Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

The Complexity of LTL on Finite Traces: Hard and Easy Fragments

Valeria Fionda and Gianluigi Greco
DeMaCS, University of Calabria, Italy
{fionda, ggreco}@mat.unical.it

Abstract

This paper focuses on LTL on finite traces (LTL_f) for which satisfiability is known to be PSPACE-complete. However, little is known about the computational properties of fragments of LTL_f. In this paper we fill this gap and make the following contributions. First, we identify several LTL_f fragments for which the complexity of satisfiability drops to NP-complete or even P, by considering restrictions on the temporal operators and Boolean connectives being allowed. Second, we study a semantic variant of LTL_f, which is of interest in the domain of business processes, where models have the property that precisely one propositional variable evaluates true at each time instant. Third, we introduce a reasoner for LTL_f and compare its performance with the state of the art.

Introduction

Linear temporal logic (LTL) is a modal logic in which modalities are temporal operators relating events happening in different time instants over a linearly ordered timeline. LTL has been introduced in the seventies as a formal tool for verifying the correctness of computer programs and reactive systems and found applications in several fields of artificial intelligence and computer science. LTL formulas are interpreted over infinite *traces* (Pnueli 1977; 1981). However, there are certain applications, in particular, related to the specification and verification of business processes (Pesic, Bosnacki, and van der Aalst 2010; Pesic, Schonenberg, and van der Aalst 2007; van der Aalst, Pesic, and Schonenberg 2009), where a more natural choice is to focus on the LTL_f variant where formulas are interpreted over *finite* traces (Baier and McIlraith 2006; De Giacomo and Vardi 2013; 2015). Note that there are LTL formulas that admit infinite models but do not have any finite model (e.g., the formula $a \wedge \mathsf{G}(\mathsf{X}(a))$). Indeed, LTL and LTL_f exhibit subtle different behaviors and known properties for LTL cannot be easily generalized to LTL_f (see, e.g., (De Giacomo et al. 2014; De Giacomo, De Masellis, and Montali 2014)).

Given an LTL_f (resp., LTL) formula φ, the most relevant reasoning task is that of deciding whether there is some finite (resp., infinite) trace π that satisfies φ. Deciding the *satisfiability* of LTL formulas (even when X, G, and F are the only available temporal operators) is a PSPACE-complete problem in general (Sistla and Clarke 1985) and efforts have been spent over the years in order to come up with more favorable complexity results, by focusing on syntactic fragments of LTL (Artale et al. 2013; Bauland et al. 2009; Chen and Lin 1993; Demri and Schnoebelen 2002; Dixon, Fisher, and Konev 2007; Hemaspaandra 2001; Markey 2004; Ono and Nakamura 1980; Schobbens and Raskin 1999). When moving to LTL_f formulas, satisfiability is known to remain PSPACE-complete (De Giacomo and Vardi 2013) and, of course, the basic NP-hard "lower bound" (of propositional logic) still holds even in absence of temporal operators. However, little was known so far about the complexity for fragments of LTL_f, where satisfiability might be NP-complete or even tractable.

This paper fills this gap by providing results that are interesting from the theoretical point of view and that have concrete implications in the design of efficient heuristics and even *exact* solution approaches. In more detail, we consider classes of LTL_f formulas defined by syntactic restrictions on the temporal operators *and* on the Boolean connectives being allowed and provide the following contributions:

- We identify those classes for which satisfiability is always witnessed by models whose length is linear in the size of the formula. In particular, a dichotomy result is shown: Either a class enjoys the linear-length model property, or it contains satisfiable formulas for which no model exists whose length is polynomially bounded.
- We study the problem of model validation, that is checking whether some given trace π is a model of a given LTL_f formula φ. We show that this problem can be solved in polynomial time.
- We perform a systematic study of the complexity of satisfiability. For each class, satisfiability emerged to be either PSPACE-complete, or NP-complete, or even tractable.

Actually, for each of the points listed above, in addition to focusing on LTL_f fragments, we also consider (fragments of) a variant of LTL_f, which we name $\text{LTL}_{f,s}$, whose syntax is the same as LTL_f and whose semantics is defined over *simple (finite) traces*, i.e., over finite traces such that exactly one propositional variable holds at each time instant. The study of $\text{LTL}_{f,s}$ is motivated by the fact that

in the applications related to the specification and verification of business processes, simple traces are generally considered (Pesic, Bosnacki, and van der Aalst 2010; Pesic, Schonenberg, and van der Aalst 2007; van der Aalst, Pesic, and Schonenberg 2009).

Motivated by the theoretical results we implemented a reasoner for LTL$_f$, called LTL2SAT[1], and compared it with Aalta (Li et al. 2014), the only existing reasoner that specifically targets the finite trace case, which has been shown to outperform existing LTL reasoners adapted to deal with finite traces (cf. (Edelkamp 2006; Gerevini et al. 2009; Pesic and van der Aalst 2006)).

Formal Framework

Syntax. Assume that a universe $\mathcal{V}$ of 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 mainly[2] focus on the temporal operators "X" (next), "G" (always), and "F" (eventually), and we allow *atomic negation* only. Formally, φ is built according to the following grammar[3]:

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

where x is any variable in $\mathcal{V}$. The set of all variables occurring (either positively or negated) in φ is denoted by $\mathcal{V}_\varphi$.

The temporal height of φ w.r.t. an operator $\mathsf{O}\in\{\mathsf{X},\mathsf{G},\mathsf{F}\}$, denoted by $th(\varphi,\mathsf{O})$, is the maximum number of *nested* O in φ. Instead, its temporal size w.r.t. O is the overall number of occurrences of O in φ, denoted by $ts(\varphi,\mathsf{O})$. The size of φ, denoted by $||\varphi||$, is the total number of symbols, excluding parenthesis, occurring in it.

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

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

Semantics. A *finite trace* over variables in $\mathcal{V}$ is a sequence π=$\pi_0,\pi_1,...,\pi_{n-1}$ 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 $len(\pi)$. Its *size* is $||\pi||$=$\sum_{i=0}^{n-1}|\pi_i|$.

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 all its *subformulas*:

$\pi,i\models x$ 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 $i<len(\pi)$-1 and $\pi,i+1\models\varphi'$;
$\pi,i\models\mathsf{G}(\varphi')$ iff $\forall j$ with $i\leq j<len(\pi)$, $\pi,j\models\varphi'$.
$\pi,i\models\mathsf{F}(\varphi')$ iff $\exists j$ with $i\leq j<len(\pi)$ s.t. $\pi,j\models\varphi'$;

Whenever $\pi,0\models\varphi$ holds, we just write $\pi\models\varphi$, and we say that π is a *model* of φ and that φ is *satisfiable*. Note that if φ does not contain temporal operators, then $\pi\models\varphi$ reduces to the notion of satisfiability for propositional logic.

A model π of φ is said *simple* if $|\pi_i|=1$ and $\pi_i\subseteq\mathcal{V}_\varphi$, for each $i\in\{0,...,len(\pi)\text{-}1\}$. In addition to LTL$_f$, we consider the logic LTL$_{f,s}$, whose syntax coincides with the syntax of LTL$_f$, and whose semantics, denoted as $\models_s$, differs only in the fact that satisfiability is defined w.r.t. simple models only.

For each $B\subseteq\{\wedge,\vee,\neg\}$ and $T\subseteq\{\mathsf{X},\mathsf{G},\mathsf{F}\}$, the class $\langle B,T\rangle$-LTL$_{f,s}$ is defined analogously to $\langle B,T\rangle$-LTL$_f$.

Example 2 Consider again the formula φ of Example 1 and check that a model for it is the trace π 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=$\{a\}$. Because of the subformula $c\wedge\mathsf{G}(a)$, φ does not admit any *simple*-model. $\lhd$

In the following, parenthesis will be often omitted when this does not originate ambiguities in the above definition of satisfiability. In particular, we shall assume as usual that, in the evaluation of Boolean expressions, conjunction has precedence over disjunction.

Linear-Length Model Property

In this section, we analyze all fragments of LTL$_f$ and LTL$_{f,s}$ obtained by constraining the allowed Boolean connectives and temporal operators to identify upper bounds on the lengths of models in terms of the size of the formula given.

Theorem 3 *Every $\langle B,T\rangle$-LTL$_f$ (resp., $\langle B,T\rangle$-LTL$_{f,s}$) satisfiable formula φ has a model (resp. simple model) π such that $len(\pi)\leq 2^{O(||\varphi||)}$.*

In the rest of this section we identify those classes of formulas $\langle B,T\rangle$-*LTL$_f$* and $\langle B,T\rangle$-*LTL$_{f,s}$* enjoying the *linear-length model property*, i.e., such that satisfiability can be always witnessed by models π whose length, $len(\pi)$, is linear in the size $||\varphi||$. More specifically, all classes of LTL$_f$ and LTL$_{f,s}$ formulas that enjoy the linear-length model property are summarized in Figure 1, in terms of the Hasse diagram built over the subsets of $\{\mathsf{X},\mathsf{G},\mathsf{F}\}$. In particular, for each subset $T\subseteq\{\mathsf{X},\mathsf{G},\mathsf{F}\}$, the figure reports the subset-maximal set $B\subseteq\{\wedge,\vee,\neg\}$ for which the property holds. For LTL$_f$, it emerges that it is crucial to forbid the interplay of X and G, for otherwise the property holds if, and only if, formulas are built over at most two Boolean connectives. Interestingly, for LTL$_{f,s}$, results are more stringent.

Theorem 4 *All LTL$_f$ and LTL$_{f,s}$ fragments summarized in Figure 1 enjoy the linear-length model property.*

Proof Sketch. The result can be shown by (standard) structural induction for the fragments $\langle\{\wedge,\vee\},\{\mathsf{X},\mathsf{G},\mathsf{F}\}\rangle$-LTL$_f$, $\langle\{\vee,\neg\},\{\mathsf{X},\mathsf{G},\mathsf{F}\}\rangle$-LTL$_f$, and $\langle\{\vee,\neg\},\{\mathsf{X},\mathsf{G},\mathsf{F}\}\rangle$-LTL$_{f,s}$.

[1] See https://ltl2sat.wordpress.com/

[2] In fact, we shall see that our algorithmic approach to deal with satisfiability immediately applies to arbitrary negation and other temporal operators, such as *until* and *weak next*.

[3] As we allow atomic negation only, no Boolean connective and no temporal operator can be rewritten in terms of the others. Accordingly, all of them are made explicit in the syntax.

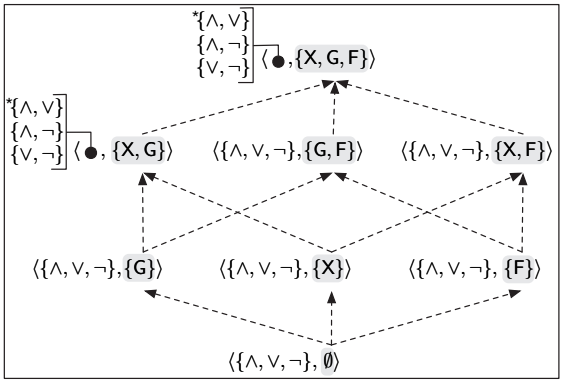

Figure 1: Classes of formulas enjoying the linear-length model property—for each T, subset-maximal sets B are reported. *The property holds on LTL$_f$, but not on LTL$_{f,s}$.

Different is the case where φ is a satisfiable formula in $\langle\{\wedge, \neg\}, \{\mathsf{X}, \mathsf{G}, \mathsf{F}\}\rangle$-LTL$_f$. Let π be a model of φ, and define the *critical* time instants as follows. The initial time instant is critical. If $\mathsf{X}(\varphi')$ or $\mathsf{F}(\varphi')$ is a subformula of φ, then the time instant where φ' is required to hold in the recursive definition of satisfiability is critical. No further time instant is critical. It can be checked that the finite trace $\bar{\pi}$ derived from π by removing each state associated with a time instant that is not critical still satisfies φ, i.e., $\bar{\pi} \models \varphi$. Moreover, if π is a simple model, then $\bar{\pi}$ is simple, too. Minor modifications in the reasoning lead to establish the property on $\langle\{\wedge, \vee, \neg\}, \{\mathsf{X}, \mathsf{F}\}\rangle$-LTL$_f$ and $\langle\{\wedge, \vee, \neg\}, \{\mathsf{X}, \mathsf{F}\}\rangle$-LTL$_{f,s}$.

Consider then the case where φ is a satisfiable formula in $\langle\{\wedge, \vee, \neg\}, \{\mathsf{G}, \mathsf{F}\}\rangle$-LTL$_f$. The idea is to encode φ in terms of an "equivalent" LTL formula $\bar{\varphi}$ over $\mathcal{V}_\varphi \cup \{end\}$, where end is a fresh Boolean variable, by using the approach discussed by (De Giacomo, De Masellis, and Montali 2014). In particular, by specializing that result over the fragment, we obtain that: (1) $\bar{\varphi}$ does not contain X and G as temporal operator; (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 $\{end\}$ infinitely often; (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) $ts(\bar{\varphi}, \mathsf{F}) = ts(\varphi, \mathsf{F}) + ts(\varphi, \mathsf{G}) + 4$. Eventually, by Theorem 3.4 in (Sistla and Clarke 1985)—which holds on LTL formulas defined over F only—and given that all models of $\bar{\varphi}$ have the form $\omega(\pi)$, we are guaranteed about the existence of a model $\omega(\pi')$ for $\bar{\varphi}$ such that $len(\pi') \leq ts(\bar{\varphi}, \mathsf{F}) + 1$. In fact, we know that π' is a finite model of φ and we get $len(\pi') \leq ts(\bar{\varphi}, \mathsf{F}) + 1 = ts(\varphi, \mathsf{F}) + ts(\varphi, \mathsf{G}) + 4 + 1$.

Finally, if φ is a $\langle\{\wedge, \vee, \neg\}, \{\mathsf{G}, \mathsf{F}\}\rangle$-LTL$_{f,s}$ formula, then we define φ^s as the formula

$$\varphi \wedge \mathsf{G}(\bigvee_{x_i \in \mathcal{V}_\varphi} (x_i \wedge \bigwedge_{x_j \in \mathcal{V}_\varphi \setminus \{x_i\}} \neg x_j)).$$

Note that π is a model of φ^s if, and only if, π is a simple model of φ. In fact, we have just observed that if φ^s is satisfiable, then it has a model π such that $len(\pi) \leq ts(\varphi^s, \mathsf{F}) + ts(\varphi^s, \mathsf{G}) + 5$. The result then follows since $ts(\varphi^s, \mathsf{F}) = ts(\varphi, \mathsf{F})$ and $ts(\varphi^s, \mathsf{G}) = ts(\varphi, \mathsf{G}) + 1$. □

Interestingly, our characterization is complete, as the classes of LTL$_f$ and LTL$_{f,s}$ that are not covered by Figure 1 do not enjoy the linear-length model property, and formulas can be exhibited for them for which no model has a length that is polynomially bounded at all.

Theorem 5 *There are satisfiable $\langle\{\wedge, \vee, \neg\}, \{\mathsf{X}, \mathsf{G}\}\rangle$-LTL$_f$ formulas φ for which there is no model whose length is polynomial w.r.t. $||\varphi||$.*

Proof Sketch. The behavior of an n-bits counter can be encoded in a formula φ in $\langle\{\wedge, \vee, \neg\}, \{\mathsf{X}, \mathsf{G}\}\rangle$-LTL$_f$over the variables $\{x_1, ..., x_n\}$, where each x_i is meant to represent the i-bit of the counter. Formally, we define $\varphi = (\bigwedge_{i=1}^{n} \neg x_i \wedge cont) \wedge \mathsf{G}(\neg cont \leftrightarrow \bigwedge_{i=1}^{n} x_i) \wedge \bigwedge_{i=1}^{n} \mathsf{G}(\neg cont \vee \hat{C}_i)$, where $\hat{C}_i$ encodes the move from one number to the successive one (in correspondence with the time instants): $\hat{C}_1 = \mathsf{X}(x_1) \leftrightarrow \neg x_1$ and $\hat{C}_i = \mathsf{X}(x_i) \leftrightarrow ((\neg x_i \wedge \bigwedge_{j=1}^{i-1} x_j) \vee (x_i \wedge \bigvee_{j=1}^{i-1} \neg x_j))$. Any model of φ needs 2^n states. □

The result for $\langle\{\wedge, \vee\}, \{\mathsf{X}, \mathsf{G}\}\rangle$-*LTL$_{f,s}$* formulas, follows by combining Theorem 5 with the following lemma, showing that negation can be simulated in $\langle\{\wedge, \vee\}, \{\mathsf{X}\}\rangle$-LTL$_{f,s}$.

Lemma 6 *For each $\langle\{\wedge, \vee, \neg\}, T\rangle$-LTL$_f$ formula φ with $T \supseteq \{\mathsf{X}\}$, a $\langle\{\wedge, \vee\}, T\rangle$-LTL$_{f,s}$ formula φ^s can be built in polynomial time such that: (1) if $\pi \models \varphi$, then there is a model π^s of φ^s with $len(\pi^s) = len(\pi) \times (|\mathcal{V}_\varphi| + 1)$; (2) if $\pi^s \models_s \varphi^s$, then there is a model π of φ with $len(\pi) \leq len(\pi^s)$.*

Proof Sketch. The idea is to encode φ into a formula φ^s that associate a sequence of $|\mathcal{V}_\varphi|$ time instants to each time instant in φ. In particular, the i time instant in each sequence of each model π^s is associated to the variable x_i in $\mathcal{V}_\varphi$ and either $\pi^s, i \models x_i$ or $\pi^s, i \models \bar{x}_i$ holds, where $\bar{x}_i$ encodes the complement of x_i. □

Model Verification for Finite Traces

In this section, we study the verification problem, i.e., the problem of deciding whether some given finite trace is a model of a given formula.

Given a formula φ, let us denote by $pt(\varphi) = (V, E, \lambda)$ its associated *parse tree*, which is a rooted tree (V, E) with a labeling function $\lambda : V \to \{\wedge, \vee, \mathsf{X}, \mathsf{F}, \mathsf{G}\} \cup \{x, \neg x \mid x \in \mathcal{V}_\varphi\}$. In particular, leaves are labeled with literals, and internal nodes are labeled with Boolean or temporal operators. Since the parse tree is a well-known concept in the analysis of formal languages, we omit its formal definition, and help the intuition by reporting in Figure 2 the parse tree associated with the formula discussed in Example 1.

Given a trace π, we define an algorithm based on equipping each node $v \in V$ with a set $sat(v, \pi)$ such that:

- If $\lambda(v) = x$ or $\lambda(v) = \neg x$, then $sat(v, \pi) = \{i \mid \pi, i \models \lambda(v)\}$;
- If $\lambda(v) = \wedge$ (resp., $\lambda(v) = \vee$), then $sat(v, \pi) = sat(v_1, \pi) \cap sat(v_2, \pi)$ (resp., $sat(v, \pi) = sat(v_1, \pi) \cup sat(v_2, \pi)$) 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 $pt(\varphi)$;

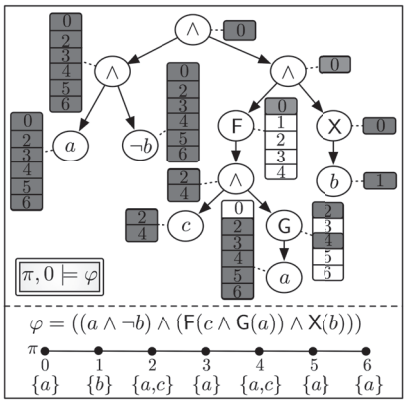

Figure 2: The parse tree of the formula φ of Example 1. The set $sat(v, \pi)$ is reported next to the vertex v.

- If $\lambda(v)$=F, then $sat(v,\pi)$=$\{j \mid \exists i \in sat(v',\pi)$ such that $j \leq i\}$ with v' being the only child of v in $pt(\varphi)$;
- If $\lambda(v)$=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 $pt(\varphi)$.

Example 7 In Figure 2, each vertex v is equipped with the set $sat(v, \pi)$ for the model π discussed in Example 2, where $sat(v, \pi)$ consists of all time instants where the subformula associated with the parse tree rooted at v hold in π. $\lhd$

The intuition in the above example is now generalized.

Theorem 8 *Checking whether a finite trace π is a model of an* LTL$_f$ *(resp.,* LTL$_{f,s}$*) formula φ is feasible in $O(len(\pi)^2 \times ||\varphi||)$.*

Proof Sketch. The definition of $sat(v, \pi)$ "encodes" the semantics of the subformula φ_v associated with the parse tree rooted at v. By structural induction, we can hence show that $sat(v,\pi)$=$\{i \mid \pi, i \models \varphi_v\}$. By letting r be the root, we get that $0 \in sat(r, \pi)$ holds if, and only if, $\pi, 0 \models \varphi_r$ (with φ_r coinciding with φ). Additionally, if φ is a LTL$_{f,s}$ formula, then we have to check that π is *simple*, which is easily feasible by iterating over the time instants of π. Running time follows by simple calculations and by noticing that each vertex $v \in V$ can be associated with $sat(v, \pi)$ by processing the vertices of $pt(\varphi)$ bottom-up (for example, see Figure 2 where time instants that are propagated from each vertex to its parent are highlighted in gray). $\square$

Satisfiability of LTL$_f$ and LTL$_{f,s}$

In this section, we analyze the computational complexity of the satisfiability problem. Our results are summarized in Figure 3, from which it emerges that reasoning about LTL$_{f,s}$ is more complex than reasoning about LTL$_f$. As it is not possible to illustrate detailed proofs for all the cases reported in Figure 3, in the following we provide some general intuitions and focus on some selected classes.

Classes of Hard formulas

Membership in PSPACE for the satisfiability problem of arbitrary LTL$_f$ formulas has been shown by (De Giacomo and Vardi 2013); the extension to LTL$_{f,s}$ formulas is immediate. Similarly, they exhibited a class of LTL$_f$ formulas over which satisfiability is PSPACE-hard. All PSPACE-hardness results in Figure 3 are obtained by syntactic manipulations of that encoding, aimed to guarantee membership in the given LTL fragment of interest. Results are extended to LTL$_{f,s}$ formulas via Lemma 6.

We continue our analysis by considering the cases that emerge to be NP-complete. To this end, observe first that membership in NP clearly holds for all classes of $\langle B, T\rangle$-LTL$_f$ (resp., $\langle B, T\rangle$-LTL$_{f,s}$) formulas that enjoy the linear-length model property. Indeed, given a formula φ belonging to a class enjoying such property, a non deterministic Turing Machine can guess a trace π having linear length, and then check in polynomial time (cf. Theorem 8) whether $\pi \models \varphi$ or $\pi \models_s \varphi$. Note that $||\pi||$ is polynomially bounded w.r.t. $||\varphi||$, because in the worst case each state of π contains all the variables over which φ is defined.

Accordingly, in the proofs of the NP-completeness results, we focus on showing the NP-hardness part only. When all Boolean connectives are allowed, the result is immediately entailed by the NP-hardness of the satisfiability for Boolean formulas. In its turn, NP-hardness of $\langle\{\wedge, \vee\}, \{\mathsf{X}, \mathsf{F}\}\rangle$-LTL$_{f,s}$ follows by this observation and Lemma 6. And, eventually, the only remaining NP-complete case for LTL$_{f,s}$ formulas without negation is when the allowed temporal operators are G and F—see, again, Figure 3.

Theorem 9 *Satisfiability of $\langle\{\wedge, \vee\}, \{\mathsf{G}, \mathsf{F}\}\rangle$-LTL$_{f,s}$ formulas is* NP*-hard.*

Proof Sketch. Consider the NP-hard ONE-IN-THREE POSITIVE 3-SAT problem (Garey and Johnson 1979): Given a Boolean formula $\phi = C_1 \wedge \cdots \wedge C_m$ 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 $x_{j_1}, x_{j_2}, x_{j_3} \in \{x_1, ..., x_n\}$, we have to decide whether there is a truth assignment such that, for each clause, *precisely* one variable evaluates true.

Based on ϕ, we build a $\langle\{\wedge, \vee\}, \{\mathsf{G}, \mathsf{F}\}\rangle$-LTL$_{f,s}$ formula φ_ϕ over the same 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

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

It can be checked that there is a truth assignment such that, for each clause of ϕ, precisely one variable evaluates true if, and only if, φ_ϕ is satisfiable. $\square$

When restrictions are considered on the Boolean connectives being allowed, LTL$_f$ formulas are tractable but for the case addressed below—see, again, Table 3.

Theorem 10 *Satisfiability of $\langle\{\wedge, \neg\}, \{\mathsf{X}, \mathsf{G}, \mathsf{F}\}\rangle$-LTL$_f$ formulas is* NP*-hard.*

LTL$_f$							
	$\wedge,\vee,\neg$	$\wedge,\vee$	$\wedge,\neg$	$\vee,\neg$	$\wedge$	$\vee$	$\neg$
G	NP-c	in P	in P	in P	in P	in P	in P
F	NP-c	in P	in P	in P	in P	in P	in P
X	NP-c	in P	in P	in P	in P	in P	in P
XF	NP-c	in P	in P	in P	in P	in P	in P
GF	NP-c	in P	in P	in P	in P	in P	in P
XG	PSPACE-c	in P	in P	in P	in P	in P	in P
XGF	PSPACE-c	in P	NP-c	in P	in P	in P	in P

LTL$_{f,s}$							
	$\wedge,\vee,\neg$	$\wedge,\vee$	$\wedge,\neg$	$\vee,\neg$	$\wedge$	$\vee$	$\neg$
G	in P	in P	in P	in P	in P	in P	in P
F	in P	in P	in P	in P	in P	in P	in P
X	NP-c	NP-c	in P	in P	in P	in P	in P
XF	NP-c	NP-c	in P	in P	in P	in P	in P
GF	NP-c	NP-c	in P	in P	in P	in P	in P
XG	PSPACE-c	PSPACE-c	in P	in P	in P	in P	in P
XGF	PSPACE-c	PSPACE-c	NP-c	in P	NP-c	in P	in P

Figure 3: Summary of results for the satisfiability problem on $\langle B,T\rangle$-LTL$_f$ and $\langle B,T\rangle$-LTL$_{f,s}$ formulas. All results are either tractability results (P), or completeness results for the classes NP or PSPACE. Tractability results are based on algorithms over the parse trees of formulas, and their correctness is shown by exploiting the bounds on the length of models discussed in Section "Linear-Length Model Property". Over $\{\wedge,\vee\}$ fragments, such algorithms are computationally trivial.

Proof Sketch. As in the proof of Theorem 9, we exhibit a reduction from the ONE-IN-THREE POSITIVE 3-SAT problem. Given ϕ, we build the $\langle\{\wedge,\neg\},\{\mathsf{X},\mathsf{G},\mathsf{F}\}\rangle$-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

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

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

We now claim that: there is a truth assignment 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 subformulas κ'_j, we have that $\pi_0 \supseteq \{y_1,...,y_m\}$. Moreover, 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. The three conjuncts of the subformula ϱ_j must be mapped to time instants where y_j holds and one of them must evaluate true in the initial time instant. Thus, 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.

(only-if part) Consider a truth assignment σ such that, for each clause of φ_ϕ, precisely one variable evaluates true. Consider the trace π with $len(\pi)=2\times m+2$ such that $\pi_0 = \{y_1,...,y_m\} \cup \{x_i | \sigma(x_i)=true\}$ and for each clause C_j the two variables evaluating $false$ in σ are $true$ in $\pi_{2\times j-1}$ and $\pi_{2\times j}$. It is immediate to check that $\pi \models \varphi_\phi$. □

A rather elaborate adaptation of the above proof leads to establish NP-hardness for $\langle\{\wedge\},\{\mathsf{X},\mathsf{G},\mathsf{F}\}\rangle$-LTL$_{f,s}$.

Tractable Classes of LTL$_f$ and LTL$_{f,s}$

In this section we describe the tractable cases. First, we observe that tractability for $\langle\{\wedge,\neg\},\{\mathsf{G},\mathsf{F}\}\rangle$-LTL$_f$ and $\langle\{\wedge,\neg\},\{\mathsf{G},\mathsf{F}\}\rangle$-LTL$_{f,s}$ follows from (Hemaspaandra 2001) (Theorem 2.1). For the other fragments, tractability results can be established by providing constructive arguments for the proofs of Section *Linear-Length Model Property*. An example is reported below.

Theorem 11 *Satisfiability of* $\langle\{\wedge,\neg\},\{\mathsf{X},\mathsf{F}\}\rangle$-LTL$_f$ *and* $\langle\{\wedge,\neg\},\{\mathsf{X},\mathsf{F}\}\rangle$-LTL$_{f,s}$ *formulas is in* P.

Proof Sketch. Let us consider a $\langle\{\wedge,\neg\},\{\mathsf{X},\mathsf{F}\}\rangle$-LTL$_f$ or $\langle\{\wedge,\neg\},\{\mathsf{X},\mathsf{F}\}\rangle$-LTL$_{f,s}$ formula φ; it can be rewritten in polynomial time according to the grammar:

$$\varphi ::= \varphi' | \mathsf{X}(\varphi) | \mathsf{F}(\varphi) | (\varphi' \wedge \mathsf{X}(\varphi)) | (\varphi' \wedge \mathsf{F}(\varphi)) | (\varphi' \wedge (\mathsf{X}(\varphi) \wedge \mathsf{F}(\varphi)))$$
$$\varphi' ::= x \mid \neg x \mid (\varphi' \wedge \varphi')$$

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. If there is a vertex $v \in V'$ such that (the Boolean formula) φ_v is not satisfiable, then φ does not admit a model. Note that this condition can be checked in polynomial time, since φ_v is a Boolean formula with atomic negation and without disjunction. For each vertex $v \in V'$, let M^v be a model of φ_v that has been computed in polynomial time. In particular, w.l.o.g., assume that there is no model $\bar{M}^v$ of φ_v with $|\bar{M}^v| < |M^v|$.

In order to build a model of φ, 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—note that the algorithm is well-defined and covers all possible cases, given the syntactic form of φ:

- if $\lambda(v)=\mathsf{X}$ and c is the child of v, then π^v is obtained from π^c by inserting the state $\emptyset$ as the initial state;
- if $\lambda(v)=\mathsf{F}$ and c is the child of v, then $\pi^v = \pi^c$;
- if $\lambda(v)=\wedge$ and v has two children, c_1 and c_2, and one of them, say c_2, is such that $\lambda(c_2) = \mathsf{F}$, then π^v is obtained by appending π^{c_2} after the last time instant of π^{c_1};

- if $\lambda(v)=\wedge$ and v has two children, c_1 and c_2, with $c_1 \in V'$ and $\lambda(c_2) \neq \mathsf{F}$, then π^v is obtained from π^{c_2} by replacing the initial state (in fact, note that $\pi_0^{c_2} = \emptyset$) with M^{c_1}.

It is immediate to check that π^v is a model of φ_v, for each v. So, if r is the root, then π^r is a model of φ. In particular, note that if π^r is not *simple*, then there is some vertex $v \in V'$ such that $|M^v| > 1$. But, this immediately entails that φ does not have any simple model at all. □

Implementation

In this section we discuss a reasoner, called LTL2SAT[4], for LTL_f and $\text{LTL}_{f,s}$ formulas, built upon our findings.

Encoding Approach. For the classes on which satisfiability is feasible in polynomial time, LTL2SAT implements the algorithms described in Section *Satisfiability of LTL_f and $\text{LTL}_{f,s}$*. For the other classes, LTL2SAT rewrites the input formula into an equivalent Boolean formula and uses *glucose* (Audemard and Simon 2009) to compute a model.

Let φ be a formula, and let $pt(\varphi) = (V, E, \lambda)$ be its parse tree. Let n be a bound on the maximum length of the models, which can be fixed according to the results discussed in Section *Linear-Length Model Property*. Based on φ and n, we build a Boolean formula $\Phi_{\varphi,n}$ as follows. First, we define the variables $\ell[0], ..., \ell[n\text{-}1]$ encoding the last time instant of the model, and associate to each node v of $pt(\varphi)$ the variables $s_v[0], ..., s_v[n\text{-}1]$. Intuitively, $s_v[i]$ is meant to check whether the formula encoded in the parse tree rooted at v holds at the time instant i. Then, we set[5] $\Phi_{\varphi,n} = \bigwedge_{v \in V} \Phi_v \wedge s_{root}[0] \wedge \ell[n\text{-}1]$ and, for each v, $\Phi_v = \bigwedge_{i=0}^{n-1}(s_v[i] \leftrightarrow \Phi_v^i) \wedge \bigwedge_{i=0}^{n-2}(\ell[i] \rightarrow \ell[i\text{+}1])$, where:

- 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 $\lambda(v) = \mathsf{X}$ (resp., $\lambda(v) = \mathsf{F}$; $\lambda(v) = \mathsf{G}$), then $\Phi_v^i = s_{v'}[i\text{+}1] \wedge \neg\ell[i]$ (resp., $\Phi_v^i = s_{v'}[i] \vee s_v[i\text{+}1] \wedge \neg\ell[i]$; $\Phi_v^i = s_{v'}[i] \wedge (s_v[i\text{+}1] \vee \ell[i])$), where v' is the child of v and $s_v[n]$ is the constant `false` (resp., `false`; `true`).

Note that Φ_v^i mimics the model checking strategy illustrated in Section *Model Verification for Finite Traces*. Thus, the following can be established.

Theorem 12 *$\Phi_{\varphi,n}$ is satisfiable if, and only if, φ can be satisfied by a model π such that $len(\pi) \leq n$.*

Moreover, note that with simple modifications in the above formulas, we can handle not only simple models, but also negation that is not atomic and other temporal operators, such as *until* and *weak next*.[6] Our implementation supports these extensions by applying the bound given by Theorem 3.

[4]Downloadable at http://ltl2sat.wordpress.com/

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

[6]The support of this operator is crucial when rewriting formulas in terms of atomic negation (Li et al. 2014).

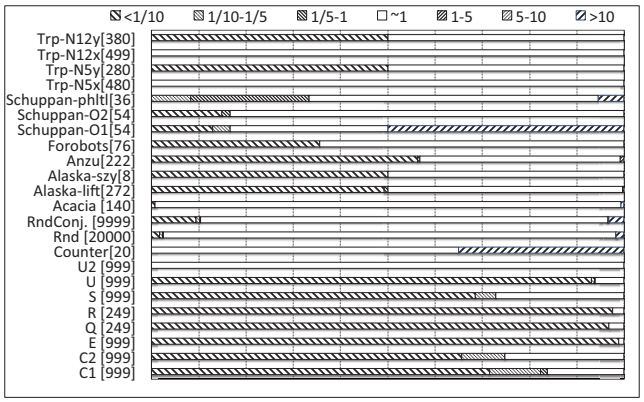

Figure 4: Ratio LTL2SAT/Aalta, on execution time.

In fact, a finer grained analysis of them, including the identification of tighter bounds and islands of tractability, are interesting questions that we leave for further research.

Experimental Results. LTL2SAT implements the *optimization strategies* in (Li et al. 2014) in addition to the SAT rewriting technique. When the optimization strategies are not applicable, the size n of the trace to be used in the SAT rewriting is initially set to 1: If no model is found, then n is doubled and the process is repeated until n exceeds the theoretical upper bound (or a 20s time-limit is reached).

Our approach, which shares the spirit of (incremental) bounded LTL model checking methods based on SAT rewritings (Biere et al. 2006), has been compared to *Aalta* (Li et al. 2014), which is the only existing reasoner that specifically targets the finite trace case. We used the same datasets used in (Li et al. 2014), where Aalta has been shown to outperform LTL reasoners adapted to deal with finite traces (cf. (Edelkamp 2006; Gerevini et al. 2009; Pesic and van der Aalst 2006)). By a preliminary analysis of the datasets we identified 8 out of 23 datasets (i.e., C1, C2, E, Q, R, S, Alaska-szy and Shuppan-O2) that consist of formulas belonging to classes enjoying the linear length model property and 6 of them (i.e., C1, C2, E, Q, R and S) actually refer to classes for which satisfiability is tractable.

Experiments have been executed on a PC Intel Core i5 2,4 GHz, 8GB RAM. For each formula, we measured the ratio between the time required by LTL2SAT to check satisfiability and that required by Aalta. Figure 4 reports the results as percentage stacked bar charts, for each dataset. The number of formulas in each dataset is also illustrated just next to its name. White bars represent the percentages of instances where the two systems behaves 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., Aalta) is faster.

LTL2SAT outperforms Aalta in $\sim$70% of the datasets (i.e., C1, C2, E, Q, R, S, U, RndConj, Alaska-lift, Alaska-szy, Anzu, Forobots, Schuppan-O2, Schuppan-phltl,Trp-5y and Trp-12y) and, by considering all LTL formulas at all, in the $\sim$17% of the formulas, while the two systems are comparable in performance in the $\sim$22% of the datasets and in the

$\sim$80% of the formulas. However, by excluding the largest dataset (i.e., Rnd), where the performance of the two system are comparable, LTL2SAT wins on the $\sim$34% of the formulas and on the $\sim$64% the two systems perform the same. Note that, the speedup obtained by LTL2SAT is particularly relevant in C1, C2, E, Q, R, S, U, Alaska-lift, Alaska-szy, Anzu, Trp-5y and Trp-12y where in the 50% or more of the formulas LTL2SAT runs at least 10 times faster than Aalta and all the datasets consisting of formulas for which satisfiability has been shown to be tractable follow in such group.

The number of formulas where both systems reach the time-limit is negligible on each dataset, but for *Counter* (12 out of 20), *RndConj* (746 out of 9999) and *Forobots* (11 out of 77). While Aalta did not give any information, LTL2SAT guarantees that there is no model with length less than the value of n used in the last iteration, that is, on average, 10500, 5150 and 4096 for each dataset, respectively.

Conclusion

We have studied satisfiability and model checking problems for a large number of LTL_f and $\text{LTL}_{f,s}$ fragments. From the theoretical point of view, a natural avenue of further research is to extend the analysis to further temporal operators. From the practical viewpoint, our results on $\text{LTL}_{f,s}$ can be used to provide runtime monitoring capabilities to business processes, as in (De Giacomo et al. 2014). Moreover, following the perspective of (Greco et al. 2015), our system can support process mining tasks, by coupling learning methods with constraints expressed in LTL_f. Another line we are currently investigating is the application of our framework for reasoning about temporal constraints over the items of combinatorial actions, especially in the context of supply chain formation (Fionda and Greco 2013; Gottlob and Greco 2013).

Acknowledgments

G. Greco's work was also supported by a Kurt Gödel Research Fellowship, awarded by the Kurt Gödel Society.

References

Artale, A.; Kontchakov, R.; Ryzhikov, V.; and Zakharyaschev, M. 2013. The complexity of clausal fragments of ltl. In *LPAR*.

Audemard, G., and Simon, L. 2009. Predicting learnt clauses quality in modern SAT solvers. In *IJCAI*.

Baier, J. A., and McIlraith, S. A. 2006. Planning with First-Order Temporally Extended Goals using Heuristic Search. In *AAAI*.

Bauland, M.; Schneider, T.; Schnoor, H.; Schnoor, I.; and Vollmer, H. 2009. The complexity of generalized satisfiability for linear temporal logic. *Logical Methods in Computer Science* 5(1).

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

Chen, C.-C., and Lin, I.-P. 1993. The Computational Complexity of Satisfiability of Temporal Horn Formulas in Propositional Linear-Time Temporal Logic. *Inf. Process. Lett.* 45(3):131–136.

De Giacomo, G., and Vardi, M. Y. 2013. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *IJCAI*.

De Giacomo, G., and Vardi, M. Y. 2015. Synthesis for LTL and LDL on finite traces. In *IJCAI*.

De Giacomo, G.; De Masellis, R.; Grasso, M.; Maggi, F. M.; and Montali, M. 2014. Monitoring Business Metaconstraints Based on LTL and LDL for Finite Traces. In *BPM*.

De Giacomo, G.; De Masellis, R.; and Montali, M. 2014. Reasoning on LTL on Finite Traces: Insensitivity to Infiniteness. In *AAAI*.

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

Dixon, C.; Fisher, M.; and Konev, B. 2007. Tractable temporal reasoning. In *IJCAI*.

Edelkamp, S. 2006. On the Compilation of Plan Constraints and Preferences. In *ICAPS*.

Fionda, V., and Greco, G. 2013. The complexity of mixed multi-unit combinatorial auctions: Tractability under structural and qualitative restrictions. *Artif. Intell.* 196:1–25.

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

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

Gottlob, G., and Greco, G. 2013. Decomposing combinatorial auctions and set packing problems. *J. ACM* 60(4):24.

Greco, G.; Guzzo, A.; Lupia, F.; and Pontieri, L. 2015. Process discovery under precedence constraints. volume 9, 32.

Hemaspaandra, E. 2001. The complexity of poor man's logic. *J. Log. Comput.* 11(4):609622.

Li, J.; Zhang, L.; Pu, G.; Vardi, M. Y.; and He, J. 2014. LTLf satisfiability checking. In *ECAI*.

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

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

Pesic, M., and van der Aalst, W. 2006. DecSerFlow: Towards a Truly Declarative Service Flow Language. In *The Role of Business Processes in Service Oriented Architectures*, number 6291 in Dagstuhl Seminar Proceedings.

Pesic, M.; Bosnacki, D.; and van der Aalst, W. 2010. Enacting Declarative Languages Using LTL: Avoiding Errors and Improving Performance. In *SPIN*.

Pesic, M.; Schonenberg, H.; and van der Aalst, W. 2007. DECLARE: Full Support for Loosely-Structured Processes. In *EDOC*.

Pnueli, A. 1977. The temporal logic of programs. In *FOCS*.

Pnueli, A. 1981. The Temporal Semantics of Concurrent Programs. *Theor. Comput. Sci.* 13:45–60.

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

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

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