphi)$.*

(2) Assume that π'' is a model of $\xi_{pf}(\varphi)$. Then, a PT*-model π''' of φ can be built from π'' in polynomial time such that $len(\pi''') = len(\pi'')$.*

Proof. The statement can be proven with arguments that are similar to those used in the proof of Lemma 10—details are provided below, for the sake of completeness.

We start by proving *(1)*. Assume that π' is a PT-model of φ. We have to show that π' is also a model of $\xi_{pf}(\varphi)$. We proceed by structural induction by considering every subformula ψ of φ and by showing that, for each time instant $i \in \{0, ..., len(\pi')\text{-}1\}$ such that $\pi', i \models_p \psi$, we have that $\pi', i \models \xi_{pf}(\psi)$. This will entail that $\pi', 0 \models \xi_{pf}(\varphi)$.

Base case. If $\psi = x$ for some variable $x \in \mathcal{V}_\varphi$, then we have $\xi_{pf}(\psi) = x \wedge \bigvee_{y \in \hat{\mathcal{V}}}(y \wedge \bigwedge_{y' \in \hat{\mathcal{V}} \setminus \{y\}} \neg y')$. Since $\pi', i \models_p \psi$ holds, we have that $\pi'_i = \{x\}$. Thus, $\pi', i \models \xi_{pf}(\psi)$.

If $\psi = \neg x$, then we have $\xi_{pf}(\psi) = \neg x \wedge \bigvee_{y \in \hat{\mathcal{V}}}(y \wedge \bigwedge_{y' \in \hat{\mathcal{V}} \setminus \{y\}} \neg y')$. Since $\pi', i \models_p \psi$ holds, we have that $\pi'_i = \{z\}$, for some variable $z \in \mathcal{V}_\varphi \cup \{p\} \setminus \{x\}$. Thus, $\pi', i \models \xi_{pf}(\psi)$.

Inductive step. Assume that the property holds over each subformula of ψ and assume that i is a time instant such that $\pi', i \models_p \psi$. We show that $\pi', i \models \xi_{pf}(\psi)$ holds, too.

- If $\psi = \psi' \vee \psi''$, then we have that $\xi_{pf}(\psi) = (\xi_{pf}(\psi') \vee \xi_{pf}(\psi''))$, by definition of ξ_{pf}. Hence, $\pi', i \models_p \psi$ directly entails, by induction on ψ' and ψ'', that $\pi', i \models (\xi_{pf}(\psi') \vee \xi_{pf}(\psi''))$. The same reasoning applies to the case where $\psi = \psi' \wedge \psi''$.

- If $\psi = \mathsf{X}(\psi')$, then we have that $\xi_{pf}(\psi) = \mathsf{X}(\xi_{pf}(\psi'))$, by definition of ξ_{pf}. Hence, $\pi', i \models_p \psi$ implies that $\pi', (i+1) \models_p \psi'$. By the inductive hypothesis, we know that $\pi', (i+1) \models \xi_{pf}(\psi')$. Hence, we derive that $\pi', i \models \mathsf{X}(\xi_{pf}(\psi'))$ and, thus, $\pi', i \models \xi_{pf}(\mathsf{X}(\psi'))$.

- If $\psi = \mathsf{X_w}(\psi')$, then we have that $\xi_{pf}(\psi) = \mathsf{X_w}(\xi_{pf}(\psi'))$, by definition of ξ_{pf}. Hence, $\pi', i \models_p \psi$ means that either $\pi', (i+1) \models_p \psi'$ and $i < len(\pi')$-1, or $i = len(\pi')$-1. In the latter case, the result directly follows. In the former case, the result can be proven with the same line of reasoning used when analyzing X.

- If $\psi = \mathsf{F}(\psi')$, then we actually have that $\xi_{pf}(\psi) = \mathsf{F}(\xi_{pf}(\psi'))$, by definition of ξ_{pf}. Hence, $\pi', i \models_p \psi$ implies the existence of a time instant $i' \in \{i, ..., len(\pi')\text{-}1\}$ such that $\pi', i' \models_p \psi'$. By the inductive hypothesis, we know that $\pi', i' \models \xi_{pf}(\psi')$. Hence, we derive that $\pi', i \models \mathsf{F}(\xi_{pf}(\psi')$ and, thus, $\pi', i \models \xi_{pf}(\mathsf{F}(\psi'))$.

- If $\psi = \mathsf{G}(\psi')$, then we have that $\xi_{pf}(\psi) = \mathsf{G}(\xi_{pf}(\psi'))$, by definition of ξ_{pf}. Hence, $\pi', i \models_p \psi$ implies that for each time instant $i' \in \{i, ..., len(\pi')\text{-}1\}$, we have that $\pi', i' \models_p \psi'$. By the inductive hypothesis, for such time instants i', we know that $\pi', i' \models \xi_{pf}(\psi')$ holds. We conclude that $\pi', i \models \mathsf{G}(\xi_{pf}(\psi'))$ and, thus, $\pi', i \models \xi_{pf}(\mathsf{G}(\psi'))$.

- If $\psi = (\psi' \;\mathsf{U}\; \psi'')$, then we have that $\xi_{pf}(\psi) = (\xi_{pf}(\psi') \;\mathsf{U}\; \xi_{pf}(\psi''))$, by definition of ξ_{pf}. Hence, $\pi', i \models_p \psi$ implies that there is a time instant $i' \geq i$ such that $\pi', i' \models_p \psi''$ and, for each time instant $i'' \in \{i, ..., i'\text{-}1\}$, we have that $\pi', i'' \models_p \psi'$. By the inductive hypothesis, we know that $\pi', i' \models \xi_{pf}(\psi'')$ and $\pi', i'' \models \xi_{pf}(\psi')$, for each $i'' \in \{i, ..., i'\text{-}1\}$, hold. Hence, we conclude that $\pi', i \models (\xi_{pf}(\psi') \;\mathsf{U}\; \xi_{pf}(\psi''))$ holds and, thus, $\pi', i \models \xi_{pf}(\psi' \;\mathsf{U}\; \psi'')$.

- If $\psi = (\psi' \;\mathsf{R}\; \psi'')$, then we have that $\xi_{pf}(\psi) = (\xi_{pf}(\psi') \;\mathsf{R}\; \xi_{pf}(\psi''))$, by definition of ξ_{pf}. Now, $\pi', i \models_p \psi$ implies that either $\pi', i' \models_p \psi''$ for each time instant $i' \geq i$, or there is a time instant $i''' \geq i$ such that $\pi', i''' \models_p \psi'$ and, for each time instant $i'' \in \{i, ..., i'''\}$, we have that $\pi', i'' \models_p \psi''$. In the first case, by the inductive hypothesis, we know that $\pi', i' \models \xi_{pf}(\psi'')$, for each $i' \geq i$, holds. In the second case, by the inductive hypothesis, we know that $\pi', i''' \models \xi_{pf}(\psi')$ and $\pi', i'' \models \xi_{pf}(\psi'')$, for each $i'' \in \{i, ..., i'''\}$, hold. Hence, we conclude that $\pi', i \models (\xi_{pf}(\psi') \;\mathsf{R}\; \xi_{pf}(\psi''))$ holds and, thus, $\pi', i \models \xi_{pf}(\psi' \;\mathsf{R}\; \psi'')$.

Now, we move to prove *(2)*. Let π'' be a model of $\xi_{pf}(\varphi)$ and let $\pi''' = \pi'''_0, ..., \pi'''_{len(\pi'')\text{-}1}$ be the trace such that $\pi'''_i = \pi''_i$ if $|\pi''_i| = 1$ and $\pi'''_i = \{p\}$ if $|\pi''_i| \neq 1$, for each $i \in \{0, ..., len(\pi'') - 1\}$. We proceed by structural induction by considering every subformula $\xi_{pf}(\psi)$ of $\xi_{pf}(\varphi)$ and by showing that, for each time instant $i \in \{0, ..., len(\pi'')\text{-}1\}$ such that $\pi'', i \models \xi_{pf}(\psi)$, we have that $\pi''', i \models_p \psi$. This will entail that $\pi''', 0 \models_p \varphi$.

Base case. If $\xi_{pf}(\psi) = x \wedge \bigvee_{y \in \hat{\mathcal{V}}}(y \wedge \bigwedge_{y' \in \hat{\mathcal{V}} \setminus \{y\}} \neg y')$ holds for some variable $x \in \mathcal{V}_\varphi$, then $\psi = x$. Since $\pi'', i \models \xi_{pf}(\psi)$, we have $\pi''_i = \{x\}$ and, hence, $\pi'''_i = \{x\}$. Thus, $\pi''', i \models_p \psi$.

If $\xi_{pf}(\psi) = \neg x \wedge \bigvee_{y \in \hat{\mathcal{V}}}(y \wedge \bigwedge_{y' \in \hat{\mathcal{V}} \setminus \{y\}} \neg y')$, then $\psi = \neg x$. Since $\pi'', i \models \xi_{pf}(\psi)$ holds, we can have two cases: either $\pi''_i = \{z\}$ for some variable $z \in \mathcal{V}_\varphi \setminus \{x\}$, or $\pi'_i = \{p\}$. In both cases $\pi'''_i = \pi''_i$ and we can conclude that $\pi''', i \models_p \neg x$ holds.

Inductive step. Assume that the property holds over each subformula of $\xi_{pf}(\psi)$ and assume that i is a time instant such that $\pi'', i \models \xi_{pf}(\psi)$. We show that $\pi'', i \models_p \psi$ holds, too.

- If $\xi_{pf}(\psi) = \xi_{pf}(\psi' \vee \psi'')$, then we have that $\xi_{pf}(\psi) = (\xi_{pf}(\psi') \vee \xi_{pf}(\psi''))$, by definition of ξ_{pf}. Hence, $\pi'', i \models \xi_{pf}(\psi)$ directly entails, by induction on ψ' and ψ'', that $\pi''', i \models_p \psi' \vee \psi''$. The same reasoning applies to the case where $\xi_{pf}(\psi) = \xi_{pf}(\psi' \wedge \psi'')$.

- If $\xi_{pf}(\psi) = \xi_{pf}(\mathsf{X}(\psi'))$, then we have that $\xi_{pf}(\psi) = \mathsf{X}(\xi_{pf}(\psi'))$, by definition of ξ_{pf}. Hence, $\pi'', i \models \xi_{pf}(\psi)$ implies that $\pi'', (i+1) \models \xi_{pf}(\psi')$. By the inductive hypothesis, we know that $\pi''', (i+1) \models_p \psi'$. Hence, we derive that $\pi''', i \models_p \mathsf{X}(\psi)$.

- If $\xi_{pf}(\psi) = \xi_{pf}(\mathsf{X_w}(\psi'))$, then we have that $\xi_{pf}(\psi) = \mathsf{X_w}(\xi_{pf}(\psi'))$, by definition of ξ_{pf}. Hence, $\pi'', i \models \xi_{pf}(\psi)$ means that either $\pi'', (i+1) \models \xi_{pf}(\psi')$ and $i < len(\pi'')\text{-}1$, or $i = len(\pi'')\text{-}1$. In the latter case, the result directly follows. In the former case, the result can be proven with the same line of reasoning used when analyzing X.

- If $\xi_{pf}(\psi) = \xi_{pf}(\mathsf{F}(\psi'))$, then we have that $\xi_{pf}(\psi) = \mathsf{F}(\xi_{pf}(\psi'))$, by definition of ξ_{pf}. Hence, $\pi'', i \models \xi_{pf}(\psi)$ implies the existence of a time instant $i' \in \{i, ..., len(\pi'')\text{-}1\}$ such that $\pi'', i' \models_p \xi_{pf}(\psi')$. By the inductive hypothesis, we know that $\pi''', i' \models_p \psi'$. Hence, we derive that $\pi''', i \models_p \mathsf{F}(\psi')$.

- If $\xi_{pf}(\psi) = \xi_{pf}(\mathsf{G}(\psi'))$, then we have that $\xi_{pf}(\psi) = \mathsf{G}(\xi_{pf}(\psi'))$, by definition of ξ_{pf}. Hence, $\pi'', i \models \xi_{pf}(\psi)$ implies that for each time instant $i' \in \{i, ..., len(\pi'')\text{-}1\}$, we have that $\pi'', i' \models \xi_{pf}(\psi')$. By the inductive hypothesis, we know that $\pi''', i' \models_p \psi'$ holds and, we conclude that $\pi''', i \models_p \mathsf{G}(\psi')$.

- If $\xi_{pf}(\psi) = \xi_{pf}(\psi' \,\mathsf{U}\, \psi'')$, then we have that $\xi_{pf}(\psi) = (\xi_{pf}(\psi') \,\mathsf{U}\, \xi_{pf}(\psi''))$, by definition of ξ_{pf}. Hence, $\pi'', i \models \xi_{pf}(\psi)$ implies that there is a time instant $i' \geq i$ such that $\pi'', i' \models \xi_{pf}(\psi'')$ and, for each time instant $i'' \in \{i, ..., i'\text{-}1\}$, we have that $\pi'', i'' \models \xi_{pf}(\psi')$. By the inductive hypothesis, we know that $\pi''', i' \models_p \psi''$ and $\pi''', i'' \models_p \psi'$, for each $i'' \in \{i, ..., i'\text{-}1\}$, hold. Hence, we conclude that $\pi''', i \models_p (\psi' \,\mathsf{U}\, \psi'')$ holds.

- If $\xi_{pf}(\psi) = \xi_{pf}(\psi' \,\mathsf{R}\, \psi'')$, then we have that $\xi_{pf}(\psi) = (\xi_{pf}(\psi') \,\mathsf{R}\, \xi_{pf}(\psi''))$, by definition of ξ_{pf}. Hence, $\pi'', i \models \xi_{pf}(\psi)$ implies that either $\pi'', i' \models \xi_{pf}(\psi'')$ for each time instant $i' \geq i$, or there is a time instant $i''' \geq i$ such that $\pi'', i''' \models \xi_{pf}(\psi')$ and, for each time instant $i'' \in \{i, ..., i'''\}$, we have that $\pi'', i'' \models \xi_{pf}(\psi'')$. In the first case, by the inductive hypotesis, we know that $\pi''', i' \models_p \psi''$, for each $i' \geq i$, holds. In the second case, by the inductive hypothesis, we know that $\pi''', i''' \models_p \psi''$ and $\pi''', i'' \models_p \psi'$, for each $i'' \in \{i, ..., i'''\}$, hold. Hence, we conclude that $\pi''', i \models_p (\psi' \,\mathsf{R}\, \psi'')$ holds.

□

Note that the above result immediately provides us with upper bounds on the complexity of reasoning about LTL$_p$, as soon as the complexity of reasoning about LTL$_f$ is known. In particular, the following is immediate after the work of De Giacomo and Vardi (2013), showing that satisfiability of LTL$_f$ formulas is in PSPACE.

Corollary 13 *Satisfiability of* LTL$_p$ *formulas is in* PSPACE.

In addition, for all fragments defined with $T \subseteq \{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}, \mathsf{R}\}$, upper bounds on the complexity of $\langle T \rangle$-LTL$_p$ formulas will derive after we establish in Section 5 the corresponding results for LTL$_f$ restricted on the same syntactic fragment defined by T. In fact, the translation ξ_{pf} of Theorem 12 produces an LTL$_f$ formula where negation occurs, whereas we know from Lemma 10 that negation can be removed from LTL$_p$ formulas without altering their expressiveness (and complexity). This suggests that LTL$_p$ and LTL$_f$ might behave in a completely different way if restricted over negation-free fragments—this will be confirmed by our analysis in Section 5 (see Figure 4).

3.2 From LTL$_f$ to LTL$_p$

We now look for polynomial-time satisfiability preserving translations from LTL$_f$ to LTL$_p$. The goal is to assess whether we can establish lower bounds on the complexity of reasoning about LTL$_p$, based on the knowledge of the complexity of LTL$_f$. Combined with Theorem 5 such lower bounds would immediately characterize the complexity of LTL$_p$ in terms of that of LTL$_f$. As a matter of fact, however, in the analysis that follows we are able to exhibit translations only for certain syntactic fragments defined in terms of the temporal operators that are allowed. In Section 5, we shall show that this is not by chance, since it will emerge that the two logics have different computational properties on some fragments (see, again, Figure 4).

We start by considering classes of LTL$_p$ formulas where the operator X is allowed. In this case, we can exhibit a polynomial-time satisfiability preserving translation ξ_{fp} that, given an LTL$_f$ formula φ, returns the LTL$_p$ formula $\xi_{fp}(\varphi)$ that forces each state of any finite model of φ to be encoded as a succession of m states each one being univocally associated with one of the m variables on which φ is defined. The translation is more complex than those discussed so far (in Lemma 10 and Theorem 12), so that we just state its existence in the statement of the result below, while deferring the details of its construction to the proof.

Theorem 14 *A polynomial-time (satisfiability preserving) translation ξ_{fp} exists that, given a $\langle T \rangle$-*LTL$_f$ *formula φ, returns a negation-free $\langle T \cup \{\mathsf{X}\} \rangle$-*LTL$_p$ *formula $\xi_{fp}(\varphi)$ enjoying the following properties:*

(1) *Assume that π is a model of φ. Then, a* PT-*model π' of $\xi_{fp}(\varphi)$ can be built from π in polynomial time such that $len(\pi') = len(\pi) \times (|\mathcal{V}_\varphi| + 1)$;*

(2) *Assume that π'' is a* PT-*model of $\xi_{fp}(\varphi)$. Then, a model π''' of φ can be built from π'' in polynomial time such that $len(\pi''') \leq len(\pi'')$.*

Proof. Let φ be a $\langle T \rangle$-LTL$_f$ formula with $\mathcal{V}_\varphi = \{x_1, ..., x_m\}$. Based on φ, we build a negation-free $\langle T \cup \{\mathsf{X}\} \rangle$-LTL$_p$ formula $\xi_{fp}(\varphi)$ over the set of variables

$$\mathcal{V}_{\xi_{fp}(\varphi)} = \{x_j, \bar{x}_j \mid 1 \leq j \leq m\} \cup \{t\}.$$

Intuitively, each variable x_j is associated with its overlined version $\bar{x}_j$, standing for its negation. In addition, t is a special variable that will keep track of the time instants for the models of φ.

Consider the $\langle T \cup \{\mathsf{X}\}\rangle$-LTL$_p$ formula A that enforces the distinguished variable t to hold at the current time instant, and that enforces either x_j or $\bar{x}_j$ to hold at the j-th time instant following the current one, with $j \in \{1, ..., m\}$:

$$A = t \wedge \bigwedge_{j=1}^{m} \quad \mathsf{X}^{\mathsf{j}}(x_j \vee \bar{x}_j),$$

where $\mathsf{X}^{\mathsf{j}}(\varphi')$ denotes the repeated application of X on a formula φ' for a total of j applications.

Note that if π^A is a trace such that $\pi^A \models_p A$, then we have $\pi^A_0 = \{t\}$ and $|\pi^A_j \cap \{x_j, \bar{x}_j\}| = 1$, for each $j \in \{1, ..., m\}$. Hence, π^A can be intuitively meant to encode a truth assignment for the variables in $\{x_1, ..., x_m\}$ such that x_j evaluates true in the assignment if, and only if, $\pi^A_j \models x_j$.

Equipped with the above notation, we can now move to detail the construction of $\xi_{fp}(\varphi)$. Formally, we define $\xi_{fp}(\varphi)$ as the formula $A \wedge \tau(\varphi)$, where τ is the function inductively built as follows over subformulas ψ of φ:

- If $\psi = x_j$ for some variable $x_j \in \mathcal{V}_\varphi$, then $\tau(\psi) = \mathsf{X}^{\mathsf{j}}(x_j)$;
- If $\psi = \neg x_j$ for some variable $x_j \in \mathcal{V}_\varphi$, then $\tau(\psi) = \mathsf{X}^{\mathsf{j}}(\bar{x}_j)$;
- If $\psi = (\psi' \wedge \psi'')$, then $\tau(\psi) = (\tau(\psi') \wedge \tau(\psi''))$;
- If $\psi = (\psi' \vee \psi'')$, then $\tau(\psi) = (\tau(\psi') \vee \tau(\psi''))$;
- If $\psi = \mathsf{X}(\psi')$, then $\tau(\psi) = \mathsf{X}^{\mathsf{m+1}}(A \wedge \tau(\psi'))$;
- If $\psi = \mathsf{X_w}(\psi')$, then $\tau(\psi) = \mathsf{X^m X_w}(A \wedge \tau(\psi'))$;
- If $\psi = \mathsf{F}(\psi')$, then $\tau(\psi) = \mathsf{F}(A \wedge \tau(\psi'))$;
- If $\psi = \mathsf{G}(\psi')$, then $\tau(\psi) = \mathsf{G}(\bigvee_{x \in \mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}} x \vee (A \wedge \tau(\psi')))$.
- If $\psi = (\psi' \,\mathsf{U}\, \psi'')$, then $\tau(\psi) = ((\bigvee_{x \in \mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}} x \vee (A \wedge \tau(\psi'))) \,\mathsf{U}\, (A \wedge \tau(\psi'')))$.
- If $\psi = (\psi' \,\mathsf{R}\, \psi'')$, then $\tau(\psi) = ((\bigvee_{x \in \mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}} x \vee (A \wedge \tau(\psi'))) \,\mathsf{R}\, (A \wedge \tau(\psi'')))$.

Note that $\xi_{fp}(\varphi)$ can be built in polynomial time. In particular, note that the size of $\xi_{fp}(\varphi)$ is at most quadratic in the size of φ. Then, to conclude we show that the two properties pointed out in the statement hold.

Proof of (1). Let π be a model of φ, and consider the trace π' with $len(\pi') = len(\pi) \times (m+1)$ defined as follows. For each time instant $i \in \{0, ..., len(\pi)\text{-}1\}$, if x_j is a variable in π_i (respectively, is not in π_i), then x_j (respectively, $\bar{x}_j$) is in $\pi'_{i \times (m+1)+j}$. No further variable is in $\pi_{i \times (m+1)+j}$. For each time instant $i \in \{0, ..., len(\pi)\text{-}1\}$, the variable t is in $\pi'_{i \times (m+1)}$. No further variable is in $\pi'_{i \times (m+1)}$. Note that, by construction, $\pi', i \times (m+1) \models A$ holds, for each time instant $i \in \{0, ..., len(\pi)\text{-}1\}$. We will show that π' is a PT-model of $\xi_{fp}(\varphi)$. To this end, consider any subformula ψ of φ that is required to hold (in the recursive definition of satisfiability for φ) at some time instant $i \in \{0, ..., len(\pi)\text{-}1\}$, i.e., $\pi, i \models \psi$. Then, we shall show that $\pi', i \times (m+1) \models_p A \wedge \tau(\psi)$ holds. This will entail that $\pi', 0 \models_p \xi_{fp}(\varphi)$. We proceed by structural induction.

Base case. If $\psi = x_j$ (respectively, $\psi = \neg x_j$), then x_j belongs (respectively, does not belong) to π_i. Hence, by construction, we have that $\pi', i \times (m+1) \models_p A \wedge \mathsf{X}^{\mathsf{j}}(x_j)$ (respectively, $\pi', i \times (m+1) \models_p A \wedge \mathsf{X}^{\mathsf{j}}(\bar{x}_j)$).

Inductive step. Assume that the property holds over each subformula of ψ. If $\psi = (\psi' \wedge \psi'')$ or $\psi = (\psi' \vee \psi'')$, then the property clearly holds on ψ, too.

If $\psi = \mathsf{X}(\psi')$, then $\pi, i \models \psi$ means that $\pi, i+1 \models \psi'$. By the inductive hypothesis, we know that $\pi', (i+1) \times (m+1) \models_p A \wedge \tau(\psi')$. Hence, we derive that $\pi', i \times (m+1) \models_p \mathsf{X}^{\mathsf{m+1}}(A \wedge \tau(\psi'))$.

If $\psi = \mathsf{X_w}(\psi')$, then $\pi, i \models \psi$ means that either $\pi, i+1 \models \psi'$ and $i < len(\pi)$-1, or $i = len(\pi)$-1. The first case is similar to that of X. As for the second one, we know that $len(\pi') = len(\pi) \times (m+1)$ and thus, if $i = len(\pi)$-1 then $i \times (m+1) + m$ is the last time instant of π' and $\pi', i \times (m+1) + m \models_p \mathsf{X_w}(A \wedge \tau(\psi'))$. Hence, we derive that $\pi', i \times (m+1) \models_p \mathsf{X}^{\mathsf{m}}\mathsf{X_w}(A \wedge \tau(\psi')))$.

If $\psi = \mathsf{F}(\psi')$, then $\pi, i \models \psi$ implies the existence of a time instant $i' \in \{i, ..., len(\pi)\text{-}1\}$ such that $\pi, i' \models \psi'$. By the inductive hypothesis, we know that $\pi', i' \times (m+1) \models_p A \wedge \tau(\psi')$. Hence, $\pi', i \times (m+1) \models_p \mathsf{F}(A \wedge \tau(\psi'))$.

If $\psi = \mathsf{G}(\psi')$, then $\pi, i \models \psi$ implies that for each time instant $i' \in \{i, ..., len(\pi)\text{-}1\}$, we have that $\pi, i' \models \psi'$. By the inductive hypothesis, we know that $\pi', i' \times (m+1) \models_p A \wedge \tau(\psi')$ holds. If $i'' \in \{i \times (m+1), ..., len(\pi')\text{-}1\}$ is a time instant for which there is no $i' \geq 0$ with $i'' = i' \times (m+1)$, then $\pi'_{i''}$ contains at least one variable in $\mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}$. So, $\pi', i'' \models_p \bigvee_{x \in \mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}} x$. And, we conclude that $\pi', i \times (m+1) \models_p \mathsf{G}(\bigvee_{x \in \mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}} x \vee (A \wedge \tau(\psi')))$.

If $\psi = (\psi' \mathbin{\mathsf{U}} \psi'')$, then $\pi, i \models \psi$ implies that there is a time instant $i' \geq i$ such that $\pi, i' \models \psi''$ and for each time instant $i'' \in \{i, ..., i'\text{-}1\}$ we have that $\pi, i'' \models \psi'$. By the inductive hypothesis, we know that $\pi', i' \times (m+1) \models_p A \wedge \tau(\psi'')$ and $\pi', i'' \times (m+1) \models_p A \wedge \tau(\psi')$, for each $i'' \in \{i, ..., i'\text{-}1\}$, hold. If $j \in \{i \times (m+1), ..., i' \times (m+1) - 1\}$ is a time instant for which there is no $j' \geq 0$ with $j = j' \times (m+1)$, then π'_j contains at least one variable in $\mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}$. So, $\pi', j \models_p \bigvee_{x \in \mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}} x$. Hence, $\pi', i \times (m+1) \models_p ((\bigvee_{x \in \mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}} x \vee (A \wedge \tau(\psi'))) \mathbin{\mathsf{U}} (A \wedge \tau(\psi'')))$.

If $\psi = (\psi' \mathbin{\mathsf{R}} \psi'')$, then $\pi, i \models \psi$ implies that either $\pi, i' \models \psi''$ for each $i' \geq i$, or there is a time instant $i''' \geq i$ such that $\pi, i''' \models \psi'$ and for each time instant $i'' \in \{i, ..., i'''\}$ we have that $\pi, i'' \models \psi''$. In the first case, by the inductive hypothesis, we know that $\pi', i' \times (m+1) \models_p A \wedge \tau(\psi'')$, for each $i' \geq i$, holds. If $j \geq i \times (m+1)$ is a time instant for which there is no $j' \geq 0$ with $j = j' \times (m+1)$, then π'_j contains at least one variable in $\mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}$. So, $\pi', j \models_p \bigvee_{x \in \mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}} x$. In the second case, by the inductive hypothesis, we know that $\pi', i''' \times (m+1) \models_p A \wedge \tau(\psi')$ and $\pi', i'' \times (m+1) \models_p A \wedge \tau(\psi'')$, for each $i'' \in \{i, ..., i'''\}$, hold. If $j \in \{i \times (m+1), ..., i''' \times (m+1) - 1\}$ is a time instant for which there is no $j' \geq 0$ with $j = j' \times (m+1)$, then π'_j contains at least one variable in $\mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}$. So, again, $\pi', j \models_p \bigvee_{x \in \mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}} x$. Hence, in both cases, we can conclude that $\pi', i \times (m+1) \models_p ((\bigvee_{x \in \mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}} x \vee (A \wedge \tau(\psi'))) \mathbin{\mathsf{R}} (A \wedge \tau(\psi'')))$ holds.

Proof of (2). Let π'' be a PT-model of $\xi_{fp}(\varphi)$. Let us define σ as the function that, given π'' and a time instant i such that $\pi'', i \models_p A$, returns the set $\sigma(\pi'', i)$ of propositional variables $\sigma(\pi'', i) = \{x_j \mid \pi'', i + j \models_p x_j\}$. Note that $\sigma(\pi'', i) \subseteq \mathcal{V}_\varphi$. In particular, we may view $\sigma(\pi'', i)$ as encoding a truth assignment for the variables in $\{x_1, ..., x_m\}$ such that x_j evaluates true in the assignment if, and only if, x_j belongs to $\sigma(\pi'', i)$. Let $c_0, ..., c_{\ell\text{-}1}$ be the set of all time instants of π'', in ascending order, such that $\pi'', c_i \models_p A$, for each $i \in \{0, ..., \ell\text{-}1\}$. Note that $c_0 = 0$, since $\xi_{fp}(\varphi) = A \wedge \tau(\varphi)$ holds, by construction. Then, consider the trace $\pi''' = \pi'''_0, ..., \pi'''_{\ell\text{-}1}$ such that $\pi'''_i = \sigma(\pi'', c_i)$, for each $i \in \{0, ..., \ell\text{-}1\}$. We will show that π''' is a model of φ. To this end, consider any subformula $\tau(\psi)$ of $\xi_{fp}(\varphi)$ that is required to hold (in the recursive definition of satisfiability for $\xi_{fp}(\varphi)$) at some time instant $c_i \in \{c_0, ..., c_{\ell\text{-}1}\}$. So, $\pi'', c_i \models_p A \wedge \tau(\psi)$. We shall show that $\pi''', i \models \psi$ holds. This will entail that $\pi''', 0 \models \varphi$, since we know that $\pi'', c_0 \models A \wedge \tau(\varphi)$. We proceed by structural induction on the subformulas ψ over which the function τ can be applied.

Base case. If $\psi = x_j$ (respectively, $\psi = \neg x_j$), then $\pi'', c_i \models_p A \wedge \mathsf{X}^{\mathsf{j}}(x_j)$ (respectively, $\pi'', c_i \models_p A \wedge \mathsf{X}^{\mathsf{j}}(\bar{x}_j)$) implies that x_j belongs (respectively, does not belong) to $\sigma(\pi'', c_i)$. Therefore, $\pi''', i \models x_j$ (respectively, $\pi''', i \models \neg x_j$).

Inductive step. Assume that the property holds over each subformula of ψ. If $\psi = (\psi' \wedge \psi'')$ or $\psi = (\psi' \vee \psi'')$, then the property clearly holds on ψ, too.

If $\psi = \mathsf{X}(\psi')$, then we have $\pi'', c_i \models_p A \wedge \mathsf{X}^{\mathsf{m+1}}(A \wedge \tau(\psi'))$. That is, $\pi'', c_{i+1} \models_p A \wedge \tau(\psi')$. By the inductive hypothesis, we know that $\pi''', i+1 \models \psi'$. Hence, $\pi''', i \models \mathsf{X}(\psi')$ holds.

If $\psi = \mathsf{X}_{\mathsf{w}}(\psi')$, then we have $\pi'', c_i \models_p A \wedge \mathsf{X}^{\mathsf{m}}\mathsf{X}_{\mathsf{w}}(A \wedge \tau(\psi'))$. We distinguish two cases. If $i < \ell - 1$, then we actually have that $\pi'', c_i \models_p A \wedge \mathsf{X}^{\mathsf{m+1}}(A \wedge \tau(\psi'))$. Hence, we apply the same line of reasoning used for X, in order to derive that $\pi''', i \models \mathsf{X}(\psi')$; so, $\pi''', i \models \mathsf{X}_{\mathsf{w}}(\psi')$. Consider now the case where $i = \ell - 1$. In this case, i is the last time instant of π''' and $\pi''', i \models \mathsf{X}_{\mathsf{w}}(\psi''')$, for whatever formula ψ'''. Hence, $\pi''', i \models \mathsf{X}_{\mathsf{w}}(\psi')$ holds.

If $\psi = \mathsf{F}(\psi')$, then $\pi'', c_i \models_p A \wedge \mathsf{F}(A \wedge \tau(\psi'))$ implies the existence of a time instant $c_{i'} \in \{c_i, ..., c_{\ell\text{-}1}\}$ such that $\pi'', c_{i'} \models_p A \wedge \tau(\psi')$. By the inductive hypothesis, we know that $\pi''', i' \models \psi'$. Hence, $\pi''', i \models \mathsf{F}(\psi')$.

If $\psi = \mathsf{G}(\psi')$, then $\pi'', c_i \models_p A \wedge \mathsf{G}(\bigvee_{x \in \mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}} x \vee (A \wedge \tau(\psi')))$. In particular, for each time instant $c_{i'} \in \{c_i, ..., c_{\ell\text{-}1}\}$, since we have that $\pi'', c_{i'} \models A$ holds (which implies that $\pi''_{c_{i'}} = \{t\}$), we have that $\pi'', c_{i'} \models A \wedge \tau(\psi')$ holds. By the inductive hypothesis, we know that $\pi''', i' \models \psi'$ holds, for each $i' \in \{i, ..., \ell\text{-}1\}$. Hence, $\pi''', i \models \mathsf{G}(\psi')$.

If $\psi = (\psi' \mathbin{\mathsf{U}} \psi'')$, then $\pi'', c_i \models_p A \wedge ((\bigvee_{x \in \mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}} x \vee (A \wedge \tau(\psi'))) \mathbin{\mathsf{U}} (A \wedge \tau(\psi'')))$ implies the existence of a time instant $c_{i''} \in \{c_i, ..., c_{\ell\text{-}1}\}$ such that $\pi'', c_{i''} \models A \wedge \tau(\psi'')$. Moreover, by using the same line of reasoning used for G, we can derive that, for each $c_{i'} \in \{c_i, ..., c_{i''\text{-}1}\}$, $\pi'', c_{i'} \models A \wedge \tau(\psi')$. By the inductive hypothesis, we know that $\pi''', i'' \models \psi''$ and $\pi''', i' \models \psi'$ hold, for i'' and each $i' \in \{i, ..., i''\text{-}1\}$. So, $\pi''', i \models (\psi' \mathbin{\mathsf{U}} \psi'')$.

If $\psi = (\psi' \mathbin{\mathsf{R}} \psi'')$, then $\pi'', c_i \models_p A \wedge ((\bigvee_{x \in \mathcal{V}_{\xi_{fp}(\varphi)} \setminus \{t\}} x \vee (A \wedge \tau(\psi'))) \mathbin{\mathsf{R}} (A \wedge \tau(\psi'')))$ implies either that $\pi'', c_{i'} \models A \wedge \tau(\psi'')$ for all $c_{i'} \in \{c_i, ..., c_{\ell-1}\}$, or the existence of a time instant $c_{i''} \in \{c_i, ..., c_{\ell\text{-}1}\}$ such that $\pi'', c_{i''} \models (A \wedge \tau(\psi'))$ and for each time instant $c_{i'''} \in \{c_i, ..., c_{i''}\}$ we have that $\pi, c_{i'''} \models (A \wedge \tau(\psi''))$. Moreover, by using the same line of reasoning used for G, in the first case by the inductive hypothesis we derive that $\pi''', i' \models \psi''$ holds, for $i' \in \{i, ..., \ell\text{-}1\}$. In the second case, by the inductive hypothesis, we derive that $\pi''', i'' \models \psi'$ and $\pi''', i''' \models \psi''$ hold, for each $i''' \in \{i, ..., i''\}$. In both cases, we conclude that $\pi''', i \models (\psi' \mathbin{\mathsf{R}} \psi'')$ holds. □

A simple consequence of the above result is that the complexity of reasoning over LTL$_p$ fragments including X is a lower bound on the complexity of reasoning about LTL$_f$ over the same fragments. Thus, the following characterization for the complexity of the whole logic LTL$_p$ derives after the results by De Giacomo and Vardi (2013).

Corollary 15 *Satisfiability of* LTL$_p$ *formulas is* PSPACE-*hard.*

In addition, lower bounds on the complexity of specific fragments $\langle T \cup \{\mathsf{X}\}\rangle$-LTL$_p$ will derive immediately, after we shall establish in Section 5 the corresponding results for LTL$_f$ restricted on the same syntactic fragment defined by T. With this respect, it is relevant to note that the encoding discussed in the proof of the above result can be adapted to the case where each $\langle T\rangle$-LTL$_f$ formula is translated into a negation-free $\langle T \cup \{\mathsf{X}_\mathsf{w}, \mathsf{F}\}\rangle$-LTL$_p$ formula, by using, in place of the formula A, the formula $A' \wedge \mathsf{F}(x_m \vee \bar{x}_m)$ where A' is obtained from A by replacing each occurrence of X with X_w. Indeed, in any PT-model, $A' \wedge \mathsf{F}(x_m \vee \bar{x}_m)$ guarantees the existence of $m + 1$ time instants following the one where A' holds. This suffices to preserve the properties discussed in the proof of Theorem 14, so that the following derives by its inspection.

Theorem 16 *A polynomial-time satisfiability preserving translation exists which maps* $\langle T\rangle$-LTL$_f$ *formulas into negation-free* $\langle T \cup \{\mathsf{X}_\mathsf{w}, \mathsf{F}\}\rangle$-LTL$_p$ *formulas.*

We leave this section by pointing out one further useful relationship between LTL$_p$ and LTL$_f$. In particular, we consider formulas built with the until temporal operator.

Theorem 17 *A polynomial-time (satisfiability preserving) translation* ξ_{U} *exists that, given a* $\langle \mathsf{U}\rangle$-LTL$_f$ *formula* φ*, returns a negation-free* $\langle \mathsf{U}\rangle$-LTL$_p$ *formula* $\xi_{\mathsf{U}}(\varphi)$ *enjoying the following properties:*

(1) *Assume that* π *is a model of* φ*. Then, a* PT*-model* π' *of* $\xi_{\mathsf{U}}(\varphi)$ *can be built from* π *in polynomial time such that* $len(\pi') = 2 \times len(\pi) \times (|\mathcal{V}_\varphi| + 1) - 1$*;*

(2) *Assume that* π'' *is a* PT*-model of* $\xi_{\mathsf{U}}(\varphi)$*. Then, a model* π''' *of* φ *can be built from* π'' *in polynomial time such that* $len(\pi''') \leq len(\pi'')$*.*

Proof. According to Theorem 14, whatever $\langle\{\mathsf{U}\}\rangle$-LTL$_f$ formula φ can be rewritten into an equivalent negation-free $\langle\{\mathsf{X}, \mathsf{U}\}\rangle$-LTL$_p$ formula $\xi_{fp}(\varphi)$. The line of the proof is then to show that, over such formulas, the X operator can be simulated by using the U operator. To this end, if ϕ is any given negation-free $\langle\{\mathsf{X}, \mathsf{U}\}\rangle$-LTL$_p$ formula, then we first build a negation-free $\langle\{\mathsf{U}\}\rangle$-LTL$_p$ formula $\delta(\phi)$ over the set of variables $\mathcal{V}_{\delta(\phi)} = \mathcal{V}_\phi \cup \{\bullet\}$, where $\bullet$ is a special variable that will keep track of the time instants required by the X operators in the models of ϕ. Formally, δ is inductively defined as follows over subformulas ψ of ϕ:

- If $\psi = x_j$ for some variable $x_j \in \mathcal{V}_\phi$, then $\delta(\psi) = x_j$;
- If $\psi = (\psi' \wedge \psi'')$, then $\delta(\psi) = (\delta(\psi') \wedge \delta(\psi''))$;
- If $\psi = (\psi' \vee \psi'')$, then $\delta(\psi) = (\delta(\psi') \vee \delta(\psi''))$;
- If $\psi = \mathsf{X}(\psi')$, then $\delta(\psi) = \bigvee_{x_k \in \mathcal{V}_\phi} (x_k \wedge (x_k \;\mathsf{U}\; (\bullet \wedge (\bullet \;\mathsf{U}\; \delta(\psi')))))$;
- If $\psi = (\psi' \;\mathsf{U}\; \psi'')$, then $\delta(\psi) = (\delta(\psi') \vee \bullet) \;\mathsf{U}\; (\delta(\psi''))$.

Eventually, we just define $\xi_{\mathsf{U}}(\varphi)$ as the result of the application of δ over $\xi_{fp}(\varphi)$, i.e., $\xi_{\mathsf{U}}(\varphi) = \delta(\xi_{fp}(\varphi))$. Now, we show that the two properties pointed out in the statement hold.

Proof of (1). We prove a general result related to the application of δ to negation-free $\langle\{\mathsf{X}, \mathsf{U}\}\rangle$-LTL$_p$ formulas. The required property will then follow by specializing the result to $\xi_{fp}(\varphi)$.

Let ϕ be a satisfiable negation-free $\langle\{\mathsf{X}, \mathsf{U}\}\rangle$-LTL$_p$ formula and let $\bar{\pi}$ be a PT-model of it. Consider the trace π' with $len(\pi') = 2 \times len(\bar{\pi}) - 1$ defined as follows: $\pi'_{2\times i} = \bar{\pi}_i$ and $\pi'_{2\times i+1} = \{\bullet\}$, for each $i \in \{0, ..., len(\bar{\pi})\text{-}2\}$; $\pi'_{2\times(len(\bar{\pi})\text{-}1)} = \bar{\pi}_{len(\bar{\pi})\text{-}1}$. We claim that π' is a PT-model of $\delta(\phi)$.

To prove this claim, consider any subformula ψ of ϕ that is required to hold (in the recursive definition of satisfiability) at some time instant $i \in \{0, ..., len(\bar{\pi})\text{-}1\}$, i.e., $\bar{\pi}, i \models_p \psi$. We shall show that $\pi', 2 \times i \models_p \delta(\psi)$ holds. This will entail that $\pi', 0 \models_p \delta(\phi)$. We proceed by structural induction on the subformulas ψ.

Base case. If $\psi = x_j$, then $\bar{\pi}_i = \{x_j\}$. Hence, by construction, we have that $\pi', 2 \times i \models_p x_j$.

Inductive step. Assume that the property holds over each subformula of ψ. If $\psi = (\psi' \wedge \psi'')$ or $\psi = (\psi' \vee \psi'')$, then the property clearly holds on ψ, too.

If $\psi = \mathsf{X}(\psi')$, then $\bar{\pi}, i \models_p \psi$ means that $\bar{\pi}, i+1 \models_p \psi'$. Then, by inductive hypothesis, we know that $\pi', 2 \times (i+1) \models_p \delta(\psi')$. By construction, we have that $\pi', 2 \times i + 1 \models \bullet$. Thus, we have that $\pi', 2 \times i + 1 \models_p \bullet \wedge (\bullet \,\mathsf{U}\, \delta(\psi'))$. Furthermore, $\bar{\pi}_i = \{x_k\}$ holds, for some variable $x_k \in \mathcal{V}_\phi$. Then, we have that $\pi'_{2\times i} = \{x_k\}$ and we derive $\pi', 2 \times i \models_p x_k \wedge (x_k \,\mathsf{U}\, (\bullet \wedge (\bullet \,\mathsf{U}\, \delta(\psi'))))$.

If $\psi = (\psi' \,\mathsf{U}\, \psi'')$, then $\bar{\pi}, i \models_p \psi$ implies that there is a time instant $i' \geq i$ such that $\bar{\pi}, i' \models_p \psi''$ and, for each time instant $i'' \in \{i, ..., i'\text{-}1\}$, we have that $\bar{\pi}, i'' \models_p \psi'$. By the inductive hypothesis, we know that $\pi', 2 \times i' \models_p \delta(\psi'')$ and $\pi', 2 \times i'' \models_p \delta(\psi')$, for each $i'' \in \{i, ..., i'\text{-}1\}$. By construction, we also know that $\pi', 2 \times i'' + 1 \models_p \bullet$, for each $i'' \in \{i, ..., i'\text{-}1\}$. Thus, we conclude that $\pi', 2 \times i \models_p (\delta(\psi') \vee \bullet) \,\mathsf{U}\, (\delta(\psi''))$ holds.

As we have anticipated, the first property in the statement follows by specializing the above result to the case where $\phi = \xi_{fp}(\varphi)$ and by recalling that if π is a model of φ, then a PT-model $\bar{\pi}$ of $\xi_{fp}(\varphi)$ can be built from π in polynomial time with $len(\bar{\pi}) = len(\pi) \times (|\mathcal{V}_\varphi| + 1)$ (cf. Theorem 14).

Proof of (2). Assume that π'' is a PT-model of $\xi_{\mathsf{U}}(\varphi)$. Based on π'', we shall build a model π''' of φ such that $len(\pi''') \leq len(\pi'')$, by using intermediate transformations via traces π^0 and $\hat{\pi}$.

From π'' to π^0. For each $i \in \{0, ..., len(\pi'') - 1\}$, consider the trace π^i obtained from π'' by removing all consecutive duplicate states starting from the state i. We shall show that π^0 is a PT-model of $\xi_{\mathsf{U}}(\varphi)$. To this end, for every subformula $\delta(\psi)$ of $\xi_{\mathsf{U}}(\varphi)$ and time instant $i \in \{0, ..., len(\pi'')-1\}$, we claim that $\pi'', i \models_p \delta(\psi)$ implies $\pi^i, i \models_p \delta(\psi)$. We proceed by induction on i.

Base case. If $i = len(\pi'') - 1$, then we have $i = len(\pi^i) - 1$ and $\pi^i = \pi''$. Hence, we trivially have that $\pi^i, i \models_p \delta(\psi)$ if, and only if, $\pi'', i \models_p \delta(\psi)$, for every formula $\delta(\psi)$.

Inductive step. Assume that the property holds on all $j > i$. We proceed by a nested induction over the subformulas $\delta(\psi)$, in order to show that the property holds on i, too. In the base case where $\delta(\psi) = x_k$, for some variable x_k, after noticing that $\pi^i_i = \pi''_i$ holds by construction of π^i, we derive that $\pi'', i \models_p \delta(\psi)$ if, and only if, $\pi^i, i \models_p \delta(\psi)$. Moving to the inductive step, note first that the property trivially holds on $\delta(\psi)$, if $\delta(\psi) = (\delta(\psi') \wedge \delta(\psi''))$ or $\delta(\psi) = (\delta(\psi') \vee \delta(\psi''))$.

Assume now that $\pi'', i \models_p \delta(\psi)$, where $\delta(\psi) = \bigvee_{x_k} (x_k \wedge (x_k \,\mathsf{U}\, (\bullet \wedge (\bullet \,\mathsf{U}\, \delta(\psi')))))$. So, there is a variable x_k and two time instants i', i'' such that: $\pi''_j = \{x_k\}$, for each $i \leq j < i'$; $\pi''_j = \{\bullet\}$, for each $i' \leq j < i''$; and $\pi'', i'' \models_p \delta(\psi')$. By construction of π^i, this entails that $\pi^i_i = \{x_k\}$ and $\pi^i_{i+1} = \{\bullet\}$. Moreover, by inductive hypothesis, we know that $\pi'', i'' \models_p \delta(\psi')$ implies $\pi^{i''}, i'' \models_p \delta(\psi')$. Now, let π^* be the trace derived from $\pi^{i''}$ by considering all time instants from i'' onwards. We have two possibilities: either π^* coincides with the trace derived from π^i by considering all time instants from $i+1$ onwards, or it coincides with the trace derived from π^i by considering all time instants from $i+2$ onwards (depending on whether $\pi''_{i''} = \{\bullet\}$ or $\pi''_{i''} \neq \{\bullet\}$). Therefore, in the former case we have $\pi^i, i+1 \models_p \delta(\psi')$, whereas in the latter we have $\pi^i, i+2 \models_p \delta(\psi')$. In both cases, we conclude that $\pi^i, i \models_p \delta(\psi)$.

Finally, consider the case where $\delta(\psi) = ((\delta(\psi') \vee \bullet) \,\mathsf{U}\, \delta(\psi''))$. Note that $\pi'', i \models_p \delta(\psi)$ implies that either $\pi'', i \models_p \delta(\psi'')$, or we have $\pi'', i \models_p (\delta(\psi') \vee \bullet)$ and $\pi'', i+1 \models_p \delta(\psi)$. In the former case, we conclude that $\pi^i, i \models_p \delta(\psi'')$, because of the inductive hypothesis; hence, $\pi^i, i \models_p \delta(\psi)$. In the latter case, note that $\pi^{i+1}, i+1 \models_p \delta(\psi)$ holds, again, by the inductive hypothesis. Define π^* as the trace derived from π^{i+1} by considering all time instants from $i+1$ onwards. If π^* coincides with the trace derived from π^i by considering all time instants from i onwards, then we immediately conclude that $\pi^i, i \models_p \delta(\psi)$. Otherwise, we have that π^* coincides with the trace derived from π^i by considering all time instants from $i+1$ onwards, so that $\pi^i_{i+1} = \pi^{i+1}_{i+1}$. Now, if $\pi''_i = \{\bullet\}$, then the fact that $\pi^{i+1}, i+1 \models_p \delta(\psi)$ implies that $\pi^i, i \models_p \delta(\psi)$. Eventually, if $\pi'', i \models_p \delta(\psi')$, then we get by the inductive hypothesis that $\pi^i, i \models_p \delta(\psi')$; hence, we derive again $\pi^i, i \models_p \delta(\psi)$.

From π^0 *to* $\hat{\pi}$. Let $c_0, ..., c_{\ell\text{-}1}$ be the set of all time instants of π^0, in ascending order, such that $\pi^0_{c_i} \neq \{\bullet\}$, for each $i \in \{0, ..., \ell\text{-}1\}$. Consider the trace $\hat{\pi} = \hat{\pi}_0, ..., \hat{\pi}_{\ell\text{-}1}$ such that $\hat{\pi}_i = \pi^0_{c_i}$, for each $i \in \{0, ..., \ell\text{-}1\}$. Consider any subformula $\delta(\psi)$ of $\xi_{\mathsf{U}}(\varphi)$ that is required to hold (in the recursive definition of satisfiability for $\xi_{\mathsf{U}}(\varphi)$) at some time instant $c_i \in \{c_0, ..., c_{\ell\text{-}1}\}$. We shall show that $\hat{\pi}, i \models_p \psi$ holds. This will entail that $\hat{\pi}, 0 \models_p \xi_{fp}(\varphi)$. We proceed by structural induction on the subformulas ψ over which the function δ can be applied.

Base case. If $\psi = x_k$ holds for some variable x_k (so that $\delta(\psi) = \psi$), then $\pi^0, c_i \models_p \delta(\psi)$ implies that $\pi^0_{c_i} = \hat{\pi}_i = \{x_k\}$. Therefore, we clearly have that $\hat{\pi}, i \models x_k$.

Inductive step. Assume that the property holds over each subformula of ψ. If $\psi = (\psi' \wedge \psi'')$ or $\psi = (\psi' \vee \psi'')$, then the property clearly holds on ψ.

If $\psi = \mathsf{X}(\psi')$, then we have $\pi^0, c_i \models_p \bigvee_{x_k} (x_k \wedge (x_k \,\mathsf{U}\, (\bullet \wedge (\bullet \,\mathsf{U}\, \delta(\psi')))))$. Observe that, since π^0 does not contain consecutive time instants in which the same variable evaluates true, it must be the case that $\pi^0, c_i + 1 \models_p \bullet$ and $\pi^0, c_i + 2 \models_p \delta(\psi')$ and $\pi^0_{c_i+2} \neq \{\bullet\}$. Hence, $c_{i+1} = c_i + 2$ and we have that $\pi^0, c_{i+1} \models_p \delta(\psi')$. By the inductive hypothesis, we know that $\hat{\pi}, i+1 \models \psi'$ and we can conclude that $\hat{\pi}, i \models \mathsf{X}(\psi')$.

If $\psi = (\psi' \,\mathsf{U}\, \psi'')$, then we have that $\pi^0, c_i \models_p (\delta(\psi') \vee \bullet) \,\mathsf{U}\, (\delta(\psi''))$. We can consider two cases. The first case happens if $\pi^0, c_i \models_p \delta(\psi'')$ holds. By the inductive hypothesis, we have that $\hat{\pi}, i \models \psi''$ and, thus, we can conclude that $\hat{\pi}, i \models \psi$. In the second case, there exists a time instant $i'' > c_i$ such that $\pi^0, i'' \models \delta(\psi'')$ holds and, for each time instant $i' \in \{c_i, ..., i''\text{-}1\}$, we have that either $\pi^0, i' \models \delta(\psi')$ or $\pi^0, i' \models \bullet$. Thus, we have that $\pi^0, c_j \models \delta(\psi')$ holds (and by the inductive hypothesis $\hat{\pi}, j \models \psi'$ holds) in all the time instants $c_j \in \{c_0, ..., c_{l-1}\}$ with $c_i \leq c_j < i''$, since by construction $\pi^0_{c_j} \neq \{\bullet\}$. To conclude, note that, because of the definition of the function δ, we can assume, w.l.o.g., that $\pi^0_{i''} \neq \{\bullet\}$. Hence, i'' is a time instant, say c_h, in $\{c_0, ..., c_{l-1}\}$. By the inductive hypothesis, it follows that $\hat{\pi}, h \models \psi''$ and we conclude that $\hat{\pi}, i \models (\psi' \,\mathsf{U}\, \psi'')$.

From $\hat{\pi}$ *to* π'''. Recall that $\hat{\pi}$ is a model of $\xi_{fp}(\varphi)$ and note that $len(\hat{\pi}) \leq len(\pi'')$. The result then follows because, according to Theorem 14, we can build in polynomial time from $\hat{\pi}$ a model π''' of φ such that $len(\pi''') \leq len(\hat{\pi})$. $\square$

4. Linear-Length Model Property

In this section, we analyze all fragments of LTL$_f$ and LTL$_p$ obtained by constraining the allowed temporal operators with the aim of identifying those formulas φ enjoying the *linear-length model*

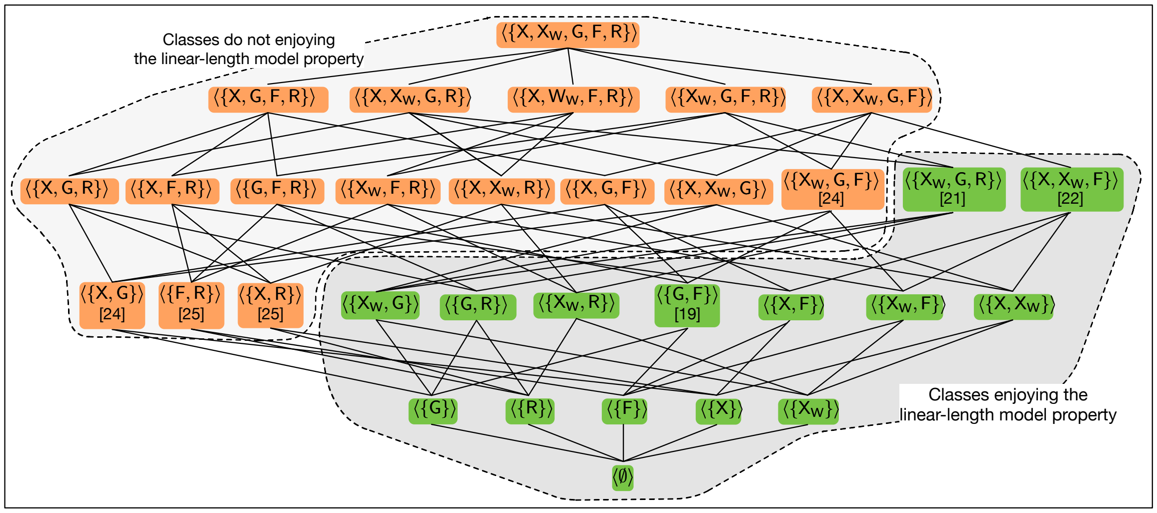


Figure 3: Summary of results in Section 4 for $\langle T\rangle$-LTL$_f$ formulas. The figure evidences all sets of operators $T \subseteq \{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{R}\}$ for which the corresponding classes of formulas enjoy the linear-length model property—the classes corresponding to all supersets $T \supseteq \{\mathsf{U}\}$ do not enjoy the linear-length model property. Precisely the same results hold for $\langle T\rangle$-LTL$_p$ and negation-free $\langle T\rangle$-LTL$_p$ formulas.

property, i.e., such that satisfiability can be always witnessed by models π whose length, $len(\pi)$, is linear in the size of the formula $||\varphi||$. It will emerge that, for each $T \subseteq \{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}, \mathsf{R}\}$, either $\langle T\rangle$-LTL$_f$ (respectively, $\langle T\rangle$-LTL$_p$) enjoys the linear-length model property, or it contains satisfiable formulas φ having no model (respectively, PT-model) whose length is polynomial with respect to $||\varphi||$.

In particular, we shall observe that the linear-length model property holds on all classes of negation-free LTL$_f$ formulas. By contrast, for arbitrary LTL$_f$ formulas (i.e., if negation is allowed), a more complex picture will emerge. Results for such formulas will incidentally coincide with the results for LTL$_p$ formulas, but with the absence of negation being immaterial in this latter case (see Lemma 10). A summary of these results is reported in Figure 3, in terms of the Hasse diagram built over the subsets of $\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{R}\}$. The sets containing U are not reported in the figure since even the class of formulas where only U is allowed does not enjoy the linear-length model property.

4.1 Formulas Enjoying the Linear-Length Model Property

We start the illustration of the results by considering LTL$_f$ formulas defined without any restriction on the temporal operators that are allowed, but forbidding negation. The result is shown by standard structural induction, and the proof is reported for the sake of completeness, only.

Theorem 18 *Every satisfiable negation-free* $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}, \mathsf{R}\}\rangle$-LTL$_f$ *formula* φ *has a model* π *such that* $len(\pi) \leq th(\varphi, \mathsf{X}) + th(\varphi, \mathsf{X_w}) + 1$.

Proof. Let φ be a satisfiable negation-free formula in $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}\}\rangle$-LTL$_f$. We shall show, by structural induction, that φ admits a model π, said *φ-canonical*, such that $len(\pi) \leq th(\varphi, \mathsf{X}) +$

$th(\varphi, \mathsf{X_w}) + 1$. In particular, such canonical models π have the additional property that $\pi_i = \mathcal{V}_\varphi$, for each $i \in \{0, ..., len(\pi)\text{-}1\}$. Let ψ be any subformula of φ.

Base case. If $\psi = x$, then ψ is satisfiable by the trace π having length 1 and such that $\pi_0 = \mathcal{V}_\varphi$. Clearly, π is a ψ-canonical model.

Inductive step. Assume that, for each subformula $\bar{\varphi}$ of ψ, if $\bar{\varphi}$ is satisfiable, then $\bar{\varphi}$ admits a $\bar{\varphi}$-canonical model $\bar{\pi}$, i.e., such that $len(\bar{\pi}) \leq th(\bar{\varphi}, \mathsf{X}) + th(\bar{\varphi}, \mathsf{X_w}) + 1$ and $\bar{\pi}_i = \mathcal{V}_\varphi$, for each $i \in \{0, ..., len(\bar{\pi})\text{-}1\}$. We show that if ψ is satisfiable, then it admits a ψ-canonical model. Indeed,

- If $\psi = (\varphi' \vee \varphi'')$, then one of the two subformulas, say φ', is satisfiable. So, we can take a φ'-canonical model π', and observe that $\pi' \models \psi$. Since $len(\pi') \leq th(\varphi', \mathsf{X}) + th(\varphi', \mathsf{X_w}) + 1 \leq th(\psi, \mathsf{X}) + th(\psi, \mathsf{X_w}) + 1$, we conclude that π' is ψ-canonical.

- If $\psi = (\varphi' \wedge \varphi'')$, then both φ' and φ'' are satisfiable and, by inductive hypothesis, we can take a φ'-canonical (respectively, φ''-canonical) model π' (respectively, π''). W.l.o.g., assume $len(\pi') \leq len(\pi'')$. Then it can be checked that $\pi'' \models \psi$, and it directly follows that π'' is ψ-canonical.

- If $\psi = \mathsf{F}(\varphi')$, then φ' is satisfiable and it admits a φ'-canonical model π'. Indeed, π' is trivially a ψ-canonical model, too.

- If $\psi = \mathsf{G}(\varphi')$, then φ' is satisfiable. Moreover, φ' must actually admit a φ'-canonical model π' having length 1, for otherwise ψ cannot be satisfied by a finite trace. Then $\pi' \models \psi$ and π' is trivially ψ-canonical.

- If $\psi = \mathsf{X}(\varphi')$ or $\psi = \mathsf{X_w}(\varphi')$, then φ' is satisfiable and hence it admits a φ'-canonical model π'. Then ψ can be satisfied by a model π such that $len(\pi) = len(\pi') + 1 \leq th(\varphi', \mathsf{X}) + th(\varphi', \mathsf{X_w}) + 2 \leq th(\psi, \mathsf{X}) + th(\psi, \mathsf{X_w}) + 1$. In particular, $\pi_0 = \mathcal{V}_\varphi$ and $\pi_i = \pi'_{i-1}$, for each $i \in \{0, ..., len(\pi')\text{-}1\}$. Hence, π is ψ-canonical, too.

- If $\psi = (\varphi' \,\mathsf{U}\, \varphi'')$, then φ'' is satisfiable by a φ''-canonical model π''. It can be checked that $\pi'' \models \psi$ and that π'' is ψ-canonical.

- If $\psi = (\varphi' \,\mathsf{R}\, \varphi'')$, then we can have two cases. If φ'' is satisfiable by a φ''-canonical model π'' having length 1, then it can be checked that $\pi'' \models \psi$ and that π'' is ψ-canonical. Otherwise, both φ' and φ'' are satisfiable and, by inductive hypothesis, we can take a φ'-canonical (respectively, φ''-canonical) model π' (respectively, π''). W.l.o.g., assume $len(\pi') \leq len(\pi'')$. Then, it can be checked that $\pi'' \models (\varphi' \wedge \varphi'')$ and, thus $\pi'' \models \psi$. Eventually, it directly follows that π'' is ψ-canonical.

□

By considering LTL$_f$ formulas with negation, a finer-grained analysis is needed. First, we analyze formulas where the next (X) operator is forbidden. The idea is to reuse known bounds for LTL formulas (Sistla & Clarke, 1985), by taking care of subtle semantic issues that arise when moving from standard LTL to LTL over finite traces (De Giacomo & Vardi, 2013; De Giacomo, De Masellis, & Montali, 2014b). The result holds on LTL$_p$ formulas, too.

Theorem 19 *Every satisfiable $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$-LTL$_f$ (respectively, $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$-LTL$_p$) formula φ has a model (respectively,* PT*-model) π such that $len(\pi) \leq ts(\varphi, \mathsf{F}) + ts(\varphi, \mathsf{G}) + 6$.*

Proof. Let φ be a satisfiable formula in $\langle\{\mathsf{G},\mathsf{F}\}\rangle$-LTL$_f$. Let us consider the transformation rules (for arbitrary LTL$_f$ formulas) discussed by De Giacomo et al. (2014b), which based on φ produce an LTL formula $\bar{\varphi}$ over $\mathcal{V}_\varphi \cup \{end\}$, where end is a fresh Boolean variable. In particular, the properties of the transformation are stated below—and follow by inspecting and specializing the results of De Giacomo et al. (see, also, De Giacomo & Vardi, 2013) to formulas containing only F and G as temporal operators.

Claim 20 (cf. De Giacomo et al., 2014b) *For each $\langle\{\mathsf{G},\mathsf{F}\}\rangle$-LTL$_f$ formula φ, there is an* LTL *formula $\bar{\varphi}$, where negation is not necessarily atomic, such that:*

(1) *there is no temporal operator in $\bar{\varphi}$ but* F*;*

(2) *if π is a finite model of φ, then $\omega(\pi)$ is a model of $\bar{\varphi}$, where $\omega(\pi)$ is the infinite trace obtained from π by appending the state $\{end\}$ indefinitely;*

(3) *if π' is a model of $\bar{\varphi}$, then there is a finite model π of φ such that $\pi' = \omega(\pi)$; and*

(4) $ts(\bar{\varphi},\mathsf{F}) = ts(\varphi,\mathsf{F}) + ts(\varphi,\mathsf{G}) + 4$.

Proof. Consider the formula $\tilde{\varphi} = \Phi_1 \wedge \Phi_2 \wedge \Phi_3 \wedge \Phi_4 \wedge t(\varphi)$ such that: $\Phi_1 = \neg end$, meaning that end is false at the initial time instant; $\Phi_2 = \mathsf{F}(end)$, meaning that end will eventually hold at some time instant; $\Phi_3 = \mathsf{G}(\neg end \vee \mathsf{G}(end))$, meaning that once end becomes true, then it always remains true; $\Phi_4 = \mathsf{G}(\neg end \vee \bigwedge_{x\in\mathcal{V}_\varphi} \neg x)$, meaning that all variables in $\mathcal{V}_\varphi$ must be false as soon as end is true. Define $t(\varphi)$ recursively as follows: $t(x) = x$, for each $x \in \mathcal{V}_\varphi$; $t(\neg\varphi) = \neg t(\varphi)$; $t(\varphi_1 \wedge \varphi_2) = t(\varphi_1) \wedge t(\varphi_2)$; $t(\varphi_1 \vee \varphi_2) = t(\varphi_1) \vee t(\varphi_2)$; $t(\mathsf{F}(\varphi)) = \mathsf{F}(t(\varphi) \wedge \neg end)$; and $t(\mathsf{G}(\varphi)) = \mathsf{G}(t(\varphi) \vee end)$. Now, observe that, for each formula φ', $\mathsf{G}(\varphi')$=$\neg\mathsf{F}(\neg\varphi')$. That is, $\tilde{\varphi}$ can be rewritten in terms of F and arbitrary negation, leading to an equivalent formula $\bar{\varphi}$; so (1) holds. Then the fact that (2) and (3) hold on $\tilde{\varphi}$, hence on $\bar{\varphi}$, has been observed by De Giacomo et al. (2014b) (see, also, De Giacomo & Vardi, 2013). Eventually, $\sum_{i=1}^{4} ts(\Phi_i,\mathsf{F}) + \sum_{i=1}^{4} ts(\Phi_i,\mathsf{G}) = 4$, and we get $ts(\bar{\varphi},\mathsf{F}) = ts(t(\varphi),\mathsf{F}) + ts(t(\varphi),\mathsf{G}) + 4 = ts(\varphi,\mathsf{F}) + ts(\varphi,\mathsf{G}) + 4$. Hence, (4) holds. $\diamond$

Consider now the formula $\bar{\varphi}$ built as discussed above. By Theorem 3.4 in the the work of Sistla and Clarke (1985)—which holds on LTL formulas defined over F only—and given that all models of $\bar{\varphi}$ have the form $\omega(\pi)$ because of (3), it is guaranteed the existence of a model $\omega(\pi^*)$ of $\bar{\varphi}$ such that $len(\pi^*) \leq ts(\bar{\varphi},\mathsf{F}) + 1$. In addition, we know that $\pi^* \models \varphi$ and we get $len(\pi^*) \leq ts(\bar{\varphi},\mathsf{F}) + 1 = ts(\varphi,\mathsf{F}) + ts(\varphi,\mathsf{G}) + 4 + 1 = ts(\varphi,\mathsf{F}) + ts(\varphi,\mathsf{G}) + 5$.

The above result can be easily extended to deal with PT-models (by using the approach discussed in the introduction to Section 3.1). Assume that φ is satisfiable by a PT-model. Define $\hat{\varphi}$ as the following formula: $\varphi \wedge \mathsf{G}(\bigvee_{y\in\mathcal{V}_\varphi}(y \wedge \bigwedge_{y'\in\mathcal{V}_\varphi\setminus\{y\}} \neg y'))$. Note that $\hat{\varphi}$ is satisfiable and that, if π is a model of $\hat{\varphi}$, then it is a PT-model of φ. Now, we have already shown that $\hat{\varphi}$ has a model π^* such that $len(\pi^*) \leq ts(\hat{\varphi},\mathsf{F}) + ts(\hat{\varphi},\mathsf{G}) + 5$. Since $ts(\hat{\varphi},\mathsf{F}) = ts(\varphi,\mathsf{F})$ and $ts(\hat{\varphi},\mathsf{G}) = ts(\varphi,\mathsf{G}) + 1$, we have $len(\pi^*) \leq ts(\varphi,\mathsf{F}) + ts(\varphi,\mathsf{G}) + 6$. $\square$

Note that the above result provides a bound of $ts(\varphi,\mathsf{G}) + 6$ for a formula φ containing only G as temporal operator. However, in this special case, it can be checked that if π is a model of φ, then $\pi, i \models \varphi$ actually holds, for each time instant $i \in \{0, ..., len(\pi)\text{-}1\}$. So, the trace consisting of the initial state π_0 only is a model, too. It is also easy to see that this property still holds in the presence of $\mathsf{X_w}$ and R as further temporal operators. In particular, note that if φ is a formula having the form

$(\varphi' \mathsf{R}\, \varphi'')$ and π is a finite model of φ, then $\pi, 0 \models \varphi''$ necessarily holds. Thus, again, the trace consisting of the initial state π_0 only is a finite model of φ.

Fact 21 *Every satisfiable $\langle\{\mathsf{X_w}, \mathsf{G}, \mathsf{R}\}\rangle$-LTL$_f$ (respectively, $\langle\{\mathsf{X_w}, \mathsf{G}, \mathsf{R}\}\rangle$-LTL$_p$) formula φ has a model (respectively,* PT-*model) π such that $len(\pi) = 1$.*

We now discuss the case where the temporal operators G and U are forbidden. Note that we cannot reuse the approach discussed in the proof of Theorem 19, because the transformation rules introduced by De Giacomo et al. (2014b) and De Giacomo and Vardi (2013) do require the use of G.

Theorem 22 *Every satisfiable $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{F}\}\rangle$-LTL$_f$ (respectively, $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{F}\}\rangle$-LTL$_p$) formula φ has a model (respectively,* PT-*model) π such that $len(\pi) \leq ts(\varphi, \mathsf{X}) + ts(\varphi, \mathsf{F}) + ts(\varphi, \mathsf{X_w}) + 1$.*

Proof. Let φ be a satisfiable formula in $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{F}\}\rangle$-LTL$_f$ (respectively, $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{F}\}\rangle$-LTL$_p$). Let π be a model (respectively, PT-model) of φ. Let φ'' be a subformula of φ and let i be a time instant such that $\pi, i \models \varphi''$ (respectively, $\pi, i \models_p \varphi''$). We recursively associate with φ'' and i a set of time instants $\mathcal{C}_{\varphi'',i}$ as follows. If $\varphi'' = \mathsf{F}(\varphi')$, then we define $\mathcal{C}_{\varphi'',i}$ as the set $\{j\} \cup \mathcal{C}_{\varphi',j}$, where j is the maximum time instant with $i \leq j < len(\pi)$ and such that $\pi, j \models \varphi'$. If $\varphi'' = \mathsf{X}(\varphi')$, then we define $\mathcal{C}_{\varphi'',i}$ as the set $\{i+1\} \cup \mathcal{C}_{\varphi',i+1}$. If $\varphi'' = \mathsf{X_w}(\varphi')$ and $i < len(\pi)$-1, then we define $\mathcal{C}_{\varphi'',i}$ as the set $\{i+1\} \cup \mathcal{C}_{\varphi',i+1}$. Finally, in all other cases, we define $\mathcal{C}_{\varphi'',i} = \{\}$.

Define now $\mathcal{C} = \{0\} \cup \mathcal{C}_{\varphi,0}$ as the set of *critical* time instants of φ. According to the recursive definition of satisfiability reported in Section 2, only time instants in $\mathcal{C}$ are required to exist in the model for having $\pi, 0 \models \varphi$ (respectively, $\pi, 0 \models_p \varphi$). Therefore, the finite trace $\bar{\pi}$ derived from π by removing each state associated with a time instant that is not critical still satisfies φ. The result follows since the length of $\bar{\pi}$ is at most $ts(\varphi, \mathsf{X}) + ts(\varphi, \mathsf{F}) + ts(\varphi, \mathsf{X_w}) + 1$. □

4.2 Models of Exponential Length

So far, we have singled out a number of classes of formulas enjoying the linear-length model property. Now, we provide a general upper bound on the length of the models.

Theorem 23 *For each $T \subseteq \{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}, \mathsf{R}\}$, every satisfiable $\langle T\rangle$-LTL$_f$ (respectively, $\langle T\rangle$-LTL$_p$) formula φ has a model (respectively,* PT-*model) π such that $len(\pi) \leq 2^{O(||\varphi||)}$.*

Proof. Consider again the proof of Claim 20, showing that every $\langle T\rangle$-LTL$_f$ formula φ, with $T \subseteq \{\mathsf{G}, \mathsf{F}\}$, can be translated into an LTL formula $\bar{\varphi}$ (preserving the satisfiability) by exploiting the transformation $t(\varphi)$ defined in the work of De Giacomo et al. (2014b). In order to deal with the temporal operators not used there, we have just to consider the additional translation rules $t(\mathsf{X}(\varphi)) = \mathsf{X}(t(\varphi) \wedge \neg end)$, $t(\mathsf{X_w}(\varphi)) = \mathsf{X}(t(\varphi) \vee end)$, $t((\varphi_1 \,\mathsf{U}\, \varphi_2)) = (t(\varphi_1) \,\mathsf{U}\, (t(\varphi_2) \wedge \neg end))$, and $t((\varphi_1 \,\mathsf{R}\, \varphi_2)) = ((t(\varphi_1) \wedge \neg end) \,\mathsf{R}\, (t(\varphi_2) \vee end))$. With this extension, properties (2) and (3) in Claim 20 directly follows from the arguments by De Giacomo et al. (2014b).

To conclude the proof, recall that according to the work of Sistla and Clarke (1985, Thm. 4.7), whenever the LTL formula $\bar{\varphi}$ is satisfiable, it can be satisfied by a periodic trace such that the length of the prefix preceding the infinite periodic suffix has length $l \leq 2^{||\bar{\varphi}||}$. In particular, by property (2) in Claim 20, if φ is satisfiable, the finite prefix of each model of $\bar{\varphi}$ is the satisfying model of φ and the infinite periodic suffix consists of the infinite repetition of end. Thus, we can conclude that if φ is satisfiable, then it can be satisfied by that prefix, say π, such that $len(\pi) \leq 2^{||\bar{\varphi}||} = 2^{O(||\varphi||)}$.

Finally, by using similar arguments as those in the proof of Theorem 19, it can be checked that the result holds for LTL$_p$ formulas, too. □

In the rest of the section we show that this exponential bound is met over all classes of LTL$_f$ and LTL$_p$ formulas that are not covered by the analysis in Section 4.1. Proofs for LTL$_f$ formulas are based on the standard approach of encoding a n-bits counter, which counts from 0 to $2^n - 1$ over consecutive time instants of models. Even if such an approach has been already used in the literature (see, e.g., Lange, 2004), the lack of some temporal operators makes encodings somewhat complex, and worth being illustrated.

Theorem 24 *There are satisfiable $\langle\{\mathsf{U}\}\rangle$-LTL$_f$, $\langle\{\mathsf{X_w}, \mathsf{G}, \mathsf{F}\}\rangle$-LTL$_f$, and $\langle\{\mathsf{X}, \mathsf{G}\}\rangle$-LTL$_f$ (respectively, negation-free $\langle\{\mathsf{U}\}\rangle$-LTL$_p$, $\langle\{\mathsf{X_w}, \mathsf{G}, \mathsf{F}\}\rangle$-LTL$_p$, and $\langle\{\mathsf{X}, \mathsf{G}\}\rangle$- LTL$_p$) formulas φ for which there is no model whose length is polynomial with respect to $||\varphi||$.*

Proof. In the following we will construct a formula φ over the set $\mathcal{V}_\varphi = \{x_1, ..., x_n, cont\}$ of variables and belonging to the class $\langle\{\mathsf{X}, \mathsf{G}\}\rangle$-LTL$_f$, encoding the behavior of an n-bits counter. In order to improve the readability, we shall use the implication connectives "$\rightarrow$" and "$\leftrightarrow$". For their semantics, recall that $\alpha \rightarrow \beta$ is equivalent to $\neg\alpha \vee \beta$ and $\alpha \leftrightarrow \beta$ is equivalent to $(\alpha \vee \neg\beta) \wedge (\neg\alpha \vee \beta)$. Note that by rewriting the implications in the specific formula φ we are going to define, the size of the resulting formula is linearly bounded by the size of φ.

Formally, define $\varphi = \Phi_1 \wedge \Phi_2 \wedge \mathsf{G}(\neg cont \vee \Phi_{inc})$, where:

- $\Phi_1 = (\bigwedge_{i=1}^{n} \neg x_i \wedge cont)$ encodes the initial state of the counter; intuitively, $x_1, ..., x_n$ are variables one-to-one corresponding to the bits of the counter and, initially, all of them are false;
- $\Phi_2 = \mathsf{G}(\neg cont \leftrightarrow \bigwedge_{i=1}^{n} x_i)$, combined with Φ_1, encodes the fact that the variable $cont$ is forced to be true until the counter reaches the maximum value;
- $\Phi_{inc} = \bigvee_{i=1}^{n} \left(\neg x_i \wedge \mathsf{X}(x_i) \wedge \bigwedge_{j=1}^{j<i}(x_j \wedge \mathsf{X}(\neg x_j)) \wedge \bigwedge_{j=i+1}^{n}((x_j \rightarrow \mathsf{X}(x_j)) \wedge (\neg x_j \rightarrow \mathsf{X}(\neg x_j)))\right)$ encodes the behavior of the counter for moving from one number to the successive one; Φ_{inc} combined with Φ_2 implies the existence of a subsequent time instant in all time instants, but the one where the counter reaches the maximum value and the variable $cont$ becomes *false*;

Consider now a model π of φ. Initially, we have $\pi_0 = \{cont\}$. For each time instant h, let the *value* associated with h be defined as $val(h) = \sum_{x_i \in \pi_h} 2^{i-1}$.

Consider a time instant $h \geq 0$ with $\pi_h \cap \{x_1, ..., x_n\} \neq \{x_1, ..., x_n\}$, and let i be the minimum index such that $x_i \notin \pi_h$ and $x_j \in \pi_h$, for each $j \in \{1, ..., i-1\}$. That is, $\pi_h \cap \{x_1, ..., x_i\} = \{x_1, ..., x_{i-1}\}$. Let us analyze the time instant $h+1$. First, observe that due to Φ_2 and Φ_{inc}, this time instant is guaranteed to exist since we have $cont \in \pi_h$. Second, because of the formula Φ_{inc} which is enforced to hold at the time instant h, for each $k \in \{i+1, ..., n\}$, $x_k \in \pi_{h+1}$ if, and only if, $x_k \in \pi_h$ and thus, $\pi_h \cap \{x_{i+1}, ..., x_n\} = \pi_{h+1} \cap \{x_{i+1}, ..., x_n\}$. In addition, because of Φ_{inc}, we have that $\pi_{h+1} \cap \{x_1, ..., x_i\} = \{x_i\}$ and, hence, $val(h+1) = val(h) + 1$; that is, π_{h+1} encodes the next configuration of the counter with respect to the configuration encoded by π_h.

Eventually, given the subformula $\mathsf{G}(\neg cont \vee \Phi_{inc})$ and since we focus on finite traces, a time instant h^* is guaranteed to exist such that $\pi_{h^*} = \{x_1, ..., x_n\}$, i.e., $val(h^*) = 2^n - 1$. By combining this value with the above expressions relating the values of subsequent time instants, we conclude that $h^* = 2^n - 1$. Hence, $len(\pi) > h^* = 2^n - 1$. Thus, all models have at least exponential length. Note that the formula is actually satisfiable; for instance, we can take the model π with $len(\pi) = 2^n$ and such that $val(h) = h + 1$, for each $h \in \{0, ..., len(\pi) - 1\}$.

The above arguments have shown that the result holds over $\langle\{\mathsf{X}, \mathsf{G}\}\rangle$-LTL$_f$ formulas. By just replacing X with $\mathsf{X_w}$ and Φ_1 with $\Phi_1' = (\bigwedge_{i=1}^{n} \neg x_i \wedge cont) \wedge \mathsf{F}(\bigwedge_{i=1}^{n} x_i)$, it can be checked that the result holds over $\langle\{\mathsf{X_w}, \mathsf{G}, \mathsf{F}\}\rangle$-LTL$_f$ formulas. In particular, note that given the subformulas $\mathsf{F}(\bigwedge_{i=1}^{n} x_i)$ and $\mathsf{G}(\neg cont \vee \Phi_{inc})$, the subformula Φ_{inc} exhibits the same behavior when X is replaced with $\mathsf{X_w}$. Eventually, note that the above encoding can be easily adapted to work with formulas of the fragment $\langle\{\mathsf{U}\}\rangle$-LTL$_f$ by replacing X and G with U as follows: $\varphi = \Phi_1'' \wedge (\Phi_{inc}' \mathsf{U} \bigwedge_{i=1}^{n} x_i)$, where $\Phi_1'' = \bigwedge_{i=1}^{n} \neg x_i$ and $\Phi_{inc}' = \bigvee_{i=1}^{n}((\neg x_i \wedge \bigwedge_{j=1}^{j<i} x_j) \; \mathsf{U} \; (x_i \wedge \bigwedge_{j=1}^{j<i} \neg x_j) \wedge \bigwedge_{k=i+1}^{n}((x_k \; \mathsf{U} \; (x_i \wedge \bigwedge_{j=1}^{j<i} \neg x_j \wedge x_k)) \vee (\neg x_k \; \mathsf{U} \; (x_i \wedge \bigwedge_{j=0}^{j<i} \neg x_j \wedge \neg x_k)))$.

In order to conclude the proof, note that the results for negation-free $\langle\{\mathsf{U}\}\rangle$-LTL$_p$, $\langle\{\mathsf{X_w}, \mathsf{G}, \mathsf{F}\}\rangle$-LTL$_p$, and $\langle\{\mathsf{X}, \mathsf{G}\}\rangle$- LTL$_p$) formulas follow by combining the above proof with Theorem 14, Theorem 16 and Theorem 17. □

By exploiting the above result, we can moreover single out two LTL$_f$ and two LTL$_p$ fragments involving the R operator and which do not enjoy the linear-length model property.

Corollary 25 *There are satisfiable $\langle\{\mathsf{F}, \mathsf{R}\}\rangle$-LTL$_f$ and $\langle\{\mathsf{X}, \mathsf{R}\}\rangle$-LTL$_f$ (respectively, negation-free $\langle\{\mathsf{F}, \mathsf{R}\}\rangle$-LTL$_p$ and $\langle\{\mathsf{X}, \mathsf{R}\}\rangle$-LTL$_p$) formulas φ for which there is no model whose length is polynomial with respect to $||\varphi||$.*

Proof. Let us first focus on LTL$_f$. Note that G and U can be rewritten in terms of R and F:

(1) $\mathsf{G}(\varphi) = ((f \wedge \neg f) \; \mathsf{R} \; \varphi)$, where f is a fresh variable not in $\mathcal{V}_\varphi$;

(2) $(\varphi_1 \; \mathsf{U} \; \varphi_2) = \mathsf{F}(\varphi_2) \wedge (\varphi_2 \; \mathsf{R} \; (\varphi_1 \vee \varphi_2))$.

Indeed, by inspecting rule (1), one can notice that the subformula $(f \wedge \neg f)$ is not satisfiable and, thus, according to the semantics of R, the formula φ must remain true until the end of the trace. This behavior encodes exactly the semantics of the G operator. By looking at (2), note instead that the subformula $\mathsf{F}(\varphi_2)$ guarantees the existence of a time instant in which φ_2 becomes true. Moreover, according to the semantics of R, the subformula $(\varphi_1 \vee \varphi_2)$ must be true until and including the time instant in which φ_2 becomes true. This translates in the fact that φ_1 must be true until φ_2 becomes true, encoding exactly the semantics of the U operator.

Consider now negation-free LTL$_p$. Rule (2) clearly holds on this logic, too. Moreover, (1) can be modified as $\mathsf{G}(\varphi) = ((f_1 \wedge f_2) \; \mathsf{R} \; \varphi)$, where f_1 and f_2 are two fresh variables not in $\mathcal{V}_\varphi$. Clearly enough, the modification preserves the semantics of G over process traces.

The result eventually follows by combining the rewritings discussed above with Theorem 24. □

5. Complexity of Satisfiability over Finite Traces

In this section, we study the computational complexity of reasoning problems related to LTL$_p$ and LTL$_f$. In fact, it is easily seen that the problem of checking whether a trace is a model of a for-

Fragments	*negation-free* LTL$_p$	*negation-free* LTL$_f$	LTL$_p$	LTL$_f$
$\{X_w\}$ $\{G\}$ $\{F\}$ $\{R\}$ $\{X_w,G\}$ $\{X_w,R\}$ $\{G,R\}$ $\{X_w,G,R\}$	in P [Th. 35]	in P [Th. 31]	in P [Th. 35]	NP-complete [Th. 27]
$\{X\}$ $\{X,X_w\}$ $\{X,F\}$ $\{X_w,F\}$ $\{G,F\}$ $\{X,X_w,F\}$	NP-complete [Th. 34]	in P [Th. 31]	NP-complete [Th. 34]	NP-complete [Th. 27]
$\{U\}$ $\{X,G\}$ $\{X,U\}$ $\{X,R\}$ $\{X_w,U\}$ $\{G,U\}$ $\{F,U\}$ $\{F,R\}$ $\{U,R\}$ $\{X,X_w,G\}$ $\{X,X_w,U\}$ $\{X,X_w,R\}$ $\{X,G,F\}$ $\{X,G,U\}$ $\{X,G,R\}$ $\{X,F,U\}$ $\{X,F,R\}$ $\{X,U,R\}$ $\{X_w,G,F\}$ $\{X_w,G,U\}$ $\{X_w,F,U\}$ $\{X_w,F,R\}$ $\{X_w,U,R\}$ $\{G,F,U\}$ $\{G,F,R\}$ $\{G,U,R\}$ $\{F,U,R\}$ $\{X,X_w,G,F\}$ $\{X,X_w,G,U\}$ $\{X,X_w,G,R\}$ $\{X,X_w,F,U\}$ $\{X,X_w,F,R\}$ $\{X,X_w,U,R\}$ $\{X,G,F,U\}$ $\{X,G,F,R\}$ $\{X,G,U,R\}$ $\{X,F,U,R\}$ $\{X_w,G,F,U\}$ $\{X_w,G,F,R\}$ $\{X_w,G,U,R\}$ $\{X_w,F,U,R\}$ $\{G,F,U,R\}$ $\{X,X_w,G,F,U\}$ $\{X,X_w,G,F,R\}$ $\{X,G,F,U,R\}$ $\{X,X_w,F,U,R\}$ $\{X,X_w,G,U,R\}$ $\{X_w,G,F,U,R\}$ $\{X,X_w,G,F,U,R\}$	PSPACE-complete [Th. 32,Th 33]	in P [Th. 31]	PSPACE-complete [Th. 32,Th 33]	PSPACE-complete [Th. 28,Th. 29, Th. 30]

Figure 4: Summary of complexity results. For the sake of readability, all fragments enjoying the same computational properties are grouped together (even though the corresponding results required different proofs).

mula provided as input[5] is feasible in polynomial time, by using standard dynamic programming algorithms for (computation tree logic) model checking (Clarke & Schlingloff, 2001)—where the given trace is viewed as Kripke structure and, hence, branching is immaterial. For the sake of completeness, note that a direct proof of the result will be given in Section 7.2, where we shall present a method and some data structures used in our prototype implementation.

Theorem 26 *Checking whether a finite trace π is a model of an* LTL$_f$ *(respectively,* LTL$_p$*) formula φ is feasible in polynomial time.*

After the above observation, we now move to study the *satisfiability* problem of deciding whether a given formula admits a model. In particular, we consider the complexity of this problem over various fragments of LTL$_p$ and LTL$_f$ defined on the basis of the temporal operators being allowed. Our results are summarized in Figure 4.

A useful insight can be gained by contrasting the entries in the table with Figure 3. Indeed, it emerges that satisfiability is PSPACE-complete on all classes of formulas that do not enjoy the linear-length model property. By contrast, it is NP-complete on the remaining classes, except for $\langle\{X_w, G, R\}\rangle$-LTL$_p$ and $\langle\{F\}\rangle$-LTL$_p$ formulas on which it is tractable—on $\langle\{X_w, G, R\}\rangle$-LTL$_f$ and $\langle\{F\}\rangle$-LTL$_f$, the problem remains NP-complete, as it inherits the NP-hardness of propositional logic. Note also that, in order to shed further lights on the differences between LTL$_p$ and LTL$_f$,

5. Hence, we are considering a *combined complexity* setting where neither the trace nor the formula is fixed.

we have also analyzed their complexity over formulas where negation is not allowed. On negation-free LTL$_f$ formulas, satisfiability is always tractable independently on the allowed operators. By contrast, negation is transparent as far as LTL$_p$ formulas are concerned.

The rest of the section illustrates proofs of these results.

5.1 Complexity of LTL$_f$

Let us start by considering the satisfiability problem for LTL$_f$ formulas. Combining the results in the previous section with those in Section 4.1, the following is easily established.

Theorem 27 *Assume that* $T \subseteq \{\mathsf{G}, \mathsf{F}\}$*, or* $T \subseteq \{\mathsf{X_w}, \mathsf{G}, \mathsf{R}\}$*, or* $T \subseteq \{\mathsf{X}, \mathsf{X_w}, \mathsf{F}\}$*. Then satisfiability of* $\langle T \rangle$-LTL$_f$ *formulas is* NP*-complete.*

Proof. Observe first that the given classes of $\langle T \rangle$-LTL$_f$ formulas enjoy the linear-length model property—see Section 4.1. Therefore, given a formula φ belonging to one of these classes, a non-deterministic Turing Machine can guess a trace π such that $len(\pi)$ is linear in $||\varphi||$ (precise bounds on the length can be found in Section 4.1) and then check in polynomial time (cf. Theorem 26) that $\pi \models \varphi$. Note that, for such formulas φ, representing π takes polynomial space only, because in the worst case each state of π contains all the variables over which φ is defined. It follows that satisfiability is in NP over these classes.

In order to conclude, note that NP-hardness directly follows from the NP-hardness of the satisfiability for propositional logic. □

For classes that do not enjoy the linear-length model property, a complexity blow up occurs. This is illustrated in the following two results.

Theorem 28 *Assume that* $T \supseteq \{\mathsf{X}, \mathsf{G}\}$ *or* $T \supseteq \{\mathsf{X_w}, \mathsf{G}, \mathsf{F}\}$*. Then satisfiability of* $\langle T \rangle$-LTL$_f$ *formulas is* PSPACE*-complete.*

Proof. Membership in PSPACE for LTL$_f$ was shown by De Giacomo and Vardi (2013).

We show that satisfiability of $\langle \{\mathsf{X}, \mathsf{G}\} \rangle$-LTL$_f$ formulas is PSPACE-hard by exhibiting a reduction from the *propositional* STRIPS *planning problem with one effect only*, which is known to be PSPACE-hard (Bylander, 1994). We have a set P of propositional variables and a set Act of possible actions. Each action $A \in Act$ is a pair (φ_A, E_A) where φ_A (the precondition) is a conjunction of literals built over P and E_A (the effect) is a literal. We are given a goal φ_g defined as a conjunction of literals, and a set $I \subseteq P$ of variables that hold in the initial situation. The problem asks whether, starting from the initial situation, there is a sequence of actions that can be executed leading to reach a situation where the goal is achieved. In particular, an action A can be executed if its associated precondition holds in the current situation, and the effect of its execution is to build a novel situation where E_A holds and the truth value of the other variables remains unchanged. This semantics can be formally encoded via the formula $\varphi = \Phi_1 \wedge \Phi_2 \wedge \bigwedge_{A \in Act}(\Phi_A^i \wedge \Phi_A^{ii} \wedge \Phi_A^{iii})$ defined over the variables $P \cup Act$ (i.e., actions are transparently viewed as variables) such that:

- $\Phi_1 = \bigwedge_{c \in I} c \wedge \bigwedge_{c \in P \setminus I} \neg c$ is a subformula encoding the initial situation;
- $\Phi_2 = \mathsf{G}((\varphi_g \vee \bigvee_{A \in Act} \mathsf{X}(A)) \wedge (\varphi_g \vee \bigwedge_{A, A' \in Act, A \neq A'} (\mathsf{X}(\neg A) \vee \mathsf{X}(\neg A'))))$ is a subformula encoding that either the goal is reached or exactly one action is executed in the next time instant;

- $\Phi_A^i = \mathsf{G}(\varphi_g \vee \mathsf{X}(\bigvee_{A' \in Act, A' \neq A} A') \vee \varphi_A)$ is a subformula encoding (combined with Φ_2) that an action A can be executed in the next time instant only when the associated precondition φ_A currently holds;

- $\Phi_A^{ii} = \mathsf{G}(\varphi_g \vee \mathsf{X}(\bigvee_{A' \in Act, A' \neq A} A') \vee \mathsf{X}(E_A))$ is a subformula encoding that the execution of an action A causes that the associated effect E_A must hold;

- $\Phi_A^{iii} = \mathsf{G}(\varphi_g \vee \mathsf{X}(\bigvee_{A' \in Act, A' \neq A} A') \vee \bigwedge_{c \in P, c \neq E_A}((\neg c \vee \mathsf{X}(c)) \wedge (c \vee \mathsf{X}(\neg c))))$ is a subformula encoding that the value of variables not affected by the execution of an action remains unchanged.

It can be checked that φ is satisfiable if, and only if, there is a sequence of actions that can be executed leading to reach a situation where φ_g holds. Details are omitted as the STRIPS encoding approach is standard for LTL (and also for LTL$_f$, see De Giacomo & Vardi, 2013).

The result for $\langle\{\mathsf{X_w}, \mathsf{G}, \mathsf{F}\}\rangle$-LTL$_f$ formulas can be derived by adapting the above encoding. Indeed, given the formula φ, we build the formula $\bar{\varphi} = \varphi' \wedge \mathsf{F}(\varphi_g)$, where φ' is obtained from φ by substituting X with $\mathsf{X_w}$. Clearly, $\bar{\varphi}$ is satisfiable if, and only if, φ is satisfiable. □

Theorem 29 *Assume that $T \supseteq \{\mathsf{U}\}$. Then satisfiability of $\langle T\rangle$-LTL$_f$ formulas is* PSPACE*-complete.*

Proof. Membership in PSPACE for LTL$_f$ was shown by De Giacomo and Vardi (2013). We show that satisfiability of $\langle\{\mathsf{U}\}\rangle$-LTL$_f$ formulas is PSPACE-hard by adapting the reduction from the *propositional* STRIPS *planning problem with one effect only* of the proof of Theorem 28. In particular, it is possible to rewrite the formula φ into the formula $\bar{\varphi} = \Phi_p \wedge \bar{\Phi}_1 \wedge \bar{\Phi}_2 \wedge \bigwedge_{A \in Act}(\bar{\Phi}_A^i \wedge \bar{\Phi}_A^{ii} \wedge \bar{\Phi}_A^{iii})$ defined over the variables $P \cup Act \cup \{p\}$ as follows:

- $\Phi_p = ((p \,\mathsf{U}\, \neg p) \wedge (\neg p \,\mathsf{U}\, p)) \,\mathsf{U}\, \varphi_g$ is a subformula encoding the different time instants of the trace;

- $\bar{\Phi}_1 = p \wedge \bigwedge_{c \in I} c \wedge \bigwedge_{c \in P \setminus I} \neg c$ is a subformula encoding the initial situation;

- $\bar{\Phi}_2 = (((p \,\mathsf{U}\, (\neg p \wedge \bigvee_{A \in Act} A)) \vee (\neg p \,\mathsf{U}\, (p \wedge \bigvee_{A \in Act} A))) \,\mathsf{U}\, \varphi_g) \wedge (((p \,\mathsf{U}\, (\neg p \wedge \bigwedge_{A, A' \in Act, A \neq A'} (\neg A \vee \neg A'))) \vee (\neg p \,\mathsf{U}\, (p \wedge \bigwedge_{A, A' \in Act, A \neq A'} (\neg A \vee \neg A')))) \,\mathsf{U}\, \varphi_g)$ is a subformula encoding that either the goal is reached or exactly one action is executed in the next time instant;

- $\bar{\Phi}_A^i = (\varphi_A \vee (p \,\mathsf{U}\, (\neg p \wedge (\bigvee_{A' \in Act, A' \neq A} A'))) \vee (\neg p \,\mathsf{U}\, (p \wedge (\bigvee_{A' \in Act, A' \neq A} A')))) \,\mathsf{U}\, \varphi_g)$ is a subformula encoding (combined with $\bar{\Phi}_2$) that an action A can be executed in the next time instant only when the associated precondition φ_A currently holds;

- $\bar{\Phi}_A^{ii} = ((p \,\mathsf{U}\, (\neg p \wedge (E_A \vee \bigvee_{A' \in Act, A' \neq A} A'))) \vee (\neg p \,\mathsf{U}\, (p \wedge (E_A \vee \bigvee_{A' \in Act, A' \neq A} A')))) \,\mathsf{U}\, \varphi_g)$ is a subformula encoding that the execution of an action A causes that the associated effect E_A must hold;

- $\bar{\Phi}_A^{iii} = \bigwedge_{c \in P, c \neq E_A}((p \wedge c \,\mathsf{U}\, (\neg p \wedge (\bigvee_{A' \in Act, A' \neq A} A' \vee c))) \vee (p \wedge \neg c \,\mathsf{U}\, (\neg p \wedge (\bigvee_{A' \in Act, A' \neq A} A' \vee \neg c))) \vee (\neg p \wedge c \,\mathsf{U}\, (p \wedge (\bigvee_{A' \in Act, A' \neq A} A' \vee c))) \vee (\neg p \wedge \neg c \,\mathsf{U}\, (p \wedge (\bigvee_{A' \in Act, A' \neq A} A' \vee \neg c)))) \,\mathsf{U}\, \varphi_g)$ is a subformula encoding that the value of variables not affected by the execution of an action remains unchanged.

It can be checked that $\bar{\varphi}$ is satisfiable if, and only if, there is a sequence of actions that can be executed leading to reach a situation where φ_g holds. $\Box$

Classes of formulas involving the R operator are next analyzed.

Theorem 30 *Assume that $T \supseteq \{\mathsf{X}, \mathsf{R}\}$ or $T \supseteq \{\mathsf{F}, \mathsf{R}\}$. Then satisfiability of $\langle T \rangle$-LTL$_f$ formulas is* PSPACE-*complete.*

Proof. Membership in PSPACE for LTL$_f$ derives from the work of De Giacomo and Vardi (2013). In the light of the rewriting rules discussed in the proof of Corollary 25, hardness results for $\langle\{\mathsf{X}, \mathsf{R}\}\rangle$-LTL$_f$ and $\langle\{\mathsf{F}, \mathsf{R}\}\rangle$-LTL$_f$ formulas are a direct consequence of the hardness results reported in Theorem 28 and Theorem 29, respectively. $\Box$

We leave the section by illustrating a tractable case for LTL$_f$.

Theorem 31 *Deciding the satisfiability of negation-free $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}, \mathsf{R}\}\rangle$-LTL$_f$ formulas and computing a model, if any exists, are feasible in polynomial time.*

Proof. Let φ be a negation-free $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}, \mathsf{R}\}\rangle$-LTL$_f$ formula. By Theorem 18, if φ is satisfiable, then it admits a model of length at most $th(\varphi, \mathsf{X}) + th(\varphi, \mathsf{X_w}) + 1$. In particular, by inspection in the proof, the model π is such that $\pi_i = \mathcal{V}_\varphi$, for each $i \in \{0, ..., len(\pi)\text{-}1\}$. Thus, the result directly follows by Theorem 26. $\Box$

5.2 Complexity of LTL$_p$

We now analyze the complexity of LTL$_p$ formulas. In this setting, a number of results can be established by using the complexity results for LTL$_f$ and the mechanisms to "translate" LTL$_f$ formulas into LTL$_p$ formulas provided in Section 3.

Theorem 32 *The following properties hold:*

- *Assume that $T \subseteq \{\mathsf{X}, \mathsf{X_w}, \mathsf{F}\}$, with $\{\mathsf{X}\} \subseteq T$ or $\{\mathsf{X_w}, \mathsf{F}\} \subseteq T$. Then satisfiability of $\langle T \rangle$-LTL$_p$ formulas is* NP-*complete. Hardness results hold even over negation-free formulas.*
- *Assume that $T \supseteq \{\mathsf{U}\}$, or $T \supseteq \{\mathsf{X_w}, \mathsf{G}, \mathsf{F}\}$, or $T \supseteq \{\mathsf{X}, \mathsf{G}\}$. Then satisfiability of $\langle T \rangle$-LTL$_p$ formulas is* PSPACE-*complete. Hardness results hold even over negation-free formulas.*

Proof. Assume that $T \subseteq \{\mathsf{X}, \mathsf{X_w}, \mathsf{F}\}$. Recall from Section 4.1 that the classes considered in the statement enjoy the linear-length model property. So, membership in NP is established as in the proof of Theorem 27. Since satisfiability of $\langle\{\}\rangle$-LTL$_f$ (that is, of propositional formulas) is NP-hard, we conclude, by Theorem 14 and Theorem 16, that satisfiability of negation-free $\langle\{\mathsf{X}\}\rangle$-LTL$_p$ and negation-free $\langle\{\mathsf{X_w}, \mathsf{F}\}\rangle$-LTL$_p$ formulas is NP-hard, too.

Assume that $T \supseteq \{\mathsf{U}\}$, or $T \supseteq \{\mathsf{X_w}, \mathsf{G}, \mathsf{F}\}$, or $T \supseteq \{\mathsf{X}, \mathsf{G}\}$. The fact that satisfiability is in PSPACE follows by Corollary 13. As for the lower bound, it is enough to note that satisfiability of negation-free $\langle\{\mathsf{U}\}\rangle$-LTL$_p$, $\langle\{\mathsf{X_w}, \mathsf{G}, \mathsf{F}\}\rangle$-LTL$_p$, and $\langle\{\mathsf{X}, \mathsf{G}\}\rangle$-LTL$_p$ formulas is PSPACE-hard as a direct consequence of Theorem 14, Theorem 16, and Theorem 17 combined with the hardness results reported in Theorem 28 and Theorem 29. $\Box$

Then, we analyze the classes of formulas involving the R operator.

Theorem 33 *Assume that $T \supseteq \{\mathsf{X}, \mathsf{R}\}$, or $T \supseteq \{\mathsf{F}, \mathsf{R}\}$. Then satisfiability of $\langle T\rangle$-LTL$_p$ formulas is* PSPACE*-complete. Hardness results hold even over negation-free formulas.*

Proof. The fact that satisfiability is in PSPACE follows by Corollary 13. As for the lower bound, note that satisfiability of negation-free $\langle\{\mathsf{X}, \mathsf{R}\}\rangle$-LTL$_p$ and negation-free $\langle\{\mathsf{F}, \mathsf{R}\}\rangle$-LTL$_p$ formulas is PSPACE-hard because of the hardness results reported in Theorem 32 and the rewriting rules discussed in the proof of Corollary 25. □

After the above results, it remains to consider the complexity of satisfiability over (sub-)classes of $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$-LTL$_p$ and $\langle\{\mathsf{X_w}, \mathsf{G}, \mathsf{R}\}\rangle$-LTL$_p$ formulas. Concerning the first case, observe that we already know that satisfiability is NP-hard over the corresponding LTL$_f$ fragments. However, we do not have a general tool to translate this result over LTL$_f$—just check that Theorem 14, Theorem 16 and Theorem 17 do not apply here. Therefore, an ad-hoc analysis is in order, which is provided by the following two results.

Theorem 34 *Satisfiability of $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$-LTL$_p$ formulas is* NP*-complete. Hardness holds even over negation-free formulas.*

Proof. From the fact that the class $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$-LTL$_p$ of formulas enjoy the linear-length model property, membership in NP is established as in the proof of Theorem 27. For the hardness, we exhibit a reduction from the NP-hard ONE-IN-THREE POSITIVE 3-SAT problem (Garey & Johnson, 1979). Let ϕ be a propositional formula in conjunctive normal form $\phi = C_1 \wedge \cdots \wedge C_m$ over the set $\{x_1, ..., x_n\}$ of variables, where each clause C_j has the form $C_j = x_{j_1} \vee x_{j_2} \vee x_{j_3}$ with $x_{j_1}, x_{j_2}, x_{j_3} \in \{x_1, ..., x_n\}$. Based on ϕ, we build a negation-free $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$-LTL$_p$ formula φ_ϕ over the set $\{x_1, ..., x_n\}$ of variables. In particular, $\varphi_\phi = \bigwedge_{j=1}^{m}(\mathsf{F}(C_j) \wedge \bigwedge_{h=1}^{3} \mathsf{G}(\eta_{j,h}))$ where

$$\eta_{j,h} \quad = \quad \bigvee_{r\in\{1,...,n\}\setminus\{j_h\}} x_r \vee \mathsf{G}\left(\bigvee_{r\in\{1,...,n\}\setminus(\{j_1,j_2,j_3\}\setminus\{j_h\})} x_r\right).$$

We now claim that: *there is a truth assignment over $\{x_1, ..., x_n\}$ such that, for each clause of ϕ, precisely one variable evaluates true if, and only if, φ_ϕ is satisfiable.*

(if part) Assume that π is a PT-model of φ_ϕ. Consider the truth assignment σ such that x_i, for each $i \in \{1, ..., n\}$, evaluates true in σ if, and only if, there is a time instant $w \in \{0, ..., len(\pi)\text{-}1\}$ such that $\pi_w = \{x_i\}$. First, we show that σ is satisfying. Indeed, for each clause C_j, because $\pi, 0 \models_p \mathsf{F}(C_j)$, we are guaranteed about the existence of a time instant w_j where $\pi_{w_j} = \{x_r\}$ holds with $x_r \in \{x_{j_1}, x_{j_2}, x_{j_3}\}$. Now, let w_j^* be the minimum index enjoying this property and assume, without loss of generality, that $\pi_{w_j^*} = \{x_{j_1}\}$. Then the fact that $\pi, w_j^* \models_p \mathsf{G}(\eta_{j,j_1})$ holds, implies that, for each time instant $w' \geq w_j^*$, it is the case that $\pi_{w'} \cap \{x_{j_2}, x_{j_3}\} = \emptyset$. Given the choice of w_j^*, this holds even for the time instants w' antecedent to w_j^*. Therefore, x_{j_2} and x_{j_3} evaluate false in σ. It follows that, for each clause C_j, precisely one variable evaluates true in σ.

(only-if part) Consider a truth assignment σ such that, for each clause of ϕ, precisely one variable evaluates true. Let $\{x_{i_0}, ..., x_{i_{q\text{-}1}}\}$ be the set of all variables evaluating true in σ, and consider the finite trace π such that $len(\pi) = q$ and, for each $r \in \{0, ..., q\text{-}1\}$, $\pi_{i_r} = \{x_{i_r}\}$. In particular, note that π is a process trace. It can be checked that, for each $j \in \{1, ..., m\}$, the condition $\pi, w \models_p C_j$ holds at the time instant w associated with the (unique) variable $x_{i_w} \in \{x_{j_1}, x_{j_2}, x_{j_3}\}$ occurring in C_j and evaluating true in σ. Note that if it were the case that $\pi, 0 \not\models_p \mathsf{G}(\eta_{j,h})$, for

some $j \in \{1, ..., m\}$ and $h \in \{1, 2, 3\}$, then there would be two time instants w and w' such that $\pi_w = \{x_{j_h}\}$, $\pi_{w'} = \{x_{j_{h'}}\}$, with $\{x_{j_h}, x_{j'_h}\} \subset \{x_{j_1}, x_{j_2}, x_{j_3}\}$ and $h \neq h'$. But, this would entail that two variables, x_{j_h} and $x_{j_{h'}}$, belonging to the clause C_j evaluate true in σ, which is impossible. Therefore, $\pi \models_p \varphi_\phi$ holds.

The claim above established the NP-hardness of satisfiability for negation-free $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$-LTL$_p$ formulas, because φ_ϕ can be built in polynomial time. $\Box$

Finally, we leave the section by isolating two tractable classes of formulas.

Theorem 35 *Assume that either $T \subseteq \{\mathsf{X_w}, \mathsf{G}, \mathsf{R}\}$ or $T \subseteq \{\mathsf{F}\}$. Then satisfiability of $\langle T\rangle$-*LTL$_p$ *formulas is feasible in polynomial time.*

Proof. Let φ be a $\langle\{\mathsf{X_w}, \mathsf{G}, \mathsf{R}\}\rangle$-LTL$_p$ formula. If φ is satisfiable, then it is satisfiable by a trace of length 1 (cf. Fact 21). So, we can build all possible traces π such that $\pi = \pi_0$ and $|\pi_0| = 1$, by considering all variables in φ plus one further variable in $\mathcal{V} \setminus \mathcal{V}_\varphi$ (cf. Lemma 10). The result follows since there are linearly-many candidate traces and since, for each of them, checking whether the formula is satisfied is feasible in polynomial time (see Theorem 26).

Now, let φ be a $\langle\{\mathsf{F}\}\rangle$-LTL$_p$ formula. Without loss of generality, assume that the Boolean constants `true` and `false` are allowed in the syntax of φ, with their usual intended semantics. Consider the following algorithm that tries to build a PT-model π of φ. We start with an empty trace π, i.e., such that $len(\pi) = 0$. At the generic step, we take an arbitrary subformula $\mathsf{F}(\varphi')$ such that φ' is a propositional formula (without any temporal operator). Note that deciding whether a propositional formula φ' admits a PT-model π' (and compute one, if any) is feasible in polynomial time. We distinguish two cases: (i) if φ' is not satisfiable by a PT-model, then we replace the formula $\mathsf{F}(\varphi')$ with the constant `false`; (ii) if π' is a PT-model of φ' (with $len(\pi') = 1$), then we replace the formula $\mathsf{F}(\varphi')$ with the constant `true` and we modify π by inserting π'_0 as the initial state. The above procedure is repeatedly applied till we end up with a propositional formula $\bar{\varphi}$, i.e., till all subformulas involving a temporal operator have been processed. It can be checked that if $\bar{\varphi}$ does not have a PT-model (which can be again checked in polynomial time), then φ is not satisfiable. Otherwise, we build a PT-model $\bar{\pi}$ of $\bar{\varphi}$, and we further modify π by inserting $\bar{\pi}_0$ as the initial state. Then note that the resulting trace π is a PT-model of φ. $\Box$

6. Further Syntactic Restrictions and Islands of Tractability

The complexity analysis carried out in the previous section evidenced that satisfiability is intractable over most of the classes of LTL$_f$ and LTL$_p$ formulas we have considered. Motivated by these bad news, we propose an analysis where formulas can be restricted also based on the Boolean connectives being allowed. In many cases, our analysis identifies precisely the border of tractability, that is, the maximal combinations of Boolean connectives and temporal operators leading to tractability. As a subject of further research, it would be interesting to trace completely the border and to provide a finer-grained analysis of the tractable classes with the aim of assessing the precise complexity of the tractable fragments (e.g., in terms of circuit complexity).

A summary of the complexity results derived in this section is in Figure 5. Note that the border of tractability is completely charted over $\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}$, and U. Moreover, note that all tractability

Logics	Fragments	Complexity
negation-free LTL$_f$	T with $T \subseteq \{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}, \mathsf{R}\}$	in P [Th. 31]
LTL$_f$ LTL$_p$	T without conjunctions, with $T \subseteq \{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}\}$ T without disjunctions, with $T \subseteq \{\mathsf{X}, \mathsf{F}\}$ or $T \subseteq \{\mathsf{X}, \mathsf{X_w}, \mathsf{G}\}$ or $T \subseteq \{\mathsf{G}, \mathsf{F}\}$	in P [Th. 36, Th. 39, Th. 40, Th. 41]
LTL$_f$ negation-free LTL$_p$	T without conjunctions, with $T \supseteq \{\mathsf{X}, \mathsf{R}\}$ T without disjunctions, with $T \supseteq \{\mathsf{X}, \mathsf{G}, \mathsf{F}\}$ or $T \supseteq \{\mathsf{X_w}, \mathsf{F}\}$ or $T \supseteq \{\mathsf{U}\}$	NP-hard [Th. 37, Th. 38, Th. 42, Th. 43, Th. 44, Th. 45]

Figure 5: Summary of complexity results of Section 6. For the sake of readability, all fragments enjoying the same computational property are grouped together (even though the corresponding results required different proofs).

results we derive are established via constructive polynomial-time algorithms that either compute a (PT-) model or check that the given formula φ is not satisfiable. In particular, algorithms work on the *parse tree* $pt(\varphi) = (V, E, \lambda)$ of φ, which is a rooted tree (V, E) with a labeling function $\lambda : V \to \{\wedge, \vee, \mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}, \mathsf{R}\} \cup \{x, \neg x \mid x \in \mathcal{V}_\varphi\}$ inductively defined as follows:

- For any *literal* $\ell \in \{x, \neg x\}$ with $x \in \mathcal{V}_\varphi$, $pt(\ell) = (\{r\}, \emptyset, \lambda)$ and $\lambda(r) = \ell$.
- If $pt(\varphi_i) = (V_i, E_i, \lambda_i)$, with $i \in \{1, 2\}$, is a parse tree rooted at vertex $r_i \in V_i$ and if $\mathsf{O} \in \{\wedge, \vee\}$ is a Boolean connective or $\mathsf{O} \in \{\mathsf{U}, \mathsf{R}\}$, then $pt(\varphi_1 \ \mathsf{O} \ \varphi_2) = (\{r\} \cup V_1 \cup V_2, \{(r, r_1), (r, r_2)\} \cup E_1 \cup E_2, \lambda)$ is the parse tree rooted at the fresh node r, having r_1 and r_2 as its two children, and where λ is such that $\lambda(r) = \mathsf{O}$ and its restriction over V_i coincides wit λ_i.
- If $pt(\varphi') = (V', E', \lambda')$ is a parse tree rooted at r' and if $\mathsf{O} \in \{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}\}$ is a temporal operator, then $pt(\mathsf{O}(\varphi')) = (\{r\} \cup V', \{(r, r')\} \cup E', \lambda)$ is the tree rooted at the fresh node r whose only child is r', and where λ is such that $\lambda(r) = \mathsf{O}$ and its restriction over V' coincides with λ'.

6.1 Formulas without Conjunctions

We start with the case where conjunction is not allowed. In this case, we can establish tractability results for both LTL$_f$ and LTL$_p$, by do not allowing R as a temporal operator.

Theorem 36 *Satisfiability of* $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}\}\rangle$-LTL$_f$ *and* $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}\}\rangle$-LTL$_p$ *formulas without conjunctions is feasible in polynomial time.*

Proof. Let φ be a $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}\}\rangle$-LTL$_f$ formula without conjunctions, let $pt(\varphi) = (V, E, \lambda)$ be its parse tree, and let r be the root of $pt(\varphi)$—a similar line of reasoning applies to the case of $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}\}\rangle$- LTL$_p$ formulas. Consider an algorithm that processes $pt(\varphi)$ bottom-up, i.e., from the leaves to the root, by equipping each vertex $v \in V$ with a label $f(v)$ encoding either a trace or the special symbol $\perp$. In particular, if v is the vertex that is currently processed and v is a leaf with $\lambda(v) = x$ (respectively, $\lambda(v) = \neg x$), then we set $f(v) = \{x\}$ (respectively, $f(v) = \emptyset$). So, for all leaves, the labels consist of traces having length 1. By contrast, if v is an internal vertex (and its child/children have been already processed and have been equipped by the algorithm with a label), then we distinguish two cases:

- Assume that v has one child only, say c.
  - if $\lambda(v) = \mathsf{X}$ and $f(c) \neq \bot$, then $f(v)$ is defined as the trace obtained by concatenating the empty set $\emptyset$ (as the initial state) with the trace $f(c)$;
  - if $\lambda(v) = \mathsf{X}$ and $f(c) = \bot$, then $f(v) = \bot$;
  - if $\lambda(v) = \mathsf{X_w}$, then $f(v) = \emptyset$;
  - if $\lambda(v) = \mathsf{F}$, then $f(v) = f(c)$;
  - if $\lambda(v) = \mathsf{G}$ and $len(f(c)) > 1$, then $f(v) = \bot$;
  - if $\lambda(v) = \mathsf{G}$ and $len(f(c)) = 1$, then $f(v) = f(c)$.
- Assume that v has two children, say c_1 and c_2.
  - if $\lambda(v) = \vee$ and $f(c_1) = f(c_2) = \bot$, then $f(v) = \bot$;
  - if $\lambda(v) = \vee$ and $f(c_1) = \bot$ and $f(c_2) \neq \bot$, then $f(v) = f(c_2)$;
  - if $\lambda(v) = \vee$ and $f(c_2) = \bot$ and $f(c_1) \neq \bot$, then $f(v) = f(c_1)$;
  - if $\lambda(v) = \vee$ and $f(c_1) \neq \bot$ and $f(c_2) \neq \bot$, then $f(v)$ is defined as the trace having the minimum length between $f(c_1)$ and $f(c_2)$;
  - if $\lambda(v) = \mathsf{U}$, then $f(v) = f(c_2)$;

Now, for each vertex $v \in V$, let φ_v denote the subformula whose associated parse tree is given by the one rooted at v. By structural induction on the parse tree (again, from the leaves to the root), it can be checked that $f(v) = \bot$ holds if, and only if, φ_v is not satisfiable; in particular, if $f(v) \neq \bot$, then $f(v) \models \varphi_v$ and there is no trace π' with $len(\pi') < len(f(v))$ such that $\pi' \models \varphi_v$. Therefore, in order to check the satisfiability of φ, we can apply the above algorithm and then look at the label $f(r)$ of the root. The result follows as the algorithm is clearly feasible in polynomial time. $\Box$

Indeed, we can show an intractability result for R, actually coupled with X.

Theorem 37 *Satisfiability of $\langle T \rangle$-LTL$_f$ formulas without conjunctions, with $T \supseteq \{\mathsf{X}, \mathsf{R}\}$, is* NP-*hard.*

Proof. We exhibit a reduction from the 3-SAT problem (Garey & Johnson, 1979). Let ϕ be a propositional formula in conjunctive normal form $\phi = C_1 \wedge \cdots \wedge C_m$ over the set $\{x_1, ..., x_n\}$ of variables, where each clause C_j has the form $C_j = l_{j_1} \vee l_{j_2} \vee l_{j_3}$ with each l_{j_i} having the form x_k or $\neg x_k$, with $x_k \in \{x_1, ..., x_n\}$. Based on ϕ, we build the $\langle\{\mathsf{X}, \mathsf{R}\}\rangle$-LTL$_f$ formula φ_ϕ, over the set $\{x_1, ..., x_n\}$ of variables, as follows. Let s be a variable not occurring in ϕ, and define $\bar{C}_j = C_j \mathbin{\mathsf{R}} s$, for each $j \in \{1, ..., m\}$. Then, let $\varphi_\phi = (...((\bar{C}_1 \mathbin{\mathsf{R}} \bar{C}_2) \mathbin{\mathsf{R}} \bar{C}_3)...\mathsf{R}\ \bar{C}_m) \mathbin{\mathsf{R}} (s \mathbin{\mathsf{R}} \mathsf{X}(\neg s))$.

We claim that: *there is a truth assignment over $\{x_1, ..., x_n\}$ such that, for each clause of ϕ, at least one variable evaluates true if, and only if, φ_ϕ is satisfiable.*

(if part) Assume that π is a finite model of φ_ϕ. Consider the subformula $(s \mathbin{\mathsf{R}} \mathsf{X}(\neg s))$, and note that, according to the semantics of R, since its right hand involves one X operator, there must actually exist some time instant l where both s and $\mathsf{X}(\neg s)$ hold. Moreover, if $l > 0$, then $\mathsf{X}(\neg s)$ must hold for each time instant l' with $0 \leq l' < l$. This clearly implies that $l = 0$ and $\pi, 0 \models s$ while $\pi, 1 \models \neg s$. Then, because $\bar{C}_j$, for each $j \in \{1, ..., m\}$, includes s in its right hand side, it must be the case that $\pi, 0 \models C_j$, for each $j \in \{1, ..., m\}$. Therefore, the truth assignment σ such that x_i, for each $i \in \{1, ..., n\}$, evaluates true in σ if, and only if, $\pi, l \models x_i$ holds, is satisfying.

(only-if part) Consider any truth assignment σ such that, for each clause of ϕ at least one of its literals evaluates true. Consider the finite trace π such that $len(\pi) = 2$, $\pi_1 = \emptyset$ and π_0 includes s and all the variables evaluating true in σ. It can be easily checked that $\pi \models \varphi_\phi$.

As the formula φ_ϕ can be built in polynomial time, the NP-hardness follows. □

Theorem 38 *Satisfiability of negation-free* $\langle T \rangle$-LTL$_p$ *formulas without conjunctions, with* $T \supseteq \{\mathsf{X}, \mathsf{R}\}$*, is* NP*-hard.*

Proof. We exhibit again a reduction from the 3-SAT problem (Garey & Johnson, 1979). Let ϕ be a propositional formula in conjunctive normal form $\phi = C_1 \wedge \cdots \wedge C_m$ over the set $\{x_1, ..., x_n\}$ of variables, where each clause C_j has the form $C_j = l_{j_1} \vee l_{j_2} \vee l_{j_3}$ with each l_{j_i} having the form x_k or $\neg x_k$ with $x_k \in \{x_1, ..., x_n\}$. Based on ϕ, we build a negation-free $\langle \{\mathsf{X}, \mathsf{R}\} \rangle$-LTL$_p$ formula φ_ϕ, over the set $\{s, x_1, ..., x_n\} \cup \{\bar{s}, \bar{x}_1, ..., \bar{x}_n\}$ of variables, as follows. For each variable $x_k \in \{x_1, ..., x_n\}$, we define the two formulas φ_{x_k}=$\mathsf{X}^{k+1} x_k$ and $\varphi_{\neg x_k}$=$\mathsf{X}^{k+1} \bar{x}_k$. Moreover, for each clause C_j with $j \in \{1, ..., m\}$ we define $\varphi_{C_j} = (\varphi_{l_{j_1}} \vee \varphi_{l_{j_2}} \vee \varphi_{l_{j_3}})\ \mathsf{R}\ s$. Then, we have that

$$\varphi_\phi = (...((\varphi_{C_1}\ \mathsf{R}\ \varphi_{C_2})\ \mathsf{R}\ \varphi_{C_3})...\mathsf{R}\ \varphi_{C_m})\ \mathsf{R}\ (s\ \mathsf{R}\ \mathsf{X}(\bar{s})).$$

We claim that: *there is a truth assignment over* $\{x_1, ..., x_n\}$ *such that, for each clause of* ϕ*, at least one variable evaluates true if, and only if,* φ_ϕ *is satisfiable.*

(if part) Assume that π is PT-model of φ_ϕ. By using the same line of reasoning as in the proof of Theorem 37 and the fact that s and $\bar{s}$ cannot be simultaneously true, we derive that $\pi, 1 \models_p \bar{s}$ and $\pi, 0 \models_p s \wedge \varphi_{C_1} \wedge ... \wedge \varphi_{C_m}$. Therefore, the truth assignment σ such that x_i, for each $i \in \{1, ..., n\}$, evaluates true in σ if, and only if, $\pi, i{+}1 \models_p x_i$ holds, is satisfying.

(only-if part) Consider any satisfying truth assignment σ. Let Q be the set of variables evaluating true in σ, and consider the finite trace π such that $len(\pi) = n{+}2$, $\pi_0 = \{s\}$, $\pi_1 = \{\bar{s}\}$, and $\pi_{i+1} = \{x_i\}$ if $x_i \in Q$ and $\pi_{i+1} = \{\bar{x}_i\}$ if $x_i \notin Q$, for each $i \in \{1, ..., n\}$. It can be easily checked that $\pi \models_p \varphi_\phi$.

As the formula φ_ϕ can be built in polynomial time, the NP-hardness follows. □

6.2 Formulas without Disjunctions: Tractable Cases

Let us then move to the case where disjunctions are not allowed. In this case, we are able to identify three tractable classes.

Theorem 39 *Satisfiability of* $\langle \{\mathsf{X}, \mathsf{F}\} \rangle$-LTL$_f$ *and* $\langle \{\mathsf{X}, \mathsf{F}\} \rangle$-LTL$_p$ *formulas without disjunctions is feasible in polynomial time.*

Proof. Let φ be a $\langle\{\mathsf{X},\mathsf{F}\}\rangle$-LTL$_f$ formula without disjunctions. Observe first that φ can be rewritten in polynomial time according to the following grammar:

$$\begin{array}{rcl} \varphi & ::= & \varphi_B \mid \mathsf{X}(\varphi) \mid \mathsf{F}(\varphi) \mid (\varphi' \wedge \varphi') \mid (\varphi_B \wedge \mathsf{X}(\varphi)) \mid (\varphi_B \wedge (\mathsf{X}(\varphi) \wedge \varphi'))\mid (\mathsf{X}(\varphi) \wedge \varphi') \mid (\varphi_B \wedge \varphi') \\ \varphi' & ::= & \mathsf{F}(\varphi) \mid (\varphi' \wedge \varphi') \\ \varphi_B & ::= & x \mid \neg x \mid (\varphi_B \wedge \varphi_B) \end{array}$$

Consider then the parse tree $pt(\varphi) = (V, E, \lambda)$ and, for each vertex $v \in V$, let φ_v denote the subformula whose associated parse tree is given by the one rooted at v. Let $V' \subseteq V$ be the set of all vertices v for which φ_v is a maximal subformula without temporal operators. With each vertex $v \in V'$, let us associate a set M^v containing all variables occurring positively in φ_v. If some variable occurring negated in φ_v also occurs in M^v, then φ_v is not satisfiable (and φ is not satisfiable, too).

Let us then consider the case where the above test fails over all vertices in V'. In this case, M^v is a minimal model of φ_v, for each $v \in V'$. Moreover, note that such minimal models are built in polynomial time. Now, based on such models M^v, we build a model of φ in polynomial time. To this end, we start by associating with each vertex $v \in V'$ a trace π^v with $len(\pi^v) = 1$ and $\pi^v_0 = M^v$. Then, we process the parse tree $pt(\varphi)$ from the vertices in V' to the root by associating a trace π^v with each vertex v. In particular, depending on the expression in the above grammar which generates φ_v, the trace π^v is built as follows:

[$\varphi_v ::= \mathsf{X}(\varphi)$] Assume that c is the child of v. Then, π^v is obtained from π^c by inserting the state $\emptyset$ as the initial state;

[$\varphi_v ::= \mathsf{F}(\varphi)$] Assume that c is the child of v. Then, $\pi^v = \pi^c$;

[$\varphi_v ::= (\varphi' \wedge \varphi')$] Assume that c_1 and c_2 are the children of v. Then, π^v is obtained by appending π^{c_2} to the trace π^{c_1};

[$\varphi_v ::= (\varphi_B \wedge (\mathsf{X}(\varphi) \wedge \varphi'))$] Assume that c_1 and c_2 are the children of v and that c_1 is the child such that $c_1 \in V'$. Then, π^v is obtained from π^{c_2} by replacing its initial state (in fact, note that $\pi^{c_2}_0 = \emptyset$) with M^{c_1};

[$\varphi_v ::= (\varphi_B \wedge \mathsf{X}(\varphi))$] Assume that c_1 and c_2 are the children of v and that c_1 is the child such that $c_1 \in V'$. Then, π^v is obtained from π^{c_2} by replacing its initial state (in fact, note that $\pi^{c_2}_0 = \emptyset$) with M^{c_1};

[$\varphi_v ::= (\mathsf{X}(\varphi) \wedge \varphi')$] Assume that c_1 and c_2 are the children of v and that c_1 is the child such that $\lambda(c_1) = \mathsf{X}$. Then, π^v is obtained by appending π^{c_2} to the trace π^{c_1};

[$\varphi_v ::= (\varphi_B \wedge \varphi')$] Assume that c_1 and c_2 are the children of v and that c_1 is the child such that $c_1 \in V'$. Then, π^v is obtained by inserting M^{c_1} before the first time instant of π^{c_2}.

By structural induction on the subformulas of φ, we can easily check that π^v is a model of φ_v, for each vertex v. So, if r is the root of the parse tree, then π^r is a model of φ.

The result for LTL$_p$ formulas follows by first checking after the above procedure that, for each vertex $v \in V'$, $|M^v| \leq 1$ holds. If this is not the case, then there is no PT-model. Otherwise, on the resulting trace π^r, we just replace each state π^r_i such that $\pi^r_i = \emptyset$ with the state $\{p\}$, with p being a fresh variable not in $\mathcal{V}_\varphi$ (cf. Lemma 10). $\Box$

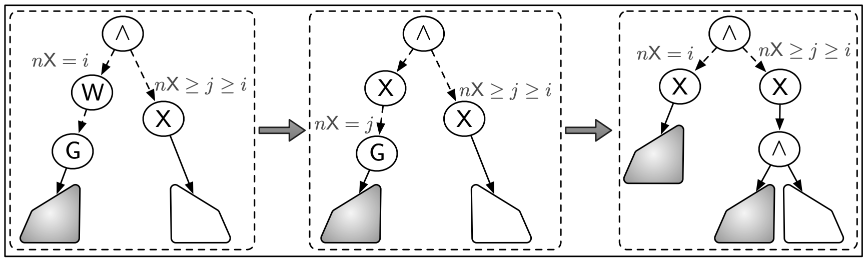


Figure 6: Example modification of the parse tree used in the proof of Theorem 40—$n\mathsf{X}$ indicates the number of vertices having label X in the path connecting the given vertex to the root.

Theorem 40 *Satisfiability of* $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}\}\rangle$-LTL$_f$ *and* $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}\}\rangle$-LTL$_p$ *formulas without disjunctions is feasible in polynomial time.*

Proof. Let φ be a $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}\}\rangle$-LTL$_f$ formula without disjunctions—the same line of reasoning applies to $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}\}\rangle$-LTL$_p$ formulas. We assume that the alphabet of the propositional language is naturally extended with the Boolean constant `true`, with the usual intended semantics. Hence, we next show a tractability result slightly more general than the one reported in the statement.

Note first that X, $\mathsf{X_w}$ and G distribute over $\wedge$ and we can assume, w.l.o.g., that if $\mathsf{O}(\varphi')$ is a subformula of φ with $\mathsf{O} \in \{\mathsf{X}, \mathsf{X_w}, \mathsf{G}\}$, then either φ' is a propositional formula (i.e., without temporal operators) or has the form $\mathsf{O}(\varphi'')$ with $\mathsf{O} \in \{\mathsf{X}, \mathsf{X_w}, \mathsf{G}\}$. We can rewrite φ so that G operators are never applied to $\mathsf{X_w}$ subformulas by exploiting the equivalences $\mathsf{G}(\mathsf{X_w}(\varphi')) = \mathsf{X_w}(\mathsf{G}(\varphi'))$ and $\mathsf{G}(\mathsf{G}(\varphi')) = \mathsf{G}(\varphi')$. After the rewriting, if φ has a subformula $\mathsf{G}(\varphi')$ and, in its turn, φ' has the form $\mathsf{X}(\varphi'')$, then φ is not satisfiable. This condition can be trivially checked in polynomial time. Hence, we consider in the following the remaining case where if $\mathsf{G}(\varphi')$ is a subformula of φ, then φ' is actually a propositional formula.

Let $pt(\varphi)$ be the parse tree of φ with root r and let us indicate by n the maximum number of consecutive vertices having label X among all the paths connecting r to either a vertex having label G or $\mathsf{X_w}$ or to a vertex associated with a literal. We modify φ in three steps, which are next presented in terms of modifications made on the associated parse tree $pt(\varphi)$. In the first step, we reduce the number of vertices labeled $\mathsf{X_w}$ by substituting them with vertices labeled X. Let W be the set of vertices r' of $pt(\varphi)$ that satisfy the following three conditions: *(i)* r' has label $\mathsf{X_w}$; *(ii)* there is no other vertex having label $\mathsf{X_w}$ in the path between r and r'; and *(iii)* the number h of vertices having label X in the path connecting r to r' is lower that n. Then, we set the label of all the nodes in W to X, and we repeat the process by computing the new value of n and the associated set W, until this set is empty. Note that because of property *(iii)*, we are guaranteed that the semantics of the formula after the modification is not altered.

At the end of the first step, all vertices r' in $pt(\varphi)$ labeled as $\mathsf{X_w}$ are such that the number h of vertices having label X in the path connecting r' to r is such that $h \geq n$. Due to the semantics of $\mathsf{X_w}$, this means that we can replace the subtree rooted at r' with a vertex `true` as a child of the parent of r', or just return `true` if $r = r'$. Note that at the end of the second step, the current parse tree $pt(\varphi)$ has vertices labeled only as X, G, $\wedge$ and literals, while being completely equivalent to the original parse tree provided as input.

For the final step, let $pt(\mathsf{G}(\varphi'))$ be any subtree of $pt(\varphi)$ such that its root r' has label G, and let h be the number of vertices having label X in the path connecting r' to r. By recalling that φ' is actually a propositional formula, we modify $pt(\varphi)$ by performing the following steps—see Figure 6, for an illustration:

1. remove the subtree $pt(\mathsf{G}(\varphi'))$, and append $pt(\varphi')$ as a child of the parent of r' (or just return $pt(\varphi')$, if $r = r'$);
2. for each subtree $pt(\mathsf{X}(\varphi''))$ whose root r'' is labeled as X and such that there are at least h vertices labeled X in the path connecting r'' to r: add as a child of r'' a new vertex $\bar{v}$ labeled as $\wedge$; append as a child of $\bar{v}$ the subtree $pt(\varphi')$; append as the other child of v the subtree $pt(\varphi'')$.

Note that the modified parse tree encodes a formula where the semantics of the subformula $\mathsf{G}(\varphi')$ is made explicit. Accordingly, we can repeatedly apply the above transformation, until we end up with a formula in $\langle\{\mathsf{X}\}\rangle$-LTL$_f$ without disjunctions. Therefore, we have shown that satisfiability of $\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}\}\rangle$-LTL$_f$ formulas without disjunctions can be solved by means of a polynomial-time preprocessing ending up with equivalent $\langle\{\mathsf{X}\}\rangle$-LTL$_f$ formulas without disjunctions. The result then follows because of Theorem 39. $\square$

Theorem 41 *Satisfiability of* $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$-LTL$_f$ *and* $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$-LTL$_p$ *formulas without disjunctions is feasible in polynomial time.*

Proof. We shall show that satisfiability is feasible in polynomial time over fragments that are actually larger than those reported in the statement. Indeed, we shall exhibit a polynomial-time satisfiability algorithm for LTL$_f$ and LTL$_p$ formulas φ built without disjunctions, where G and F are the only allowed temporal operators, and where the Boolean constant `true` can be additionally used (as in the proof of Theorem 40). So, let φ be a formula of this kind. More formally, observe that φ can be rewritten in polynomial time according to the following grammar:

$$\begin{array}{rcl} \varphi & ::= & \varphi_B \mid (\varphi_B \wedge \mathsf{G}(\varphi_B)) \mid (\varphi_B \wedge \varphi') \mid (\varphi_B \wedge (\mathsf{G}(\varphi_B) \wedge \varphi')) \\ \varphi' & ::= & \mathsf{F}(\varphi) \mid (\varphi' \wedge \varphi') \\ \varphi_B & ::= & x \mid \neg x \mid (\varphi_B \wedge \varphi_B) \mid \texttt{true} \end{array}$$

In order to assess the correctness of the grammar, note first that $\mathsf{G}(\varphi' \wedge \varphi'') = \mathsf{G}(\varphi') \wedge \mathsf{G}(\varphi'')$ and $\mathsf{G}(\mathsf{G}(\varphi)) = \mathsf{G}(\varphi)$ hold. Thus, we can assume, without loss of generality, that G operators are applied either to propositional formulas or to F subformulas. And, to conclude, we just observe that formulas having the form $\mathsf{G}(\mathsf{F}(\varphi'))$ can be safely replaced by $\mathsf{F}(\mathsf{G}(\varphi'))$—hence, we can furthermore assume that G operators are applied to propositional formulas only. Indeed, if π is a (PT-)model of $\mathsf{G}(\mathsf{F}(\varphi'))$, then φ' must hold in particular at the last time instant and, hence, π is a (PT-)model of $\mathsf{F}(\varphi')$ and of $\mathsf{F}(\mathsf{G}(\varphi'))$, too. On the other hand, if π is a (PT-)model of $\mathsf{F}(\mathsf{G}(\varphi'))$, then there exists a time instant from which φ' always holds. In particular, φ' holds at the last time instant and, hence, π is a (PT-)model of $\mathsf{F}(\varphi')$ and of $\mathsf{G}(\mathsf{F}(\varphi'))$, too.

Consider the parse tree $pt(\varphi) = (V, E, \lambda)$ rooted at the vertex $r \in V$. For each vertex $v \in V$, let φ_v denote the subformula whose associated parse tree is given by the one rooted at v. Each vertex v is also associated with a pair (Ω_v^+, Ω_v^-) where Ω_v^+ (respectively, Ω_v^-) is the set containing all variables x for which a leaf node c occurs in the subtree rooted at v with $\lambda(c) = x$ (respectively, $\lambda(c) = \neg x$). Note that, for all v such that φ_v is a propositional formula, φ_v is satisfiable if, and only if, $\Omega_v^+ \cap \Omega_v^- = \emptyset$. In particular, $M \models \varphi_v$ if, and only if, $M \supseteq \Omega_v^+$ and $M \cap \Omega_v^- = \emptyset$.

Let L be the set of all nodes $\ell \in V$ such that φ_ℓ is a maximal Boolean (sub-)formula such that either $\ell = r$ or the parent p of ℓ is such that $\lambda(p) \neq \mathsf{G}$. Let G be the set of all nodes $g \in V$ such that φ_g is a maximal Boolean (sub-)formula and such that the parent of g is p and we have $\lambda(p) = \mathsf{G}$. With each node $\ell \in L$, we associate a pair $(\Lambda_\ell^+, \Lambda_\ell^-)$ as follows. For each $\ell \in L$ and $g \in G$, let $root(\ell, g)$ be their minimum common ancestor. If there is $g \in G$ for which each vertex $v' \neq g$ in the path between $root(\ell, g)$ and g is such that $\lambda(v') \in \{\wedge, \mathsf{G}\}$, then Λ_ℓ^+ (respectively, Λ_ℓ^-) is defined as Ω_g^+ (respectively, Ω_g^-). Note that there can be at most one vertex $g \in G$ satisfying the above condition. Whenever a node $g \in G$ of this kind does not exist, then we set $\Lambda_\ell^+ = \emptyset$ and $\Lambda_\ell^- = \emptyset$.

By construction, if there is a node $\ell \in L$ such that $(\Omega_\ell^+ \cup \Lambda_\ell^+) \cap (\Omega_\ell^- \cup \Lambda_\ell^-) \neq \emptyset$, then the formula φ is not satisfiable. Indeed, in the recursive definition of satisfiability of φ, all variables in Λ_l^+ (respectively, Λ_l^-) must necessarily evaluate true (respectively, false) in all time instants coinciding with and following the one where φ_ℓ evaluates true. So, in the following, we consider the remaining case where $(\Omega_\ell^+ \cup \Lambda_\ell^+) \cap (\Omega_\ell^- \cup \Lambda_\ell^-) = \emptyset$, for each $\ell \in L$. In particular, equipped with the notions introduced above, consider the set F of edges over L built as follows:

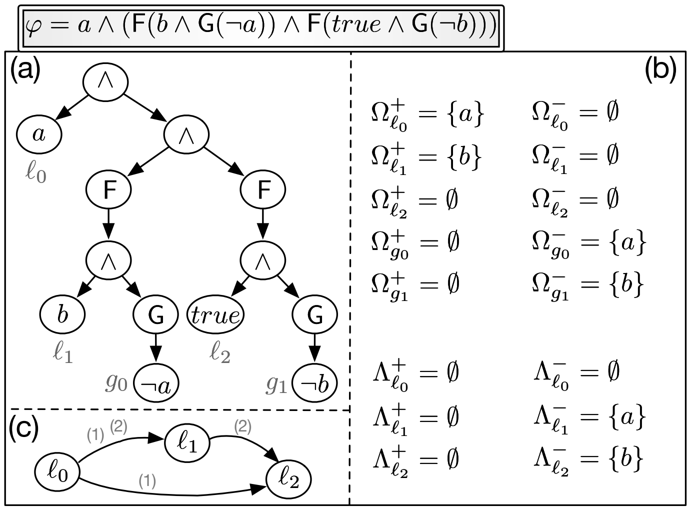
1. for each pair of distinct vertices ℓ, ℓ' in L such that ℓ is a child of a vertex p, and p is an ancestor of ℓ', the edge (ℓ, ℓ') is in F; note that in this case, there is at least one vertex v' in the path between p and ℓ' such that $\lambda(v') = \mathsf{F}$;

2. for each pair of distinct vertices ℓ, ℓ' in L such that $(\Lambda_\ell^+ \cup \Omega_\ell^+ \cup \Lambda_{\ell'}^+) \cap (\Lambda_\ell^- \cup \Omega_\ell^- \cup \Lambda_{\ell'}^-) \neq \emptyset$, the edge (ℓ, ℓ') is in F;

3. no further edge is in F.

Note that the construction is feasible in polynomial time. In order to illustrate it, consider the formula $\varphi = a \wedge (\mathsf{F}(b \wedge \mathsf{G}(\neg a)) \wedge (\mathsf{F}(\texttt{true} \wedge G(\neg b))$ built over the variables in $\{a, b\}$. The parse tree is shown in Figure 7 (a). In this case, we have $L = \{\ell_0, \ell_1, \ell_2\}$ and $G = \{g_0, g_1\}$. Figure 7 (b) reports for each ℓ_i, with $i \in \{0, 1, 2\}$ the sets $\Omega_{\ell_i}^+, \Omega_{\ell_i}^-, \Lambda_{\ell_i}^+, \Lambda_{\ell_i}^-$. Note that, for example, $\Omega_{\ell_1}^+ = \{b\}$ and $\Lambda_{\ell_1}^- = \Omega_{g_0}^- = \{a\}$ since $root(\ell_1, g_0) = v$ and all the nodes in the path from v to g_0 have label $\wedge$ or G. Finally, Figure 7 (c) reports the graph (L, F), where each edge is associated with the indication about whether it has been inserted because of rule (1) or (2). As an example, the edge (ℓ_1, ℓ_2) has been inserted because $(\Lambda_{\ell_1}^+ \cup \Omega_{\ell_1}^+ \cup \Lambda_{\ell_2}^+) \cap (\Lambda_{\ell_1}^- \cup \Omega_{\ell_1}^- \cup \Lambda_{\ell_2}^-) = \{b\} \cap \{a, b\} = \{b\} \neq \emptyset$.

For the correctness of the approach, suppose first that (L, F) is acyclic. Assume that $\ell_0, ..., \ell_{m\text{-}1}$ is a topological order of (L, F). We claim that the trace π such that $\pi_\ell = \Omega_\ell^+ \cup \bigcup_{\ell' \leq \ell} \Lambda_{\ell'}^+$ is a model of φ. In particular, we claim that the formula φ_i corresponding to the node $\ell_i \in L$ can be required to hold at the time instant i, in the recursive definition of satisfiability (see Section 2). Consider two subformulas of φ, $\mathsf{F}(\varphi_i \wedge \Psi_i)$ and $\mathsf{F}(\varphi_j \wedge \Psi_j)$, where φ_i and φ_j are propositional formulas and where Ψ_i and Ψ_j are the (possibly empty) remaining conjunctions. If φ_j is nested in Ψ_i, according to the recursive definition of satisfiability, φ_j must be satisfied in a time instant j greater than or equal to i. In fact, we shall show that $j > i$ holds. Indeed, according to the rule (1) above, the edge (ℓ_i, ℓ_j) is added to F and, thus, in the topological order, ℓ_i precedes ℓ_j. Assume now that neither φ_i is nested in Ψ_j nor φ_j is nested in Ψ_i. Then, the satisfaction of φ_j (in the time instant j) is forced to follow the satisfaction of φ_i (in the time instant i) in all the models of φ, if Ψ_j contains a



$\varphi = a \wedge (\mathsf{F}(b \wedge \mathsf{G}(\neg a)) \wedge \mathsf{F}(true \wedge \mathsf{G}(\neg b)))$
(a)
(b)
(c)
$\wedge$
a
ℓ_0
F
b
ℓ_1
G
g_0
$\neg a$
$true$
ℓ_2
g_1
$\neg b$
$\Omega^+_{\ell_0} = \{a\}$ $\Omega^-_{\ell_0} = \emptyset$
$\Omega^+_{\ell_1} = \{b\}$ $\Omega^-_{\ell_1} = \emptyset$
$\Omega^+_{\ell_2} = \emptyset$ $\Omega^-_{\ell_2} = \emptyset$
$\Omega^+_{g_0} = \emptyset$ $\Omega^-_{g_0} = \{a\}$
$\Omega^+_{g_1} = \emptyset$ $\Omega^-_{g_1} = \{b\}$
$\Lambda^+_{\ell_0} = \emptyset$ $\Lambda^-_{\ell_0} = \emptyset$
$\Lambda^+_{\ell_1} = \emptyset$ $\Lambda^-_{\ell_1} = \{a\}$
$\Lambda^+_{\ell_2} = \emptyset$ $\Lambda^-_{\ell_2} = \{b\}$
(1) (2)
(2)
(1)

Figure 7: Example construction of the graph (L, F) of Theorem 41.

subformula $\mathsf{G}(\varphi_k)$ such that the satisfaction of φ_k implies that a literal x must be true (respectively, false) and φ_i requires the same literal to be false (respectively, true). In that case, we have again to show that $j > i$. Indeed, according to rule (2) above, the edge (ℓ_i, ℓ_j) is added to F, and thus, in the topological order, ℓ_i precedes ℓ_j.

On the other hand, assume that (L, F) contains a cycle. Note that the edges generated by the application of rule (1) alone can never generate cycles. Since we have preprocessed the instance by ensuring that $(\Omega^+_\ell \cup \Lambda^+_\ell) \cap (\Omega^-_\ell \cup \Lambda^-_\ell) = \emptyset$, for each $\ell \in L$, F contains at least one edge generated by rule (2). So, consider the cycle $\ell_{i_1}, ..., \ell_{i_{k-1}}, \ell_{i_k}$, with $\ell_{i_1} = \ell_{i_k}$. Let $(\ell_{i_j}, \ell_{i_{j+1}})$ be an edge in this cycle and let φ_{i_j} and $\varphi_{i_{j+1}}$ be the formulas corresponding to ℓ_{i_j} and $\ell_{i_{j+1}}$, respectively. If the edge is generated by rule (1), then the time instant where $\varphi_{i_{j+1}}$ is required to hold (in the recursive definition of satisfiability) cannot occur before the time instant where φ_{i_j} is required to hold. If the edge is generated by rule (2), then there is a literal that is required to hold by the subformula $\mathsf{G}(\varphi_B)$ of $\varphi_{i_{j+1}}$ and whose opposite is required to hold by φ_{i_j}. Thus, φ_{i_j} has to be satisfied in a time instant that precedes the time instant where $\varphi_{i_{j+1}}$ is required to hold. This reasoning applies to all the edges of the cycle and we know that there is one edge that is generated by a rule of kind (2). Consider, without loss of generality, that such an edge is $(\ell_{i_{k-1}}, \ell_{i_k} = \ell_{i_1})$. This means that the time instant i_{k-1} in which $\varphi_{i_{k-1}}$ is satisfied can never follow or be equal to i_1. However, due to the precedence relations introduced by the other edges in the cycle we have also that the time instant i_{k-1} cannot occur before i_1, thus generating a contradiction. This means that there is no model that can accommodate the corresponding requirements.

Finally, observe that the above algorithm can be extended to deal with $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$-LTL$_p$ formulas without disjunctions, by preprocessing the instance to ensure that $|\Omega^+_\ell \cup \Lambda^+_\ell| \leq 1$ holds, for each $\ell \in L$, and by adding further edges (ℓ, ℓ') in F whenever $|(\Omega^+_\ell \cup \Lambda^+_{\ell'})| > 1$ or $|(\Lambda^+_\ell \cup \Lambda^+_{\ell'})| > 1$. The first kind of edges are required to take into account the fact that all the subformulas requiring some variable to be true from the time instant in which they are satisfied onwards must follow those subformulas requiring some literal to hold. This is necessary to fulfill the PT-model requirement. The second kind of edges results in the addition of cycles of length two between those pairs of subformulas both requiring some variable to be true from the time instant in which they are satisfied onwards. Indeed, in this case there is no PT-model that can satisfy the formula since all the models require to have at least one time instant (the last one) where more than one variable is true. At the end of the computation, we have just to check that the trace π derived from any (arbitrarily chosen) topological order of the nodes in L is such that $|\pi_\ell| \leq 1$, for each $\ell \in \{0, ..., m\text{-}1\}$—if $|\pi_\ell| = 0$, the state π_ℓ has to be replaced by $\{p\}$, with $p \notin \mathcal{V}_\varphi$ (cf. Lemma 10). □

6.3 Formulas without Disjunctions: Hard Cases

Moving to the discussion of the intractable fragments for formulas without disjunctions, we start by analyzing the logic LTL$_f$.

Theorem 42 *Satisfiability of $\langle T\rangle$-LTL$_f$ formulas without disjunctions, with $T \supseteq \{\mathsf{X}, \mathsf{G}, \mathsf{F}\}$ or $T \supseteq \{\mathsf{X_w}, \mathsf{F}\}$, is* NP*-hard.*

Proof. We exhibit a reduction from the ONE-IN-THREE POSITIVE 3-SAT problem (Garey & Johnson, 1979). Let $\phi = C_1 \wedge \cdots \wedge C_m$ be a propositional formula in conjunctive normal form over the set $\{x_1, ..., x_n\}$ of variables, where $C_j = x_{j_1} \vee x_{j_2} \vee x_{j_3}$, for each $j \in \{1, ..., m\}$. Based on ϕ, we build the $\langle\{\mathsf{X}, \mathsf{G}, \mathsf{F}\}\rangle$-LTL$_f$ formula $\varphi_\phi = \varphi_1 \wedge \cdots \wedge \varphi_m \wedge \psi$ over the set $\{x_1, ..., x_n\} \cup \{y_1, ..., y_m\}$ of variables such that $\varphi_j = \kappa'_j \wedge \kappa''_j \wedge \kappa'''_j \wedge \varrho_j$, for each $C_j = x_{j_1} \vee x_{j_2} \vee x_{j_3}$, where

- $\kappa'_j = y_j$;
- $\kappa''_j = \mathsf{X}^{\mathsf{2\times j-1}}(y_j \wedge \bigwedge_{j' \neq j} \neg y_{j'})$, where X^α denotes α repeated applications of X;
- $\kappa'''_j = \mathsf{X}^{\mathsf{2\times j}}(y_j \wedge \bigwedge_{j' \neq j} \neg y_{j'})$;
- $\varrho_j = \mathsf{F}(y_j \wedge x_{j_1} \wedge \neg x_{j_2} \wedge \neg x_{j_3}) \wedge \mathsf{F}(y_j \wedge \neg x_{j_1} \wedge x_{j_2} \wedge \neg x_{j_3}) \wedge \mathsf{F}(y_j \wedge \neg x_{j_1} \wedge \neg x_{j_2} \wedge x_{j_3})$;

and where $\psi = \mathsf{X}^{\mathsf{2\times m+1}}(\mathsf{G}(\bigwedge_{j=1}^m \neg y_j))$.

We now claim that: *there is a truth assignment over $\{x_1, ..., x_n\}$ such that, for each clause of ϕ, precisely one variable evaluates true if, and only if, φ_ϕ is satisfiable.*

(if part) Assume that π is a model of φ_ϕ. For each $j \in \{1, ..., m\}$, because of the subformula κ'_j which is required to hold in π at the initial time instant, we have that $\pi_0 \supseteq \{y_j\}$. Hence, $\pi_0 \supseteq \{y_1, ..., y_m\}$. Because of the subformulas κ''_j and κ'''_j, we have that $\pi_{2\times j-1} \cap \{y_1, ..., y_m\} = \pi_{2\times j} \cap \{y_1, ..., y_m\} = \{y_j\}$. Finally, because of the subformula ψ, for each $j' > 2 \times m$, we have $\pi_{j'} \cap \{y_1, ..., y_m\} = \emptyset$. Therefore, for each clause C_j, there are precisely three time instants where y_j evaluates true, and one of them is the initial time instant 0. Now, consider the subformula ϱ_j, and note that its three conjuncts built over F must be mapped to time instants where y_j holds. Hence, one of them must evaluate true in the initial time instant. This means that, for each clause C_j, $|\pi_0 \cap \{x_{j_1}, x_{j_2}, x_{j_3}\}| = 1$ holds. Therefore, the truth assignment σ for ϕ such that x_i, with $i \in \{1, ..., n\}$, evaluates true if, and only if, $x_i \in \pi_0$ holds is satisfying. In particular, for each clause of ϕ, precisely one variable evaluates true.

(only-if part) Consider a truth assignment σ such that, for each clause of φ_ϕ, precisely one variable evaluates true. Assume, without loss of generality, that for each clause $C_j = x_{j_1} \vee x_{j_2} \vee x_{j_3}$, the variable x_{j_1} evaluates true in σ, while x_{j_2} and x_{j_3} evaluate false. Consider the trace π with $len(\pi) = 2 \times m + 2$ built as follows: $\pi_0 = \{y_1, ..., y_m\} \cup \{x_{j_1} \mid j \in \{1, ..., m\}\}$; for each $j \in \{1, ..., m\}$, $\pi_{2\times j-1} = \{y_j, x_{j_2}\}$ and $\pi_{2\times j} = \{y_j, x_{j_3}\}$; $\pi_{2\times m+1} = \{\}$. It follows directly that $\pi \models \varphi_\phi$.

Because of the above property and since the reduction is feasible in polynomial time, we conclude that satisfiability of $\langle\{\mathsf{X}, \mathsf{G}, \mathsf{F}\}\rangle$-LTL$_f$ formulas without disjunction is NP-hard. In fact, minor modifications to the encoding can be used to show that satisfiability of $\langle\{\mathsf{X_w}, \mathsf{F}\}\rangle$-LTL$_f$ formulas without disjunction is NP-hard, too. Basically, given φ_ϕ, we build a formula φ'_ϕ by replacing each occurrence of X with $\mathsf{X_w}$, and by replacing the subformula $\psi = \mathsf{X}^{2\times m+1}(\mathsf{G}(\bigwedge_{j=1}^{m} \neg y_j))$ with ${\mathsf{X_w}}^{2\times m}(\mathsf{X_w}(y_1 \wedge \neg y_1))$. Note that all models of the resulting formula φ'_ϕ have length at most $2 \times m + 1$. By following the same line of reasoning as in the proof of the above claim, it can be then checked that there is a truth assignment over $\{x_1, ..., x_n\}$ such that, for each clause of ϕ, precisely one variable evaluates true if, and only if, φ'_ϕ is satisfiable. $\square$

Theorem 43 *Satisfiability of $\langle T\rangle$-LTL$_f$ formulas without disjunctions, with $T \supseteq \{\mathsf{U}\}$, is* NP-*hard.*

Proof. Consider the class of $\langle\{\mathsf{U}\}\rangle$-LTL$_f$ formulas without disjunction. We exhibit a reduction from the 3-SAT problem (Karp, 1972). Let $\phi = C_1 \wedge \cdots \wedge C_m$ be a propositional formula in conjunctive normal form over the set $\{x_1, ..., x_n\}$ of variables, where $C_j = w_{j_1} \vee w_{j_2} \vee w_{j_3}$, for each $j \in \{1, ..., m\}$, and w_{j_1},w_{j_2}, and w_{j_3} are literals. Consider the $\langle\{\mathsf{U}\}\rangle$-LTL$_f$ formula without disjunction φ_ϕ over the set $\{x_1, ..., x_n\} \cup \{end\}$ of variables such that $\varphi_\phi = \neg end \wedge \phi'$, where ϕ' is obtained from ϕ by rewriting each clause C_j as the formula $(w_{j_1}\ \mathsf{U}\ (w_{j_2}\ \mathsf{U}\ (w_{j_3}\ \mathsf{U}\ end)))$.

We claim that: *there is a truth assignment over $\{x_1, ..., x_n\}$ such that ϕ evaluates true if, and only if, φ_ϕ is satisfiable.*

(if part) Assume that π is a model of φ_ϕ. For each clause C_j, since the subformula $(w_{j_1}\ \mathsf{U}\ (w_{j_2}\ \mathsf{U}\ (w_{j_3}\ \mathsf{U}\ end)))$ is required to hold in π at the initial time instant and we have that $\pi \models \neg end$, we can derive that $\pi_0 \models w_{j_1} \vee w_{j_2} \vee w_{j_3}$. Therefore, the truth assignment σ for ϕ such that x_i, with $i \in \{1, ..., n\}$, evaluates true if, and only if, $x_i \in \pi_0$ holds is satisfying.

(only-if part) Assume that σ is a satisfying truth assignment for ϕ. Consider the trace π with $len(\pi) = 2$ built as follows: $\pi_0 = \{x_i \mid x_i \text{ evaluates true in } \sigma\}$ and $\pi_1 = \{end\}$. It is easy to check that $\pi \models \varphi_\phi$.

The result eventually follows since φ_ϕ can be built in polynomial time. $\square$

We now discuss hardness results for LTL$_p$ in absence of disjunction. Note that, given the syntactic fragment we are considering, it is not possible to exploit the results derived in Section 3. Moreover, note that, as usual, negation is immaterial on process traces.

Theorem 44 *Satisfiability of negation-free $\langle T\rangle$-LTL$_p$ formulas without disjunctions, with $T \supseteq \{\mathsf{X}, \mathsf{G}, \mathsf{F}\}$ or $T \supseteq \{\mathsf{X_w}, \mathsf{F}\}$, is* NP-*hard.*

Proof. NP-hardness is shown again by a reduction from the ONE-IN-THREE POSITIVE 3-SAT problem (Garey & Johnson, 1979). Let $\phi = C_1 \wedge \cdots \wedge C_m$ be a propositional formula in conjunctive

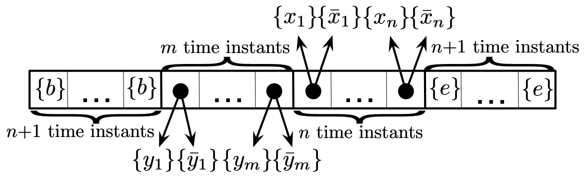


Figure 8: Example construction in the proof of Theorem 44.

normal form over the set $\{x_1, ..., x_n\}$ of variables. We adapt the reduction exhibited in the proof of Theorem 42: Based on ϕ, we build the negation-free $\langle\{\mathsf{X}, \mathsf{G}, \mathsf{F}\}\rangle$-LTL$_p$ formula without disjunction $\bar{\varphi}_\phi = \bar{\varphi}_1 \wedge \cdots \wedge \bar{\varphi}_m \wedge \bar{\psi}$ over the set $\{x_1, \bar{x}_1, ..., x_n, \bar{x}_n\} \cup \{y_1, \bar{y}_1, ..., y_m, \bar{y}_m\} \cup \{b, e\}$ of variables. In particular, for each $C_j = x_{j_1} \vee x_{j_2} \vee x_{j_3}$, we have that $\bar{\varphi}_j = \bar{\kappa}'_j \wedge \bar{\kappa}''_j \wedge \bar{\kappa}'''_j \wedge \bar{\varrho}_j$, where

- $\bar{\kappa}'_j = \bigwedge_{\alpha=0}^{n} \mathsf{X}^{\alpha}(b) \wedge \mathsf{X}^{\mathsf{n}+\alpha}(y_j) \wedge \bigwedge_{\alpha=0}^{n} \mathsf{X}^{2\times\mathsf{n}+\mathsf{m}+\alpha}(e)$;
- $\bar{\kappa}''_j = \mathsf{X}^{(2\times\mathsf{j}-1)\times(\mathsf{m}+3\times\mathsf{n}+2)}(\bar{\kappa}'_j \wedge \bigwedge_{j'\neq j} \mathsf{X}^{\mathsf{n}+\mathsf{j}'}(\bar{y}_j))$;
- $\bar{\kappa}'''_j = \mathsf{X}^{(2\times\mathsf{j})\times(\mathsf{m}+3\times\mathsf{n}+2)}(\bar{\kappa}'_j \wedge \bigwedge_{j'\neq j} \mathsf{X}^{\mathsf{n}+\mathsf{j}'}(\bar{y}_j))$;
- $$\begin{aligned}\varrho_j = \ & \mathsf{F}(\bar{\kappa}'_j \wedge \mathsf{X}^{\mathsf{m}+\mathsf{n}+1+\mathsf{j}_1}(x_{j_1}) \wedge \mathsf{X}^{\mathsf{m}+\mathsf{n}+1+\mathsf{j}_2}(\bar{x}_{j_2}) \wedge \mathsf{X}^{\mathsf{m}+\mathsf{n}+1+\mathsf{j}_3}(\bar{x}_{j_3}))\wedge \\ & \mathsf{F}(\bar{\kappa}'_j \wedge \mathsf{X}^{\mathsf{m}+\mathsf{n}+1+\mathsf{j}_1}(\bar{x}_{j_1}) \wedge \mathsf{X}^{\mathsf{m}+\mathsf{n}+1+\mathsf{j}_2}(x_{j_2}) \wedge \mathsf{X}^{\mathsf{m}+\mathsf{n}+1+\mathsf{j}_3}(\bar{x}_{j_3}))\wedge \\ & \mathsf{F}(\bar{\kappa}'_j \wedge \mathsf{X}^{\mathsf{m}+\mathsf{n}+1+\mathsf{j}_1}(\bar{x}_{j_1}) \wedge \mathsf{X}^{\mathsf{m}+\mathsf{n}+1+\mathsf{j}_2}(\bar{x}_{j_2}) \wedge \mathsf{X}^{\mathsf{m}+\mathsf{n}+1+\mathsf{j}_3}(x_{j_3})).\end{aligned}$$

and where $\bar{\psi} = \mathsf{X}^{(2\times\mathsf{m}+1)\times(\mathsf{m}+3\times\mathsf{n}+2)}(\mathsf{G}(b))$.

We claim that: *there is a truth assignment over $\{x_1, ..., x_n\}$ such that, for each clause of ϕ, precisely one variable evaluates true if, and only if, $\bar{\varphi}_\phi$ is satisfiable.*

(if part) Assume that π is a PT-model of $\bar{\varphi}_\phi$. Consider any clause C_j. Note that there are at least three time instants where the subformula $\bar{\kappa}'_j$ holds, namely, the initial time instant 0 plus $(2 \times j - 1) \times (m + 3 \times n + 2)$ and $(2 \times j) \times (m + 3 \times n + 2)$. In particular, because of κ''_j and κ'''_j, in the latter two time instants, it actually holds the subformula $\bar{\kappa}'_j \wedge \bigwedge_{j'\neq j} \mathsf{X}^{n+j'}(\bar{y}_j)$, which we denote hereinafter as $\bar{\zeta}_j$.

Let w be a time instant such that $\pi, w \models_p \bar{\zeta}_j$. In order to help the intuition, Figure 8 illustrates the structure of the time instants in the range $\{w, ..., w + m + 3 \times n + 2\}$, which are those playing a role in the inductive definition of satisfiability for the subformula $\bar{\zeta}_j$. Let $w' > w$ be another time instant such that $\pi, w' \models \bar{\zeta}_{j'}$ holds, for a clause $C_{j'}$ not necessarily different from C_j. Observe that, for each $i \in \{0, ..., n\}$, $\pi_{w+i} = \pi_{w'+i} = \{b\}$. Furthermore, $\pi_{w+n+j} = \{y_j\}$ and $b \notin \pi_{w+n+i}$ holds, for each $i \in \{1, ..., m\}$, because $\pi, w \models_p \bar{\zeta}_j$. By contrast, for each $i \in \{1, ..., m\} \setminus \{j\}$, $y_j \notin \pi_{w'+n+i}$ holds because $\pi, w' \models_p \bar{\zeta}_{j'}$. Thus, $w' \geq w+m+n+1$. Now, recall that $\pi_{w'+n} = \{b\}$. However, since $\pi, w \models_p \bar{\zeta}_j$, we know that $\pi_{w+m+2\times n+i} = \{e\}$, for each $i \in \{1, ..., n\}$. Therefore, we derive that $w' \geq w+m+2\times n+1$. Finally, by recalling again that $\pi_{w'} = \{b\}$, we actually derive $w' \geq w + m + 3 \times n + 2$.

Observe now that for each time instant $w' > (2 \times m + 1) \times (m + 3 \times n + 2)$, it holds that $\pi_{w'} = \{b\}$ because of $\bar{\Psi}$. Furthermore, observe that, for each $h \in \{1, ..., 2 \times m\}$,

$h \times (m + 3 \times n + 2)$ is a time instant where precisely one subformula having the form $\bar{\zeta}_j$ holds, and recall that for each clause C_j, there are two time instants of this kind where $\bar{\zeta}_j$ holds (for $h = 2 \times j - 1$ and $h = 2 \times j$). Combined with the inequality $w' \geq w + m + 3 \times n + 2$ relating any two time instants $w' > w$ over which subformulas $\bar{\zeta}_j$ and $\bar{\zeta}_{j'}$ hold for clauses not necessarily distinct, we conclude that, for each clause C_j, the subformula $\bar{\kappa}'_j$ holds only at the time instant 0 plus the two time instants $(2\times j-1)\times(m+3\times n+2)$ and $(2\times j)\times(m+3\times n+2)$.

Finally, for each clause C_j, consider the subformula $\bar{\varrho}_j$, and note that its three conjuncts built over F must be mapped to time instants where $\bar{\kappa}'_j$ holds. Hence, one of them must evaluate true in the initial time instant. This means that, for each clause C_j, there is precisely one variable, say without loss of generality x_{j_1}, such that $\pi_{m+n+1+j_1} = \{x_{j_1}\}$; and, $\pi_{m+n+1+j_2} = \{\bar{x}_{j_2}\}$ and $\pi_{m+n+1+j_3} = \{\bar{x}_{j_3}\}$. Therefore, the truth assignment σ for ϕ such that x_i, with $i \in \{1, ..., n\}$, evaluates true if, and only if, $x_i \in \pi_{m+n+1+i}$ holds is satisfying. In particular, for each clause of ϕ, precisely one variable evaluates true.

(only-if part) Consider a truth assignment σ such that, for each clause of φ_ϕ, precisely one variable evaluates true. Assume, without loss of generality, that for each clause $C_j = x_{j_1} \vee x_{j_2} \vee x_{j_3}$, the variable x_{j_1} evaluates true in σ, while x_{j_2} and x_{j_3} evaluate false. Consider the traces in the set $\{\bar{\pi}^0, ..., \bar{\pi}^{2\times m}\}$ all of them having length $m + 3 \times n + 2$ and such that:

- $\bar{\pi}^0 \models \bar{\kappa}'_1 \wedge \cdots \wedge \bar{\kappa}'_m$; and, for each $j \in \{1, ..., m\}$, $\bar{\pi}^0_{m+n+1+j_1} = \{x_{j_1}\}$; $\bar{\pi}^0_{m+n+1+j_2} = \{\bar{x}_{j_2}\}$ and $\bar{\pi}^0_{m+n+1+j_3} = \{\bar{x}_{j_3}\}$;
- for each $j \in \{1, ..., m\}$,
  - $\bar{\pi}^{2\times j-1} \models \bar{\kappa}'_j \wedge \bigwedge_{j' \neq j} \mathsf{X}^{\mathsf{n+j'}}(\bar{y}_j)$; $\bar{\pi}^{2\times j-1}_{m+n+1+j_2} = \{x_{j_2}\}$ and $\bar{\pi}^{2\times j}_{m+n+1+i} = \{\bar{x}_i\}$, for each $i \in \{1, ..., n\} \setminus \{j_2\}$;
  - $\bar{\pi}^{2\times j} \models \bar{\kappa}'_j \wedge \bigwedge_{j' \neq j} \mathsf{X}^{\mathsf{n+j'}}(\bar{y}_j)$; $\bar{\pi}^{2\times j}_{m+n+1+j_3} = \{x_{j_3}\}$ and $\bar{\pi}^{2\times j}_{m+n+1+i} = \{\bar{x}_i\}$, for each $i \in \{1, ..., n\} \setminus \{j_3\}$;

Note that the traces above are well-defined. In particular, $\bar{\pi}^0$ is well-defined because σ is a truth assignment such that, for each clause, precisely one variable evaluates true. Then, consider the trace π such that $len(\pi) = (2 \times m + 1) \times (m + 3 \times n + 2) + 1$, which is obtained by concatenating $\bar{\pi}^0, ..., \bar{\pi}^{2\times m}$ and where the last time instant is such that $\pi_{len(\pi)\text{-}1} = \{b\}$. We claim that $\pi \models_p \bar{\varphi}_\phi$. Indeed, by construction, for each clause C_j, we have:

- $\pi, 0 \models_p \bar{\kappa}'_j$,
- $\pi, (2 \times j - 1) \times (m + 3 \times n + 2) \models_p \bar{\kappa}'_j \wedge \bigwedge_{j' \neq j} \mathsf{X}^{\mathsf{n+j'}}(\bar{y}_j)$, and
- $\pi, (2 \times j) \times (m + 3 \times n + 2) \models_p \bar{\kappa}'_j \wedge \bigwedge_{j' \neq j} \mathsf{X}^{\mathsf{n+j'}}(\bar{y}_j)$.

Therefore, $\pi, 0 \models_p \bar{\kappa}'_j \wedge \bar{\kappa}''_j \wedge \bar{\kappa}'''_j$. In fact, the three conjuncts built over F in the subformula $\bar{\varrho}_j$ hold in π only respectively at these times instants, i.e., at 0, $(2 \times j - 1) \times (m + 3 \times n + 2)$, and $(2 \times j) \times (m + 3 \times n + 2)$. Therefore, $\pi, 0 \models_p \bar{\varrho}_j$. Finally, $\pi, 0 \models_p \bar{\Psi}$ follows because of the last time instant of π, and hence we conclude that $\bar{\varphi}_\phi$ is satisfiable.

Because of the above property and since the reduction is feasible in polynomial time, we conclude that satisfiability of negation-free $\langle\{\mathsf{X}, \mathsf{G}, \mathsf{F}\}\rangle$-LTL$_p$ formulas without disjunction is NP-hard. Minor modifications to the encoding can be used to show that satisfiability of negation-free

$\langle\{\mathsf{X_w}, \mathsf{F}\}\rangle$-LTL$_p$ formulas without disjunction is NP-hard, too. Basically, given φ_ϕ, we build a formula φ'_ϕ by replacing each occurrence of X with $\mathsf{X_w}$, and by replacing the subformula $\bar{\psi} = \mathsf{X}^{(2\times m+1)\times(m+3\times n+2)}(\mathsf{G}(b))$ with $\bar{\psi} =' {\mathsf{X_w}}^{(2\times m+1)\times(m+3\times n+2)-1}(\mathsf{X_w}(b \wedge e))$. Note that all models of the resulting formula φ'_ϕ have length at most $(2 \times m + 1) \times (m + 3 \times n + 2)$. By following the same line of reasoning as in the proof of the above claim, it can be then checked that there is a truth assignment over $\{x_1, ..., x_n\}$ such that, for each clause of φ, precisely one variable evaluates true if, and only if, φ'_ϕ is satisfiable. □

Below, we analyze the fragment for which only the U operator is allowed.

Theorem 45 *Satisfiability of negation-free $\langle T\rangle$-*LTL$_p$ *formulas without disjunctions, with $T \supseteq \{\mathsf{U}\}$, is* NP-*hard.*

Proof. We exhibit a reduction from the ONE-IN-THREE POSITIVE 3-SAT problem (Garey & Johnson, 1979). Let ϕ be a propositional formula in conjunctive normal form $\phi = C_1 \wedge \cdots \wedge C_m$ over the set $\{x_1, ..., x_n\}$ of variables, where each clause C_j has the form $C_j = x_{j_1} \vee x_{j_2} \vee x_{j_3}$ with $x_{j_1}, x_{j_2}, x_{j_3} \in \{x_1, ..., x_n\}$. Based on ϕ, we build a negation-free $\langle\{\mathsf{U}\}\rangle$-LTL$_p$ formula φ_ϕ, over the set $\{begin, end_1, x_1, ..., x_n, end_2\}$ of variables, as follows. For each $j \in \{1, ...m\}$ and $h \in \{1, 2, 3\}$, let $\mathcal{V}^{<}_{j,h} = \{begin, end_1, x_1, ..., x_{j_h}\}$ and $\mathcal{V}_{j,h} = \{begin, end_1, x_{j_h}\} \cup \{x_\alpha \mid \alpha \neq j_h$ and x_α does not occur in some clause with $x_{j_h}.\}$ Moreover, consider the following formulas:

- $\varrho_{j,h} = (begin \ \mathsf{U} \ (end_1 \ \mathsf{U} \ (x_1 \ \mathsf{U} \ (...\mathsf{U} \ x_{j_h}))))$, where in particular variables in the set $\{x_1, ..., x_{j_h}\}$ are nested within until operators according to the order induced by their indices;
- $\eta_{j,h} = begin \ \mathsf{U} \ (end_1 \ \mathsf{U} \ (\alpha_{\beta_1} \ \mathsf{U} \ (...\mathsf{U} \ \alpha_{\beta_k}(\mathsf{U} \ end_2)))))$, where in particular all variables in the set $\{\alpha_{\beta_1}, ..., \alpha_{\beta_k}\} = \mathcal{V}_{j,h} \setminus \{begin, end_1\}$ are nested within until operators according to the order induced by their indices ($\beta_1 < \beta_2 < \cdots < \beta_k$);

Then, consider the following formula

$$\varphi_\phi = begin \wedge (begin \ \mathsf{U} \ end_1) \wedge \bigwedge_{j=1}^{m} ((\varrho_{j,1} \wedge \eta_{j,1}) \ \mathsf{U} \ ((\varrho_{j,2} \wedge \eta_{j,2}) \ \mathsf{U} \ ((\varrho_{j,3} \wedge \eta_{j,3}) \ \mathsf{U} \ end_2))).$$

We claim that: *there is a truth assignment over $\{x_1, ..., x_n\}$ such that, for each clause of ϕ, precisely one variable evaluates true if, and only if, φ_ϕ is satisfiable.*

(if part) Assume that π is a PT-model of φ_ϕ. Let j be an index in $\{1, ..., m\}$ and note that there is an index $h(j) \in \{1, 2, 3\}$ such that $\pi, 0 \models_p \varrho_{j,h(j)} \wedge \eta_{j,h(j)}$. Indeed, $\pi_0 = \{begin\}$ while $\pi, 0 \models_p ((\varrho_{j,1} \wedge \eta_{j,1}) \ \mathsf{U} \ ((\varrho_{j,2} \wedge \eta_{j,2}) \ \mathsf{U} \ ((\varrho_{j,3} \wedge \eta_{j,3}) \ \mathsf{U} \ end_2)))$. Now, given the form of $\varrho_{j,h(j)}$, we are guaranteed about the existence of a time instant where $x_{j_{h(j)}}$ holds; let $k(j)$ be the first time instant of this kind. Consider then the set K of all such time instants defined over all possible clauses, i.e., $K = \{k(1), ..., k(m)\}$ and note that $|K| \leq m$, because in principle such instants are not necessarily pairwise distinct. Assume, w.l.o.g., that $k(1) \leq k(2) \leq \cdots \leq k(m)$. Note that all time instants $\ell \in \{0, ...k(m)\}$ are such that $\pi_\ell \neq \{end_2\}$. In particular, note that $\pi, 0 \models \varrho_{m,h(m)}$ implies that $j_{h(j)} \leq j'_{h(j')}$ if, and only if, $j \leq j'$.

Now, consider the truth assignment σ such that x_i, for each $i \in \{1, ..., n\}$, evaluates true in σ if, and only if, there is a time instant in $w_i \in \{k(1), ..., k(m)\}$ such that $\pi, w_i \models_p x_i$.

By construction, σ is trivially satisfying. Indeed, just observe that, for each $j \in \{1, ..., m\}$, $x_{j_{h(j)}}$ is a variable occurring in C_j and evaluating true in σ. Then, let j be an index in $\{1, ..., m\}$. We show that there is no variable in C_j, but $x_{j_{h(j)}}$, evaluating true in σ. Assume, by contradiction, that x_{j_α} is another variable in C_j evaluating true in σ. Hence, there is some time instant $k \in \{k(1), ..., k(m)\}$ with $\pi, k \models_p x_{j_\alpha}$. Consider the case where $k > k(j)$ and let j' be the clause such that $k = k(j')$. Note that $j'_{h(j')} > j_{h(j)}$ holds. This contradicts $\pi, 0 \models_p \eta_{j,h(j)}$, since this formula forces that the variable $x_{j'_{h(j')}}$ conflicting with $x_{j_{h(j)}}$ does not occur in some time instant following $k(j)$ and preceding the first occurrence of end_2. It remains to be considered the case where $k < k(j)$. Let j' be the clause such that $k = k(j')$; the variable $x_{j'_{h(j')}}$ (with $j'_{h(j')} < j_{h(j)}$) evaluates true in σ and $\pi, 0 \models_p \varrho_{j',h(j')} \wedge \eta_{j',h(j')}$. However, $\pi, 0 \models_p \eta_{j',h(j')}$ implies that $x_{j_{h(j)}}$ cannot occur in some time instant following $k(j')$ and preceding the first occurrence of end_2, which is impossible.

(only-if part) Consider any truth assignment σ such that, for each clause of ϕ, precisely one variable evaluates true. Let $\{x_{i_1}, ..., x_{i_q}\}$ be the set of all variables evaluating true in σ ordered such that $i_r < i_{r+1}$ for all $r \in \{1, ..., q\text{-}1\}$, and consider the finite trace π such that $len(\pi) = q{+}3$, $\pi_0 = \{begin\}$, $\pi_1 = \{end_1\}$ and $\pi_{q+2} = \{end_2\}$ and, for each $j \in \{1, ..., q\}$, $\pi_{r+1} = \{x_{i_r}\}$. In particular, note that π is a process trace. It can be easily checked that, for each $j \in \{1, ..., m\}$, the condition $\pi, 0 \models_p \varrho_{j,h}$ holds where x_{j_h} is the (unique) variable $x_{i_h} \in \{x_{j_1}, x_{j_2}, x_{j_3}\}$ occurring in C_j and evaluating true in σ. Moreover, note that if it were the case that $\pi, 0 \not\models_p \eta_{j,h}$, then there would be a time instant w such that $\pi_w = \{x_{j'_h}\}$, with $x_{l'_h} \in \{x_{j_1}, x_{j_2}, x_{j_3}\}$ and $h' \neq h$. But, this would entail that two variables, x_{j_h} and $x_{j_{h'}}$, belonging to the clause C_j evaluate true in σ, which is impossible. Therefore, $\pi \models_p \varphi_\phi$.

The NP-hardness of satisfiability of negation-free $\langle\{\mathsf{U}\}\rangle$-LTL$_p$ formulas without disjunction follows because φ_ϕ can be built in polynomial time. □

7. A Practical Reasoner for LTL on Finite and Process Traces

In this section we discuss a reasoner (for both LTL$_f$ and LTL$_p$) we built upon our findings, called LTL2SAT, and we illustrate results of its experimental comparison with Aalta (Li et al., 2014a), the only reasoner available in the literature that specifically targets LTL on finite traces (but not LTL$_p$ formulas), and NuSMV (Cimatti et al., 2002), a bounded model checker for LTL[6] (whose evaluation strategy is close in the spirit to the one LTL2SAT implements). For the evaluation, we used both a benchmark dataset already used by Li et al. and a dataset taken from the business process domain (hence, more specific for LTL$_p$).

7.1 Conceptual Architecture of LTL2SAT

LTL2SAT implements the algorithms discussed in the paper to target the tractable classes, plus a SAT rewriting technique sharing the spirit of (incremental) bounded LTL model checking (BMC) methods based on SAT rewritings (Biere et al., 2006). Hence, LTL2SAT can be conceptually classified as a BMC solver that, for several LTL$_f$ and LTL$_p$ fragments, implements early termination

6. For a comparison of Aalta and NuSMV on standard LTL satisfiability, the reader is referred to the work of Li, Yao, Pu, Zhang, and He (2014b).

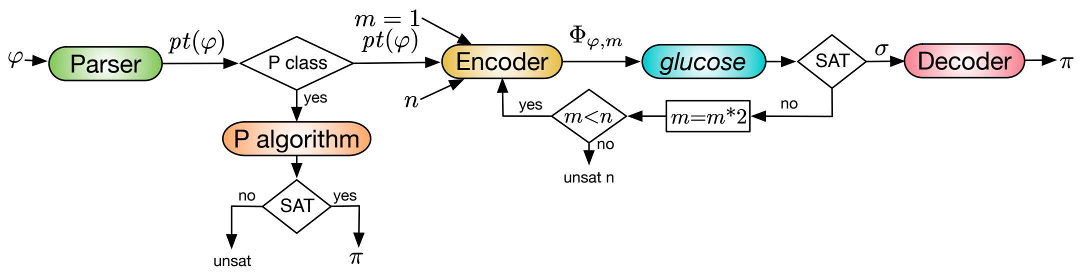


Figure 9: Architecture of the LTL2SAT reasoner.

conditions based on the linear-length model property (see Section 4). Moreover, as a distinguishing feature, it natively supports LTL$_p$ satisfiability checking via a specialized SAT rewriting that focuses on process traces only.

The architecture of LTL2SAT is reported in Figure 9. The first component is the *Parser* that builds the parse tree $pt(\varphi)$ associated to the input formula φ, which is then used to check whether φ belongs to a class for which the satisfiability checking emerged to be tractable in our analysis. If it is the case, then the corresponding polynomial algorithm is run. If not, then a SAT rewriting is applied. The SAT rewriting technique rewrites the input formula φ into an equivalent propositional formula $\Phi_{\varphi,m}$ that is satisfiable if, and only if, φ can be satisfied by a model of length at most m.

In order to check whether $\Phi_{\varphi,m}$ is satisfiable, LTL2SAT uses *glucose* (Audemard & Simon, 2009), a state-of-the-art SAT solver. In particular, $pt(\varphi)$ is given as input to the *Encoder* module that produces a SAT translation of the formula by initially considering a model length $m = 1$. Then every time no model of $\Phi_{\varphi,m}$ is found, m is doubled and the process is repeated until m exceeds the upper bound n given as input to the *Encoder*. This bound is set, by default, as the theoretical bound as studied in Section 4. If glucose finds a model σ for a SAT formula $\Phi_{\varphi,m}$, then the *Decoder* module translates σ into a trace π satisfying φ.

Finally, note that the user is allowed to modify the bound n in order to be less than the minimum theoretical one, so that LTL2SAT might also act as a sound, while not complete, solver. However, in our experimental campaign, n has been always set to the prescribed theoretical bound, so that LTL2SAT always reported correct answers. In fact, the correctness of the results provided by LTL2SAT has been checked on all the experiments we shall next describe: On the one hand, since LTL2SAT has been designed to return a model of each formula identified as satisfiable, for such formulas we simply run a checker and verified that the models returned by LTL2SAT were actually correct; on the other hand, manual inspection confirmed that no model exists for the formulas identified as unsatisfiable by LTL2SAT.

Finally, it is relevant to point out that, by exploiting the architecture discussed above, for the formulas on which LTL2SAT was unable to decide the satisfiability (in the given time limit we shall fix for running the experiments), LTL2SAT still provided us with the length up to which a satisfying trace does not exist—namely, the value of m reached in the last iteration performed before the termination. This is a feature that might be rather appealing in practical applications.

7.2 Encoding Approach

Let φ be a formula and let $pt(\varphi) = (V, E, \lambda)$ be its parse tree. In order to explain the encoding approach used by LTL2SAT to build $\Phi_{\varphi,m}$, we preliminary observe that, given a trace π, each node

$v \in V$ of the parse tree $pt(\varphi) = (V, E, \lambda)$ can be easily equipped with the set $sat(v, \pi)$ of all time instants where the subformula φ_v associated with the vertex v holds:[7]

- If $\lambda(v) = x$ or $\lambda(v) = \neg x$, then $sat(v, \pi) = \{i \mid \pi, i \models \lambda(v)\}$;
- If $\lambda(v) = \wedge$ (respectively, $\lambda(v) = \vee$), then $sat(v, \pi) = sat(v_1, \pi) \cap sat(v_2, \pi)$ (respectively, $sat(v, \pi) = sat(v_1, \pi) \cup sat(v_2, \pi)$) where v_1 and v_2 are the two children of v in $pt(\varphi)$;
- If $\lambda(v) = \mathsf{X}$, then $sat(v, \pi) = \{i\text{-}1 \mid i > 0 \text{ and } i \in sat(v', \pi)\}$ with v' being the only child of v in $pt(\varphi)$;
- If $\lambda(v) = \mathsf{X_w}$, then $sat(v, \pi) = \{len(\pi)\text{-}1\} \cup \{i\text{-}1 \mid i > 0 \text{ and } i \in sat(v', \pi)\}$ with v' being the only child of v in $pt(\varphi)$;
- If $\lambda(v) = \mathsf{G}$, then $sat(v, \pi) = \{j \in \{1, ..., len(\pi)\text{-}1\} \mid \forall i \in \{j, ..., len(\pi)\text{-}1\}, i \in sat(v', \pi)\}$ with v' being the only child of v in $pt(\varphi)$.
- If $\lambda(v) = \mathsf{F}$, then $sat(v, \pi) = \{j \mid \exists i \in sat(v', \pi)$ such that $0 \leq j \leq i\}$ with v' being the only child of v in $pt(\varphi)$;
- If $\lambda(v) = \mathsf{U}$, then $sat(v, \pi) = \{j \in \{1, ..., len(\pi)\text{-}1\} \mid j \in sat(v_2, \pi)$ or $\exists\, k \in \{j\text{+}1, ..., len(\pi)\text{-}1\}$ such that $k \in sat(v_2, \pi)$ and $i \in sat(v_1, \pi), \forall i \in \{j, ..., k\text{-}1\}\}$ where v_1 and v_2 are the two children of v in $pt(\varphi)$;
- If $\lambda(v) = \mathsf{R}$, then $sat(v, \pi) = \{j \in \{1, ..., len(\pi)\text{-}1\} \mid j \in sat(v_1, \pi) \cap sat(v_2, \pi)$ or $\exists\, k \in \{j\text{+}1, ..., len(\pi)\text{-}1\}$ such that $k \in sat(v_1, \pi) \cap sat(v_2, \pi)$ and $i \in sat(v_2, \pi), \forall i \in \{j, ..., k\text{-}1\}\}$ where v_1 and v_2 are the two children of v in $pt(\varphi)$;

Example 46 Consider again the LTL$_f$ formula φ discussed in Example 4. Its parse tree is reported in Figure 10, where each vertex v is also equipped with the set $sat(v, \pi)$ for the model π discussed in Example 5. Note that, for each vertex v, $sat(v, \pi)$ precisely consists of all time instants where the subformula associated with the parse tree rooted at v hold in π. $\lhd$

Now, the basic idea we used to built the formula $\Phi_{\varphi,m}$ is to mimic the construction of the sets $sat(v, \pi)$. Formally, we first define the variables $\ell[0], ..., \ell[m-1]$ which will be used to encode the last time instant of the model. Intuitively, $\Phi_{\varphi,m}$ will be defined in a way that there exists an index $i \in \{0, ..., m-1\}$, such that $\ell[j]$ evaluates true (respectively, false) whenever $j \geq i$ (respectively, $j < i$). The instant i is meant to denote the last time instant of a model of φ—more formally, whenever $\Phi_{\varphi,m}$ is satisfiable, there exists a model of φ having length $i \leq m$.

Then, we associate to each node v of $pt(\varphi)$ the variables $s_v[0], ..., s_v[m\text{-}1]$. Intuitively, $s_v[i]$ is meant to check whether the formula encoded in the parse tree rooted at v holds at the instant i. Moreover, in the expressions that follow, we use one further variable, $s_v[m]$, whose value will be actually a constant that will be fixed depending on the node v.

Finally, we set[8] $\Phi_{\varphi,m} = \bigwedge_{v \in V} \Phi_v \wedge s_{root}[0] \wedge \ell[m\text{-}1] \wedge \bigwedge_{i=0}^{m-2}(\ell[i] \rightarrow \ell[i\text{+}1])$ and, for each v, $\Phi_v = \bigwedge_{i=0}^{m-1}(s_v[i] \leftrightarrow \Phi_v^i)$, where:

7. Since the whole construction is feasible in polynomial time, we incidentally get a direct proof of Theorem 26. The crucial observation is that if r is the root of the parse tree, then it is easy to check that: $0 \in sat(r, \pi)$ holds if, and only if, π is a model of φ—if $|\pi_i| = 1$ holds, for all $i \in \{0, ..., len(\pi)\text{-}1\}$, then π is actually a PT-model.

8. Actually, $\Phi_{\varphi,m}$ is rewritten (in polynomial time) in *conjunctive normal form*, before it is passed to the solver.

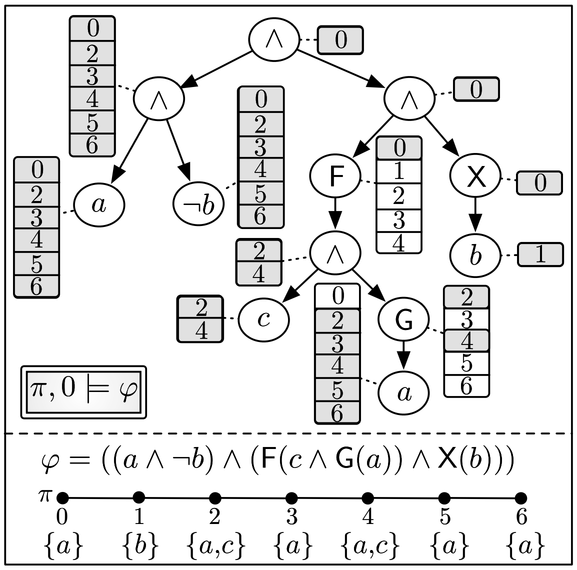

$\wedge$
0
$\wedge$
0
2
3
4
5
6
$\wedge$
0
F
0
1
2
3
4
X
0
a
0
2
3
4
5
6
$\neg b$
0
2
3
4
5
6
b
1
$\wedge$
2
4
c
2
4
G
2
3
4
5
6
a
0
2
3
4
5
6
$\pi, 0 \models \varphi$
$\varphi = ((a \wedge \neg b) \wedge (\mathsf{F}(c \wedge \mathsf{G}(a)) \wedge \mathsf{X}(b)))$
π
0 1 2 3 4 5 6
$\{a\}$ $\{b\}$ $\{a,c\}$ $\{a\}$ $\{a,c\}$ $\{a\}$ $\{a\}$

Figure 10: The parse tree of the LTL$_f$ formula φ (of Example 4). Given the model π, the set $sat(v, \pi)$ is reported next to the vertex v. Time instants that are propagated from each vertex to its parent are highlighted in gray.

- if $\lambda(v) \in \{x, \neg x\}$, then $\Phi_v^i = \bigwedge_{v' \in V, \lambda(v') \equiv \neg\lambda(v)} \neg s_{v'}[i] \wedge \bigwedge_{v' \in V, \lambda(v') \equiv \lambda(v)} s_{v'}[i]$;
- if $\lambda(v) = \wedge$ or $\lambda(v) = \vee$, then $\Phi_v^i = s_{v_1}[i]\lambda(v)s_{v_2}[i]$, where v_1 and v_2 are the left and right children of v;
- if $\lambda(v) = \mathsf{X}$ (respectively, $\lambda(v) = \mathsf{X_w}$), then $\Phi_v^i = s_{v'}[i{+}1] \wedge \neg\ell[i]$, where v' is the child of v and $s_v[m]$ is the constant `false` (respectively, `true`);
- if $\lambda(v) = \mathsf{F}$, then $\Phi_v^i = s_{v'}[i] \vee s_v[i{+}1] \wedge \neg\ell[i]$, where v' is the child of v and $s_v[m]$ is the constant `false`;
- if $\lambda(v) = \mathsf{G}$, then $\Phi_v^i = s_{v'}[i] \wedge (s_v[i{+}1] \vee \ell[i]))$, where v' is the child of v and $s_v[m]$ is the constant `true`;
- if $\lambda(v) = \mathsf{U}$, then $\Phi_v^i = s_{v_2}[i] \vee (s_{v_1}[i] \wedge s_v[i+1] \wedge \neg\ell[i])$, where v_1 and v_2 are the left and right children of v and $s_v[m]$ is the constant `false`;
- if $\lambda(v) = \mathsf{R}$, then $\Phi_v^i = (s_{v_1}[i] \wedge s_{v_2}[i]) \vee (s_{v_2}[i] \wedge (s_v[i+1] \vee \ell[i]))$, where v_1 and v_2 are the left and right children of v and $s_v[m]$ is the constant `true`.

By construction, the following can be established.

Theorem 47 *$\Phi_{\varphi,m}$ is satisfiable if, and only if, φ can be satisfied by a model π with $len(\pi) \leq m$.*

Note that with simple modifications in the above formulas we can focus the attention to PT-models only, by requiring that at each time instant precisely one variable evaluates true. In more

Dataset	#formulas	LTL$_f$ NP or P class	description
C1 (Rozier & Vardi, 2007, 2010, 2011)	999	*neg-free*$\langle\{\mathsf{G},\mathsf{F}\}\rangle$	$C_1(n) = \bigvee_{i=1}^{n} \mathsf{G}(\mathsf{F}\,p_i)$
C2 (Rozier & Vardi, 2007, 2010, 2011)	999	*neg-free*$\langle\{\mathsf{G},\mathsf{F}\}\rangle$	$C_2(n) = \bigwedge_{i=1}^{n} \mathsf{G}(\mathsf{F}\,p_i)$
E (Rozier & Vardi, 2007, 2010, 2011)	999	*neg-free*$\langle\{\mathsf{F}\}\rangle$	$E(n) = \bigwedge_{i=1}^{n} \mathsf{F}(p_i)$
Q (Rozier & Vardi, 2007, 2010, 2011)	249	*neg-free*$\langle\{\mathsf{G},\mathsf{F}\}\rangle$	$Q(n) = \bigwedge_{i=1}^{n}(\mathsf{F}\,p_i \vee \mathsf{G}\,p_{i+1})$
R (Rozier & Vardi, 2007, 2010, 2011)	249	*neg-free*$\langle\{\mathsf{G},\mathsf{F}\}\rangle$	$R(n) = \bigwedge_{i=1}^{n}(\mathsf{G}(\mathsf{F}\,p_i) \vee \mathsf{F}(\mathsf{G}\,p_{i+1}))$
S (Rozier & Vardi, 2007, 2010, 2011)	999	*neg-free*$\langle\{\mathsf{G}\}\rangle$	$Q(n) = \bigwedge_{i=1}^{n} \mathsf{G}\,p_i$
U (Rozier & Vardi, 2007, 2010, 2011)	999	*neg-free*$\langle\{\mathsf{U}\}\rangle$	$U(n) = (...(p_1\mathsf{U}p_2)\mathsf{U}...)\mathsf{U}p_n$
U2 (Rozier & Vardi, 2007, 2010, 2011)	999	*neg-free*$\langle\{\mathsf{U}\}\rangle$	$U_2(n) = p_1\mathsf{U}(p_2\mathsf{U}(...(p_{n-1}\mathsf{U}p_n)...))$
Counter (Rozier & Vardi, 2007, 2010, 2011)	20	-	Formulas describing n-bit binary counters
Rnd (Rozier & Vardi, 2007, 2010, 2011)	20000	-	Randomly-generated varying the number of variables and length of the formula
RndConj. (Rozier & Vardi, 2007, 2010, 2011)	9999	-	Randomly-generated as the conjunction of randomly selected small patterns
Acacia (Schuppan & Darmawan, 2011)	140	-	Arbiters and traffic light controllers
Alaska-lift (De Wulf et al., 2008)	272	-	Elevator specifications
Alaska-szy (De Wulf et al., 2008)	8	$\langle\{\mathsf{G},\mathsf{F}\}\rangle$	Mutual exclusion protocol
Anzu (Schuppan & Darmawan, 2011)	222	-	Buffer architecture
Forobots (Schuppan & Darmawan, 2011)	76	-	Model of a robot with properties
Schuppan-O1 (Schuppan & Darmawan, 2011)	54	-	Expon. behavior in some solvers
Schuppan-O2 (Schuppan & Darmawan, 2011)	54	$\langle\{\mathsf{G},\mathsf{F}\}\rangle$	Expon. behavior in some solvers
Schuppan-phltl (Schuppan & Darmawan, 2011)	36	-	Temporal variant of pigeonhole
Trp-N5x (Hustadt & Schmidt, 2002)	480	-	Obtained by lifting propositional CNF into fixed temporal structure
Trp-N5y (Hustadt & Schmidt, 2002)	280	-	Obtained by lifting propositional CNF into fixed temporal structure
Trp-N12x (Hustadt & Schmidt, 2002)	499	-	Obtained by lifting propositional CNF into fixed temporal structure
Trp-N12y (Hustadt & Schmidt, 2002)	380	-	Obtained by lifting propositional CNF into fixed temporal structure

Figure 11: Characteristics of the benchmark in Section 7.3.1.

details, in order for LTL2SAT to look only at PT-models, the encoding formula becomes $\Phi'_{\varphi,m} = \Phi_{\varphi,m} \wedge \bigwedge_{i=0}^{m-1} \Phi_i$ where, for each $i \in \{0,...,m-1\}$, the subformula $\Phi_i = \ell[i] \vee (\bigvee_{v\in V,\lambda(v)=x}(s_v[i] \rightarrow (\bigwedge_{v\in V,\lambda(v)=x',x'\neq x} \neg s_{v'}[i])) \wedge \bigvee_{v\in V,\lambda(v)=x} s_v[i])$ forces exactly one variable to evaluate true at each time instant. Indeed, LTL2SAT natively supports reasoning about LTL$_p$ formulas and it also supports negation that is not atomic (again with simple modifications to the above encoding).

7.3 Experimental Results

Performances of LTL2SAT have been compared to those of Aalta (Li et al., 2014a), which is the only existing reasoner that specifically targets the finite trace case, and those of NuSMV (Cimatti et al., 2002) implementing a bounded model checking strategy for LTL—in our experiments, NuSMV has been invoked with the `-bmc_length` option set to the theoretical bounds on the length of the models we obtain via our analysis (which of course are exploited in LTL2SAT, too).

Concerning the experiments on LTL$_f$, no pre/post-processing was needed with Aalta. Instead, NuSMV does not support this logic and in order to check the satisfiability of LTL$_f$ formulas φ, we invoked it to find counterexamples[9] to the LTL formula $\neg(\varphi')$ over a universal SMV model and where φ' is the encoding (from LTL$_f$ to LTL) discussed in the work of De Giacomo et al. (2014b). Hence, $\varphi' = t(\varphi) \wedge \neg end \wedge \mathsf{F}(end) \wedge \mathsf{G}(\neg end \vee \mathsf{G}(end)) \wedge \mathsf{G}(\neg end \vee \bigwedge_{x\in\mathcal{V}_\varphi} \neg x)$, while the transformation function t is such that:

9. Such counterexamples are infinite models π' of φ', and they one-to-one correspond to finite models π of φ. Indeed, π' is composed by the finite prefix π, plus an infinite suffix where the state $\{end\}$ is repeated indefinitely (cf. Claim 20).

$$\begin{array}{ll}
t(x) = x, \text{ for each } x \in \mathcal{V}_\varphi & t(\mathsf{F}(\varphi)) = \mathsf{F}(t(\varphi) \wedge \neg end) \\
t(\neg\varphi) = \neg t(\varphi) & t(\mathsf{G}(\varphi)) = \mathsf{G}(t(\varphi) \vee end) \\
t(\varphi_1 \wedge \varphi_2) = t(\varphi_1) \wedge t(\varphi_2) & t(\varphi_1 \mathsf{U} \varphi_2) = t(\varphi_1)\mathsf{U}(t(\varphi_2) \wedge \neg end) \\
t(\varphi_1 \vee \varphi_2) = t(\varphi_1) \vee t(\varphi_2) & t(\varphi_1 \mathsf{R} \varphi_2) = (t(\varphi_1) \wedge \neg end)\ \mathsf{R}\ (t(\varphi_2) \vee end) \\
t(\mathsf{X}(\varphi)) = \mathsf{X}(t(\varphi) \wedge \neg end) & t(\mathsf{X_w}(\varphi)) = \mathsf{X}(t(\varphi) \vee end)
\end{array}$$

Concerning the experiments on LTL$_p$, it must be observed that neither Aalta nor NuSMV natively support satisfiability over process traces. Accordingly, experiments have been performed by explicitly adding the constraint that at each time instant, precisely one activity must be executed. In particular, in the light of the arguments discussed in Section 3.1, for each formula φ, the formula $\varphi \wedge \mathsf{G}(\bigvee_{y \in \mathcal{V}_\varphi \cup \{p\}}(y \wedge \bigwedge_{y' \in \mathcal{V}_\varphi \cup \{p\} \setminus \{y\}} \neg y'))$ has been provided as input to Aalta, where p is an additional fresh variable not contained in $\mathcal{V}_\varphi$. Eventually, when dealing with NuSMV, the resulting formula is further processed as discussed above (with the ideas of De Giacomo et al., 2014b) to simulate the evaluation over finite models only.

Experiments have been carried out on a PC Intel Core i5 2,4 GHz, 8GB RAM and times are the average of five runs. Two groups of datasets have been considered, which are discussed below.

7.3.1 LTL$_f$ BENCHMARK

Description of the datasets. The first group of datasets we have considered is the one used by Li et al. (2014a), where Aalta has been shown to outperform LTL reasoners adapted to deal with finite traces (cf. Edelkamp, 2006; Gerevini et al., 2009; Pesic & van der Aalst, 2006a). The benchmark consists of 23 datasets with different characteristics. As summarized in Figure 11: *(i)* datasets C1, C2, E, Q, R, S, U, U2, Counter, Rnd and RndConj have been published and discussed by Rozier and Vardi (2007, 2010, 2011); *(ii)* Alaska-lift and Alaska-szy have been originally used by De Wulf et al. (2008); *(iii)* Acacia, Anzu, Forobots, Schuppan-O1, Schuppan-O2 and Schuppan-phltl have been posted by Schuppan and Darmawan (2011); *(iv)* Trp-N5x, Trp-N5y, Trp-N12x and Trp-N12y have been introduced by Hustadt and Schmidt (2002).

For all the benchmarks discussed above, we directly used the formulas posted and made available to the community by the authors. In particular, even for randomly generated benchmarks, we did not create novel datasets by using the generation procedure the authors have described. In fact, some of the datasets are randomly generated (i.e, C1, C2, E, Q, R, S, U, U2, Rnd, RndConj) and others encode the behavior or properties of some system (i.e., Counter, Acacia, Alaska-lift, Alaska-szy, Anzu, Forobots, Schuppan-O1, Schuppan-O2, Schuppan-phltl, Trp-N5x, Trp-N5y, Trp-N12x, Trp-N12y). In more detail, the characteristics of the benchmarks are illustrated in Figure 11, which reports for each dataset, its description, the number of formulas it contains, and the information about whether such formulas fall in one of the syntactic fragments we have considered in our analysis. Note that 10 out of 23 datasets (i.e., C1, C2, E, Q, R, S, U, U2, Alaska-szy and Shuppan-O2) consist of formulas belonging to classes enjoying the linear-length model property, and 8 of them (i.e., C1, C2, E, Q, R, S, U and U2) actually refer to classes for which satisfiability is tractable.

Comparison with Aalta. In a first series of experiments, we evaluated the datasets discussed above with respect to LTL$_f$ satisfiability by using both LTL2SAT and Aalta and by setting a timeout limit of 5 minutes for solving each instance. In more detail, for each formula, we compared the answer returned by LTL2SAT when checking LTL$_f$ satisfiability and that returned by Aalta. A summary of the result is reported in Figure 12.

Dataset	LTL2SAT and Aalta are comparable	LTL2SAT goes in timeout, while Aalta reports the corrrect answer
C1	100%	0%
C2	100%	0%
E	100%	0%
Q	100%	0%
R	100%	0%
S	100%	0%
U	100%	0%
U2	100%	0%
Counter	0%	100%
Rnd	92%	2%
RndConj.	86%	9%
Acacia	84%	8%
Alaska-lift	48%	26%
Alaska-szy	100%	0%
Anzu	58%	0%
Forobots	49%	51%
Schuppan-O1	55%	45%
Schuppan-O2	83%	17%
Schuppan-phltl	46%	51%
Trp-N5x	50%	34%
Trp-N5y	84%	16%
Trp-N12x	50%	39%
Trp-N12y	81%	19%
Total	90%	5%

Figure 12: Percentage of the type of results given by LTL2SAT and Aalta.

As it can be noted by inspecting the summary, LTL2SAT and Aalta provide the same correct answer over the $\sim$90% of the formulas by actually spending the same amount of time (up to 30ms of tolerance); that is, the two systems perform quite similarly on most of the given datasets and formulas. On the other hand, over the $\sim$5% of the formulas, LTL2SAT terminates in timeout while Aalta reports the correct answers, which emerged—in all the given formulas in this percentage—to be *unsat*. This is not by chance and it witnesses that some of the methods implemented in Aalta to identify unsatisfiable formulas (Li et al., 2014a) are rather effective in practice. Of course, LTL2SAT might well implement these methods (e.g., by just using Aalta as an external heuristic for unsatisfiability checking), hence becoming comparable to Aalta on these set of formulas, too—however, in order to shed lights on the intrinsic differences between the two systems, this strategy is not adopted and all results that are discussed below refer to LTL2SAT used in isolation. Finally, on the remaining formulas ($\sim$5%), the answers reported by the two systems are different and we do not discuss time comparison for them.

Comparison with NuSMV. In a second series of experiments, we used the datasets discussed above to compare the performances of LTL2SAT and NuSMV, again with respect to LTL$_f$ satisfiability. We run both tools on every formula in the dataset by using a timeout of 5 minutes. On these formulas (as well as on the other ones we shall introduce in the following section), NuSMV always reported correct answers—precisely as LTL2SAT (see again Section 7.1).

Overall, both systems terminated in timeout on the 6% of formulas, LTL2SAT was faster than NuSMV on the 20% of formulas, and the two tools were comparable on the 79% of instances. Given that the strategy of LTL2SAT is inspired by bounded model checking (and the fact that NuSMV is invoked with the same theoretical bounds we used for LTL2SAT), the better performances of LTL2SAT compared to those of NuSMV are likely due to its native support for LTL$_f$ and to the

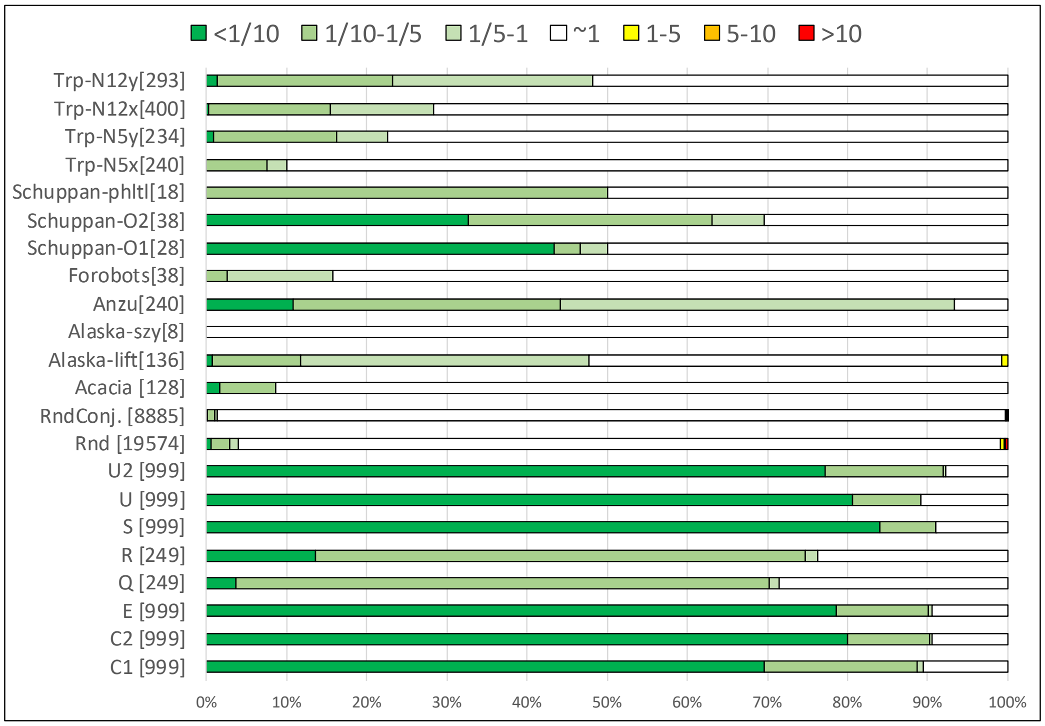


Figure 13: Ratio LTL2SAT/NuSMV on execution times, over the LTL$_f$ benchmark.

greater flexibility and generality of NuSMV (which can be used for LTL model checking) contrasted to the focused implementation of LTL2SAT tailored to LTL$_f$ and LTL$_p$ satisfiability (i.e., model checking universal models only). In particular, towards a fine-grained analysis, for each formula, we measured the ratio between the time required by LTL2SAT to check LTL$_f$ satisfiability and that required by NuSMV. The obtained results are reported in Figure 13.

The figure reports the results as percentage stacked bar charts for each dataset, where only the formulas for which at least one of the two systems produced a result are considered. White bars represent the percentages of instances where the two systems behave identically up to 30ms of tolerance, whereas bars on the left (resp., right) of the white bars are associated with ratio values less (resp., greater) than 1, so that LTL2SAT (resp., NuSMV) is faster. Note that the *Counter* dataset is not reported in the figure, as both systems reached the time-limit for all the formulas in it.

7.3.2 BUSINESS PROCESS MANAGEMENT DATASETS

Description of the setting. The second group of datasets we have considered is taken from the domain of *declarative* process management. This is an emerging area of research focusing on the definition of declarative approaches to process design, where logic-based languages are used to formalize a number of constraints each possible enactment of the process has to comply with. In these contexts, each enactment is associated with a finite trace where propositional variables transparently encode the activities of the underlying process.

The most noticeable example of these approaches is the DECLARE framework (Pesic & van der Aalst, 2006b), already introduced in Section 2.2. DECLARE constraints are reported in Figure 14.

Class	shorthand	fragment
Existence	$existence(x)$	*neg-free* $\langle\{\mathsf{F}\}\rangle$
	$existence(n, x)$	*neg-free*$\langle\{\mathsf{X}, \mathsf{F}\}\rangle$
	$absence(x)$	$\langle\{\mathsf{G}\}\rangle$
	$absence(n, x)$	$\langle\{\mathsf{X_w}, \mathsf{G}\}\rangle$
	$exactly(x)$	$\langle\{\mathsf{X_w}, \mathsf{G}, \mathsf{F}\}\rangle$
	$exactly(n, x)$	$\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}\}\rangle$
	$init(x)$	$\langle\emptyset\rangle$
Choice	$choice(x, y)$	*neg-free* $\langle\{\mathsf{F}\}\rangle$
	$exlusive\text{-}choice(x, y)$	$\langle\{\mathsf{G}, \mathsf{F}\}\rangle$
Relation	$responded\text{-}existence(x, y)$	$\langle\{\mathsf{G}, \mathsf{F}\}\rangle$
	$coexistence(x, y)$	$\langle\{\mathsf{G}, \mathsf{F}\}\rangle$
	$response(x, y)$	$\langle\{\mathsf{G}, \mathsf{F}\}\rangle$
	$precedence(x, y)$	$\langle\{\mathsf{G}, \mathsf{U}\}\rangle$
	$succession(x, y)$	$\langle\{\mathsf{G}, \mathsf{F}, \mathsf{U}\}\rangle$
	$alternate\text{-}response(x, y)$	$\langle\{\mathsf{X}, \mathsf{G}, \mathsf{U}\}\rangle$
	$alternate\text{-}precedence(x, y)$	$\langle\{\mathsf{X}, \mathsf{G}, \mathsf{U}\}\rangle$
	$alternate\text{-}succession(x, y)$	$\langle\{\mathsf{X}, \mathsf{G}, \mathsf{U}\}\rangle$
	$chain\text{-}response(x, y)$	$\langle\{\mathsf{X}, \mathsf{G}\}\rangle$
	$chain\text{-}precedence(x, y)$	$\langle\{\mathsf{X_w}, \mathsf{G}\}\rangle$
	$chain\text{-}succession(x, y)$	$\langle\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}\}\rangle$
Negation	$not\text{-}coexistence(x, y)$	$\langle\{\mathsf{G}\}\rangle$
	$neg\text{-}succession(x, y)$	$\langle\{\mathsf{G}\}\rangle$
	$neg\text{-}chain\text{-}succession(x, y)$	$\langle\{\mathsf{X_w}, \mathsf{G}\}\rangle$

Figure 14: DECLARE constraints organized in classes according to the specifications discussed by Maggi et al. (2011b). In the formulas, x and y stand for any pair of activities.

Dataset	#formulas	description
Loan Application	4	4 variants of a simple loan application process
Review Example	1	Event log of a reviewing process for a journal
Artificial	1	Event log generated randomly from 30 activities
isbpm2013	1	Benchmark dataset to test conformance algorithms
book	26	Benchmark logs provided with the ProM framework
Business Process Drift	72	A benchmark used to test concept drift in loan application
Digital Photo	1	Event log generated by simulation of a digital photo copier
bpm2013 benchmark	4	Proposed for testing Conformance Checking
Bank Transaction-BkT	2	Event logs of transaction bank processes
	112	

Figure 15: Characteristics of the benchmark in Section 7.3.2.

In particular, they are grouped into four classes: *existence* constraints, stating that some activity must/cannot be executed (a certain amount of times); *choice* constraints, modeling choice of execution; *relation* constraints, modeling that whenever the source activity is executed, then the target activity must also be executed (possibly with additional requirements); *negation* constraints, modeling that whenever the source activity is executed, then the target activity cannot be executed (possibly with additional restrictions). Each constraint has been formalized in the literature in terms of an LTL$_p$ formula, so that a set of DECLARE constraints gives rise to a formula obtained as the conjunction of the basic formulas associated with them. While for the encoding associated to the various constraints we refer to the work by Pesic and van der Aalst (2006b), we stress here that such formulas are not given in the so-called separated normal form (Fisher, 1991) (and, transforming a whole specification in this form might well take exponential time—see Section 1.1).

Dataset	LTL2SAT (LTL$_f$)	Aalta (LTL$_f$)	LTL2SAT (LTL$_p$)	Aalta (LTL$_p$)
Loan Application	4	0	4	0
Review Example	1	0	1	0
Artificial	1	1	1	0
isbpm2013	0	0	1	0
book	26	0	26	0
Business Process Drift	72	0	72	0
Digital Photo	1	0	1	0
bpm2013 benchmark	4	0	4	0
Bank Transaction-BkT	2	0	2	0
	112	1	112	0

Figure 16: Number of formulas over which the system are capable to find a model with the time-limit of 5 minutes.

The datasets of DECLARE formulas used in our experimentation have been obtained by mining all DECLARE constraints that hold on a number of logs made available by the IEEE Task-Force on Process Mining, an organization aiming at promoting the research, development, education and understanding of *process mining* (see `http://datacentrum.3tu.nl/`). In more detail, these logs consist of process traces taken from some specific application domain. On each process trace, we used the open libraries available in the well-known ProM platform (van Dongen, de Medeiros, Verbeek, Weijters, & van der Aalst, 2005) to generate the set of DECLARE constraints that hold on it (see, e.g., Maggi, 2013; Maggi, Mooij, & van der Aalst, 2011). At a conceptual level,[10] such constraints can be discovered *(i)* by first instantiating the patters shown in Figure 14 with all possible names of activities occurring in the process trace given at hand and *(ii)* by subsequently checking whether the instantiated patters actually hold on that process trace. The conjunction of all the instantiated patters that hold on the process trace is then returned as output. Note that, by construction, this conjunction is an LTL$_p$ formula that is satisfied precisely by the process trace from which it has been mined. Therefore, by processing the 112 datasets we have considered in the logs made available by the IEEE Task-Force on Process Mining, we were capable of producing 112 LTL$_p$ *satisfiable* formulas, which we used for our experimental activity. Figure 15 reports some statistics on the datasets.

Comparison with Aalta. Over the DECLARE dataset of satisfiable formulas discussed above, we focused on LTL$_f$ satisfiability checking and compared the performances of LTL2SAT and Aalta. Results are summarized in the second and third column of Figure 16 in terms of the number of formulas for each dataset for which the tools correctly identified the existence of a finite model within the timeout of 5 minutes. Then, we used LTL2SAT to check the LTL$_p$ satisfiability of each formula and results are reported in the fourth column of Figure 16.

Experiments related to LTL$_p$ satisfiability have been repeated over Aalta. From the results, it clearly emerges that LTL2SAT outperforms Aalta on all the datasets. Indeed, while LTL2SAT was able to find a model for all the 112 formulas, Aalta was able just to correctly identify 1 satisfiable formula for the LTL$_f$ case and no satisfiable formulas for the LTL$_p$ case.

Comparison with NuSMV. The DECLARE dataset discussed above has been also used to compare LTL2SAT and NuSMV over LTL$_p$ formulas. The results are reported in Figure 17.

10. Practical mining algorithms exploit ad-hoc pruning and inference methods in order to speed-up the computation.

	DECLARE formulas
$< 1/10$	71
$1/10 - 1/5$	15
$1/5 - 1$	26
1	0
$1 - 5$	0
$5 - 10$	0
> 10	0
	112

Figure 17: Ratio LTL2SAT/NuSMV, on execution time on the DECLARE LTL$_p$ benchmark.

Datasets	#formulas	LTL$_f$ sat faster	LTL$_f$ sat faster %	LTL$_p$ sat faster	LTL$_p$ sat faster %	Comparable	Comparable %
Loan Application	4	0	0%	0	0%	4	100%
Review Example	1	0	0%	0	0%	1	100%
Artificial	1	0	0%	0	0%	1	100%
isbpm2013	1	1	100%	0	0%	0	0%
book	26	0	0%	2	7%	24	93%
Business Process Drift	72	1	1%	0	0%	71	99%
Digital Photo	1	1	100%	0	0%	0	0%
bpm2013 benchmark	4	4	100%	0	0%	0	0%
Bank Transaction-BkT	2	1	50%	1	50%	0	0%
	112	8	7%	3	3%	101	90%

Figure 18: Statistics on execution time of LTL$_f$ and LTL$_p$ satisfiability checking.

As it can be noticed, LTL2SAT is faster than NuSMV on all the 112 formulas considered in our experimentation. In particular, LTL2SAT was able to correctly identify as satisfiable all the formulas in the DECLARE dataset, while NuSMV terminated in timeout on 5 formulas—as usual, the timeout limit was fixed in 5 minutes. Moreover, it is significant to observe that in the 63% of the formulas LTL2SAT emerged to be at least 10 times faster than NuSMV.

LTL$_f$ vs LTL$_p$. In our subsequent analysis and in the light of the above findings, we performed further experiments on LTL2SAT only. In particular, we first compared the behavior of the system when it reasons about LTL$_p$ and when it reasons about LTL$_f$. From Figure 16, it can be easily noted that LTL2SAT finds a model for all the formulas under both semantics. As summarized in Figure 18, timings for LTL$_f$ satisfiability checking are better than timings for LTL$_p$ satisfiability checking in $\sim$7% of the formulas, while the performance of LTL2SAT for two semantics are comparable in the $\sim$90% of the formulas. From these results, we can conclude that the overhead to reason on process traces is negligible in most of the cases. This comes with no surprise, as the careful reader might have already checked that the formulas discussed in Section 7.2 can be smoothly adapted for LTL$_p$, without any significant blow-up of the encoding.

Unsatisfiable formulas. In a further set of experiments, we investigated the behavior of our tool LTL2SAT when dealing with unsatisfiable formulas. Differently from Section 7.3.1 where we dealt with arbitrary formulas, we now focus on formulas belonging to some P or NP fragment. To this end, starting from the DECLARE constraints (see Table 14), we created a synthetic dataset containing 2110 unsatisfiable formulas belonging to the fragments $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$ (1050 formulas) and $\langle\{\mathsf{X_w}, \mathsf{G}\}\rangle$ (1050 formulas). For the generation of the dataset, we considered a number of variables that varies between 10 and 100 (with step 10) and a number of constraints that varies between 50 and 1000 (with step 50). In particular, for each combination 'number n of variables, number

m of constraints', we generated 5 formulas in the class (n, m) by inserting the two constraints $\mathit{existence}(x)$ (respectively, $\mathit{init}(x)$) and $\mathit{absence}(x)$ over the same variable x (that make the formula unsatisfiable) for the class $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$ (respectively, $\langle\{\mathsf{X_w}, \mathsf{G}\}\rangle$) and by selecting randomly the DECLARE constraint type of the m-2 remaining constraints and by picking randomly, from the pool of n variables, the variables to instantiate it. For the fragment $\langle\{\mathsf{X_w}, \mathsf{G}\}\rangle$ we considered the constraint types: $\mathit{absence}(x)$, $\mathit{absence}(2, x)$, $\mathit{absence}(3, x)$, $\mathit{init}(x)$, $\mathit{chain\text{-}precedence}(x, y)$, $\mathit{not\text{-}coexistence}(x, y)$, $\mathit{neg\text{-}succession}(x, y)$ and $\mathit{neg\text{-}chain\text{-}succession}(x, y)$. Instead, for the fragment $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$ we considered the set of constraint types: $\mathit{existence}(x)$, $\mathit{absence}(x)$, $\mathit{init}(x)$, $\mathit{choice}(x, y)$, $\mathit{exlusive\text{-}choice}(x, y)$, $\mathit{coexistence}(x, y)$, $\mathit{response}(x, y)$, $\mathit{not\text{-}\ coexistence}(x, y)$, $\mathit{responded\text{-}existence}(x, y)$, $\mathit{neg\text{-}succession}(x, y)$. In the experiments, we measured the time required by LTL2SAT to check the LTL$_p$ satisfiability of each formula and computed the average time over all the formulas in the same fragment having the same number of variables and the same number of constraints. The results are shown in Figure 19 as multiple line graphs. In particular, Figures 19 (a) and 19 (c) report the time required to solve $\langle\{\mathsf{X_w}, \mathsf{G}\}\rangle$ and $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$ formulas, respectively, where the data series correspond to the number of variables and the horizontal axis reports the number of constraints. On the contrary, Figures 19 (b) and 19 (d) report the time required to solve $\langle\{\mathsf{X_w}, \mathsf{G}\}\rangle$ and $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$ formulas, respectively, where the data series correspond to the number of constraints, and the number of variables is reported on the horizontal axis. As it can be noted, the time required by LTL2SAT to check satisfiability of the formulas in the P fragment (i.e., the $\langle\{\mathsf{X_w}, \mathsf{G}\}\rangle$ fragment) is less than 120 ms (see Figures 19 (a) and 19 (b)). The time required to check the satisfiability of the formulas in the NP fragment (i.e., the $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$ fragment) is higher but LTL2SAT is able to identify all the formulas as unsatisfiable except for few cases (i.e., when the number of variables is 100 and the number of constraints is higher than 800) where it terminates in timeout (see Figures 19 (c) and 19 (d)). In all the cases, it can be noted that the time required by LTL2SAT for LTL$_p$ satisfiability checking grows with the number of variables or constraints. Note, however, that the system exhibited a rather nice scaling, with the scaling for the P fragment being (as expected) better than the one exhibited for the NP fragment.

To gain some further insight on the behavior of the system in the unsatisfiable cases, we performed some additional analysis by considering formulas belonging to the NP-fragment $\langle\{\mathsf{G}, \mathsf{F}\}\rangle$. In particular, for the formulas belonging to the classes $(10, 50)$ (i.e., 10 variables and 50 constraints), $(10, 250)$, $(10, 500)$, $(10, 800)$ and $(10, 1000)$, we run LTL2SAT to check LTL$_p$ satisfiability of each formula φ several times, each time imposing a different fixed model length m. Actually, for these tests, we performed the first run by setting $m = 1$ and we doubled the length m at each of the subsequent runs, without stopping the process when m exceeded the theoretical bound on the length, but only when the timeout of 600 seconds has been reached in some run.

Results are depicted in Figure 20, where time is reported in seconds on the y-axis and the length is reported on the x-axis in logarithmic scale. The vertical dashed line in each chart represents the theoretical model length computed according to Theorem 19 by exploiting the peculiarity of the given fragment. On the other hand, the theoretical exponential bound prescribed by the general analysis of Theorem 23 is not reported, since it is much bigger than the length at which the timeout of 600 seconds is reached. For instance, for the class $(10, 50)$ the model length computed according to Theorem 19 is 109, while the theoretical upper bound is $\sim 10^{98}$ since the length of the formula is 327 and the timeout is reached at length $\sim 10^6$. By looking at the figures, it clearly emerges that our analysis focused on the identification of fragments enjoying favorable computational properties leaded to a practical strategy for unsatisfiability checking over them.

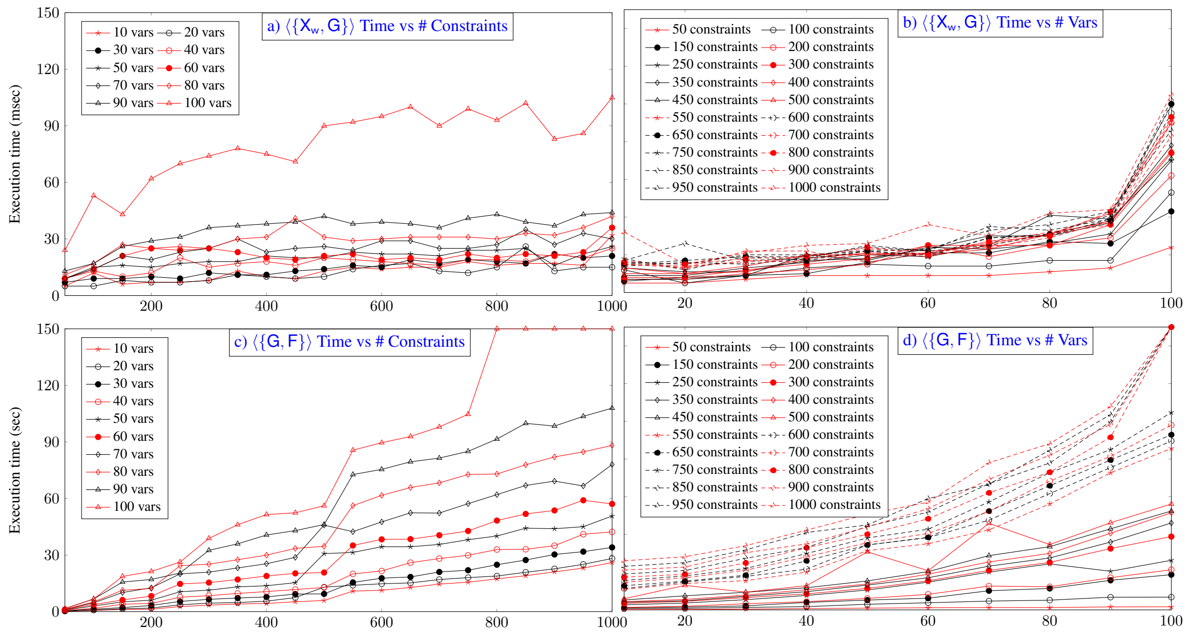


Figure 19: Multiple line graphs: execution times of LTL2SAT over process traces on the unsatisfiable benchmark datasets.

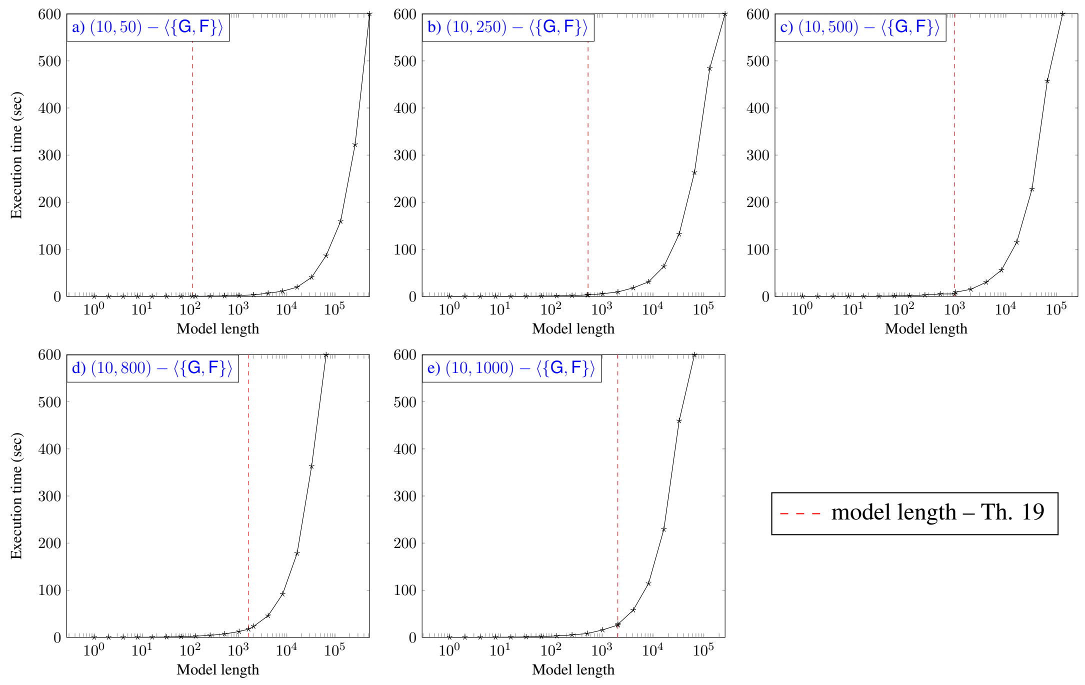


Figure 20: Execution times of LTL2SAT over process traces on different formulas for different model lengths.

8. Conclusion

We have studied satisfiability and model checking problems for LTL over finite traces, by providing a complete picture of their computational complexity over all the possible syntactic classes obtained by considering restrictions on the temporal operators (from $\{\mathsf{X}, \mathsf{X_w}, \mathsf{G}, \mathsf{F}, \mathsf{U}\}$) and by considering also negation-free formulas. We have also considered a semantic variant, where attention is focused on PT-models, i.e., on models where at each time instant, exactly one distinct propositional variable evaluates true. All algorithms and techniques proposed in the paper have been implemented and integrated into a prototype system, leveraging state-of-the-art SAT solvers.

Our results pave the way for further investigations. First, since our analysis has considered the temporal operators X, $\mathsf{X_w}$, G, F, U, and R looking at the future, a natural avenue of further research is to analyze the complexity when other temporal operators looking at the past are allowed, such as *since* and *has always been*. From the practical viewpoint, our results related to LTL_p formulas might be applied in the context of the specification and verification of processes and in the context of process mining, especially within frameworks for constraint-based representation of processes (Pesic & van der Aalst, 2006b). This is an active area of research where different kinds of formal methods have already been proven to be effective, including planning methods (Hoffmann, Weber, & Kraft, 2012). In this context, our prototype system can be used to provide operational decision support to running business processes and to check on-the-fly whether they comply with constraints and rules (as done by De Giacomo, De Masellis, Grasso, Maggi, & Montali, 2014a). Furthermore, following the perspective of Greco, Guzzo, Lupia, and Pontieri (2014), the prototype can support process mining tasks, in settings where learning methods (automatically derive a process model that can explain all the episodes recorded in an event log) can benefit of background knowledge expressed in LTL_f.

Acknowledgments

We thank the anonymous referees and the Associate Editor for their very useful comments, which helped us to significantly improve the quality of the paper.

References

Alur, R., & Henzinger, T. A. (1990). Real-time Logics: Complexity and Expressiveness. In *Proc. of LISCS*, pp. 390–401.

Artale, A., Kontchakov, R., Ryzhikov, V., & Zakharyaschev, M. (2013). The Complexity of Clausal Fragments of LTL. In *Proc. of LPAR*, pp. 35–52.

Audemard, G., & Simon, L. (2009). Predicting Learnt Clauses Quality in Modern SAT Solvers. In *Proc. of IJCAI*, pp. 399–404.

Bacchus, F., & Kabanza, F. (1998). Planning for Temporally Extended Goals. *Annals of Mathematics and Artificial Intelligence*, *22*(1-2), 5–27.

Bacchus, F., & Kabanza, F. (2000). Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, *116*(1-2), 123–191.

Baier, J. A., & McIlraith, S. A. (2006). Planning with First-Order Temporally Extended Goals using Heuristic Search. In *Proc. of AAAI*, pp. 788–795.

Bauland, M., Hemaspaandra, E., Schnoor, H., & Schnoor, I. (2006). Generalized Modal Satisfiability. In *Proc. of STACS*, pp. 500–511.

Bauland, M., Schneider, T., Schnoor, H., Schnoor, I., & Vollmer, H. (2009). The Complexity of Generalized Satisfiability for Linear Temporal Logic. *Logical Methods in Computer Science*, *5*(1), 1.

Bienvenu, M., Fritz, C., & McIlraith, S. A. (2006). Planning with Qualitative Temporal Preferences. In *Proc. of KR*, pp. 134–144.

Bienvenu, M., Fritz, C., & McIlraith, S. A. (2011). Specifying and computing preferred plans. *Artificial Intelligence*, *175*(7-8), 1308–1345.

Biere, A., Heljanko, K., Junttila, T., Latvala, T., & Schuppan, V. (2006). Linear Encodings of Bounded LTL Model Checking. *Logical Methods in Computer Science*, *2*(5), 1–64.

Bobadilla, L., Sanchez, O., Czarnowski, J., Gossman, K., & LaValle, S. (2011). Controlling Wild Bodies Using Linear Temporal Logic. In *Proc. of RSS*, pp. 17–24.

Bylander, T. (1994). The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence*, *69*(1-2), 165–204.

Calvanese, D., De Giacomo, G., & Vardi, M. Y. (2002). Reasoning about Actions and Planning in LTL Action Theories. In *Proc. of KR*, pp. 593–602.

Chen, C.-C., & Lin, I.-P. (1993). The Computational Complexity of Satisfiability of Temporal Horn Formulas in Propositional Linear-Time Temporal Logic. *Information Processing Letters*, *45*(3), 131–136.

Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., & Tacchella, A. (2002). Nusmv 2: An opensource tool for symbolic model checking. In *Proc. of CAV*, pp. 359–364. Springer.

Clarke, E. M., & Schlingloff, B.-H. (2001). Model Ckecking. In Robinson, A., & Voronkov, A. (Eds.), *Handbook of Automated Reasoning*, Vol. 2, chap. 24, pp. 1635–1790. Elsevier Science.

De Giacomo, G., De Masellis, R., Grasso, M., Maggi, F. M., & Montali, M. (2014a). Monitoring Business Metaconstraints Based on LTL and LDL for Finite Traces. In *Proc. of BPM*, pp. 1–14.

De Giacomo, G., De Masellis, R., & Montali, M. (2014b). Reasoning on LTL on Finite Traces: Insensitivity to Infiniteness. In *Proc. of AAAI*, pp. 1027–1033.

De Giacomo, G., & Vardi, M. Y. (1999). Automata-Theoretic Approach to Planning for Temporally Extended Goals. In *Proc. of ECP*, pp. 226–238.

De Giacomo, G., & Vardi, M. Y. (2013). Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *Proc. of IJCAI*, pp. 854–860.

De Wulf, M., Doyen, L., Maquet, N., & Raskin, J.-F. (2008). Antichains: Alternative algorithms for LTL satisfiability and model-checking. In *Proc. of TACAS*, Vol. 8, pp. 63–77. Springer.

Demri, S., & Schnoebelen, P. (2002). The Complexity of Propositional Linear Temporal Logics in Simple Cases. *Information and Computation*, *174*(1), 84–103.

Di Ciccio, C., Marrella, A., & Russo, A. (2015). Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary approaches. *Journal on Data Semantics*, *4*(1), 29–57.

Ding, X., Smith, S., Belta, C., & Rus, D. (2014). Optimal Control of Markov Decision Processes With Linear Temporal Logic Constraints. *IEEE Transactions on Automatic Control*, *59*(5), 1244–1257.

Dixon, C., Fisher, M., & Konev, B. (2007). Tractable Temporal Reasoning. In *Proc. of IJCAI*, pp. 318–323.

Dixon, C., Konev, B., Fisher, M., & Nietiadi, S. (2013). Deductive temporal reasoning with constraints. *Journal of Applied Logic*, *11*(1), 30–51.

Edelkamp, S. (2006). On the Compilation of Plan Constraints and Preferences. In *Proc. of ICAPS*, pp. 374–377.

Fagin, R., Halpern, J. Y., Moses, Y., & Vardi, M. Y. (1995). *Reasoning about Knowledge*. MIT Press.

Fionda, V., & Greco, G. (2016). The Complexity of LTL on Finite Traces: Hard and Easy Fragments. In *Proc. of AAAI*, pp. 971–977.

Fisher, M. (1991). A Resolution Method for Temporal Logic. In *Proc. of IJCAI*, pp. 99–104.

Gabelaia, D., Kontchakov, R., Kurucz, A., Wolter, F., & Zakharyaschev, M. (2005). Combining spatial and temporal logics: Expressiveness vs. complexity. *Journal of Artificial Intelligence Research*, *23*, 167–243.

Garey, M., & Johnson, D. (1979). *Computers and Intractability - A guide to the Theory of NP-Completeness*. Freeman.

Gerevini, A., Haslum, P., Long, D., Saetti, A., & Dimopoulos, Y. (2009). Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, *173*(5-6), 619–668.

Greco, G., Guzzo, A., Lupia, F., & Pontieri, L. (2014). Process Discovery under Precedence Constraints. *ACM Transactions on Knowledge Discovery from Data*, *9*(4), 32.

Halpern, J. Y. (1995). The Effect of Bounding the Number of Primitive Propositions and the Depth of Nesting on the Complexity of Modal Logic. *Artificial Intelligence*, *75*(2), 361–372.

Hemaspaandra, E. (2001). The complexity of poor man's logic. *Journal of Logic and Computation*, *11*(4), 609—622.

Hoffmann, J., Weber, I., & Kraft, F. M. (2012). SAP speaks PDDL: Exploiting a software-engineering model for planning in business process management. *Journal of Artificial Intelligence Research*, *44*, 587–632.

Hustadt, U., & Schmidt, R. A. (2002). Scientific benchmarking with temporal logic decision procedures. In *Proc. of KR*, Vol. 2, pp. 533–546.

IEEE Task Force on Process Mining (2011). Process Mining Manifesto. In *Proc. of BPM Workshops*, Vol. 99, pp. 169–194.

Karp, R. (1972). Reducibility among Combinatorial Problems. In Miller, R., & Thatcher, J. (Eds.), *Complexity of Computer Computations*. Plenum Press.

Kloetzer, M., & Belta, C. (2008). A Fully Automated Framework for Control of Linear Systems from Temporal Logic Specifications. *IEEE Transactions on Automatic Control*, *53*(1), 287–297.

Kwon, Y., & Agha, G. (2008). LTLC: Linear Temporal Logic for Control. In *Proc. of HSCC*, Vol. 4981, pp. 316–329.

Lange, M. (2004). A Lower Complexity Bound for Propositional Dynamic Logic with Intersection.. In *Proc. of AiML*, pp. 133–147.

Li, J., Zhang, L., Pu, G., Vardi, M. Y., & He, J. (2014a). LTLf satisfiability checking. In *Proc. of ECAI*, pp. 513–518.

Li, J., Yao, Y., Pu, G., Zhang, L., & He, J. (2014b). Aalta: an ltl satisfiability checker over infinite/finite traces. In *Proc. of ACM SIGSOFT*, pp. 731–734.

Maggi, F. M., Mooij, A. J., & van der Aalst, W. M. P. (2011). User-guided discovery of declarative process models. In *Proc. of CIDM*, pp. 192–199.

Maggi, F. M. (2013). Declarative process mining with the declare component of prom. In *Proc. of BPM Demos*.

Maggi, F., Montali, M., Westergaard, M., & van der Aalst, W. (2011a). Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata. In *Proc. of BPM*, pp. 132–147.

Maggi, F., Mooij, A., & Aalst, W. (2011b). User-Guided Discovery of Declarative Process Models. In *Proc. of CIDM*, pp. 192–199.

Maggi, F., Westergaard, M., Montali, M., & van der Aalst, W. (2011c). Runtime Verification of LTL-Based Declarative Process Models. In *Proc. of RV*, pp. 131–146.

Markey, N. (2004). Past is for free: on the complexity of verifying linear temporal properties with past. *Acta Informatica*, *40*(6-7), 431–458.

Ono, H., & Nakamura, A. (1980). On the size of refutation Kripke models for some linear modal and tense logics. *Studia Logica*, *39*(4), 325–333.

Patrizi, F., Lipovetzky, N., De Giacomo, G., & Geffner, H. (2011). Computing Infinite Plans for LTL Goals Using a Classical Planner. In *Proc. of IJCAI*, pp. 2003–2008.

Pesic, M., Bosnacki, D., & van der Aalst, W. (2010). Enacting Declarative Languages Using LTL: Avoiding Errors and Improving Performance. In *Proc. of SPIN*, pp. 146–161.

Pesic, M., Schonenberg, H., & van der Aalst, W. (2007). DECLARE: Full Support for Loosely-Structured Processes. In *Proc. of EDOC*, pp. 287–298.

Pesic, M., & van der Aalst, W. (2006a). DecSerFlow: Towards a Truly Declarative Service Flow Language. In *The Role of Business Processes in Service Oriented Architectures*, No. 6291 in Dagstuhl Seminar Proceedings.

Pesic, M., & van der Aalst, W. M. P. (2006b). A Declarative Approach for Flexible Business Processes Management. In *Proc. of BPM*, pp. 169–180.

Pnueli, A. (1977). The Temporal Logic of Programs. In *Proc. of FOCS*, pp. 46–57.

Pnueli, A. (1981). The Temporal Semantics of Concurrent Programs. *Theoretical Computer Science*, *13*, 45–60.

Post, E. L. (1941). *On The Two-Valued Iterative Systems of Mathematical Logic*. Princeton University Press.

Rovani, M., Maggi, F. M., de Leoni, M., & van der Aalst, W. M. (2015). Declarative process mining in healthcare. *Expert Systems with Applications*, *42*(23), 9236–9251.

Rozier, K. Y. (2011). Linear temporal logic symbolic model checking. *Computer Science Review*, *5*(2), 163–203.

Rozier, K. Y., & Vardi, M. Y. (2007). LTL satisfiability checking. In *Proc. of Spin*, Vol. 4595, pp. 149–167. Springer.

Rozier, K. Y., & Vardi, M. Y. (2010). LTL satisfiability checking. *International Journal on Software Tools for Technology Transfer (STTT)*, *12*(2), 123–137.

Rozier, K. Y., & Vardi, M. Y. (2011). A multi-encoding approach for LTL symbolic satisfiability checking. In *Proc. of FM*, pp. 417–431. Springer.

Schobbens, P.-Y., & Raskin, J.-F. (1999). The Logic of Initially and Next, Complete Axiomatisation and Complexity Issues. *Information Processing Letters*, *69*(5), 221–225.

Schuppan, V., & Darmawan, L. (2011). Evaluating LTL Satisfiability Solvers.. In *Proc. of ATVA*, Vol. 6996, pp. 397–413. Springer.

Sistla, A. P., & Clarke, E. M. (1985). The Complexity of Propositional Linear Temporal Logics. *Journal of the ACM*, *32*(3), 733–749.

Sohrabi, S., Baier, J. A., & McIlraith, S. A. (2011). Preferred Explanations: Theory and Generation via Planning. In *Proc. of AAAI*, pp. 261–267.

van derAalst, W., Pesic, M., & Schonenberg, H. (2009). Declarative Workflows: Balancing Between Flexibility and Support. *Computer Science - Research and Development*, *23*(2), 99–113.

van Dongen, B., de Medeiros, A., Verbeek, H., Weijters, A., & van der Aalst, W. (2005). The ProM Framework: A New Era in Process Mining Tool Support. In Ciardo, G., & Darondeau, P. (Eds.), *Applications and Theory of Petri Nets 2005*, Vol. 3536, pp. 1105–1116. Springer Berlin / Heidelberg.