# LTL on Weighted Finite Traces: Formal Foundations and Algorithms

**Carmine Dodaro** , **Valeria Fionda*** and **Gianluigi Greco**
Department of Mathematics and Computer Science, University of Calabria
{carmine.dodaro,valeria.fionda,gianluigi.greco}@unical.it

## Abstract

LTL on finite traces ($LTL_f$) is a logic that attracted much attention in recent literature, due to its ability to formalize the qualitative behavior of dynamical systems in several application domains. However, its practical usage is still rather limited, as $LTL_f$ cannot deal with any quantitative aspect, such as with the costs of realizing some desired behaviour. The paper fills the gap by proposing a weighting framework for $LTL_f$ encoding such quantitative aspects in the traces over which it is evaluated. The complexity of reasoning problems on *weighted traces* is analyzed and compared to that of standard $LTL_f$, by considering arbitrary formulas as well as classes of formulas defined in terms of relevant syntactic restrictions. Moreover, a reasoner for $LTL_f$ on weighted traces is presented, and its performance are assessed on benchmark data.

## 1 Introduction

*Linear Temporal Logic over finite traces* (short: $LTL_f$) is a logic formalism based on interpreting standard LTL formulas [Pnueli, 1977] over finite, yet unbounded, successions of states (see, e.g., [De Giacomo and Vardi, 2013]). The formalism has been recently attracting much attention in the AI literature, as it fits several scenarios where the traditional interpretation over infinite traces is not appropriate. For instance, $LTL_f$ has been exploited to formalize and validate the behavior of dynamical systems (e.g., [De Giacomo *et al.*, 2014b; De Giacomo and Vardi, 2015]), to reason about planning domains (e.g. [De Giacomo and Rubin, 2018; Zhu *et al.*, 2020; Xiao *et al.*, 2021]), and to declaratively specify, analyze and monitor business processes (e.g. [van der Aalst *et al.*, 2009; Montali *et al.*, 2010; De Giacomo *et al.*, 2014a]).

As a matter of facts, however, the usage of $LTL_f$ in real-world domains is still rather limited, despite a number of efficient reasoning engines have been already made available (see [Li *et al.*, 2020; Fionda and Greco, 2018]). Indeed, a major limitation of $LTL_f$ is that it cannot go beyond reasoning over qualitative properties of the environment, since it misses the ability to address certain quantitative aspects that are crucial in practical applications, such as time, probabilities, or costs related to realizing some desired behavior.

Clearly enough, $LTL_f$ inherits this limitation from LTL. In fact, the development of "quantitative" frameworks for LTL has been an active area of research in the last two decades. In particular, LTL extensions have been proposed within settings where constants take values from some given lattice and the semantics of Boolean and temporal operators has been recast as operations defined on that lattice [Kupferman and Lustig, 2007; Droste and Vogler, 2012]. Other works have instead studied LTL over weighted transition systems [Faella *et al.*, 2008; de Alfaro *et al.*, 2004; Droste and Rahonis, 2016; Baier *et al.*, 2014]. And, finally, a number of approaches have been defined to introduce quantitative features suited—for instance—to express a tolerance on the number of time instants where a formula that is always required to hold in the future actually does not hold [Kuperberg, 2014], to specify eventualities with a bounded wait time [Kupferman *et al.*, 2009; Alur *et al.*, 2001], to accommodate averaging [Bouyer *et al.*, 2014] or discounting [Mandrali, 2012; de Alfaro *et al.*, 2005] modalities, and to deal with propositional quality operators [Almagor *et al.*, 2016].

Compared to the above body of literature for LTL, considerably less attention has been paid to study $LTL_f$ in quantitative settings. Indeed, even though we might adapt to $LTL_f$ some of the extensions introduced for LTL (see, e.g., [Camacho *et al.*, 2018b]), the typical usage scenarios of $LTL_f$ call for ad-hoc solutions and approaches, which were missing in the literature so far. In the paper, we fill this gap by proposing and studying a "weighting" framework that is specific for $LTL_f$ and where quantitative aspects are encoded in the finite traces over which formulas are evaluated, rather than on the formulas themselves. Roughly, each variable is associated with a weight, and the weight of a state is given as a function of the weights of the variables evaluating true in it; the goal is, then, to compute an optimal model, i.e., such that the overall weight obtained by suitably aggregating the weights of the various (finitely many) states is minimized. In more details:

- We define the concept of *valuation structures* as mechanisms to flexibly equip each trace with a weight, and we identify and exemplify certain monotonicity properties of that structures guaranteing that *optimal models* always exist for satisfiable $LTL_f$ formulas (Section 2).

*Contact Author

- We depict a clear picture of the computational costs to be paid for the rich expressiveness of the resulting framework compared to standard LTL on finite traces, by considering arbitrary formulas as well as formulas belonging to some relevant syntactic fragments (Section 3).
- We illustrate the architecture of WELTL, a system prototype implemented to reason about LTL$_f$ over weighted traces exploiting our theoretical findings and a rewriting in terms of *Answer Set Programming (ASP)* [Brewka *et al.*, 2011]; and, finally, we assess and discuss its performance on some benchmark data (Section 4).[1]

## 2 Formal Framework

In this section, we introduce a framework for evaluating LTL$_f$ formulas on weighted finite traces, and we illustrate its salient features by discussing some example applications.

### 2.1 Basic Notions and Results

**LTL on finite traces.** Let $\mathcal{V}$ be a universe of propositional variables. An LTL$_f$ formula $\varphi$ over $\mathcal{V}$ is an expression[2] built according to the grammar $\varphi ::= \texttt{true} \mid \texttt{false} \mid x \mid \neg x \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid \mathsf{X}(\varphi) \mid \mathsf{X_w}(\varphi) \mid (\varphi \,\mathsf{U}\, \varphi) \mid (\varphi \,\mathsf{R}\, \varphi)$, where: $x \in \mathcal{V}$; "$\wedge$", "$\vee$", and "$\neg$" are the standard Boolean connectives; and "$\mathsf{X}$" (next), "$\mathsf{X_w}$" (weak next), "$\mathsf{U}$" (until), and "$\mathsf{R}$" (release) are temporal operators. A *finite trace* over $\mathcal{V}$ is a sequence $\pi = \pi_0, \pi_1, ..., \pi_{n-1}$ such that $\pi_i \subseteq \mathcal{V}$, for each $i \in \{0, ..., n-1\}$. We say that the formula $\varphi$ holds in $\pi$ at the *instant* $i \in \{0, ..., n-1\}$, denoted by $\pi, i \models \varphi$, by inductively considering *subformulas* as follows:

| | |
|---|---|
| $\pi, i \models y$ | iff $y \in \pi_i$ or $y = \texttt{true}$; |
| $\pi, i \models \neg y$ | iff $y \notin \pi_i$ or $y = \texttt{false}$; |
| $\pi, i \models (\varphi_1 \wedge \varphi_2)$ | 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 $i < n-1$ and $\pi, i+1 \models \varphi'$; |
| $\pi, i \models \mathsf{X_w}(\varphi')$ | iff $\pi, i \models \mathsf{X}(\varphi')$ or $i = n-1$; |
| $\pi, i \models (\varphi_1 \,\mathsf{U}\, \varphi_2)$ | iff $\exists j \geq i$ such that $\pi, j \models \varphi_2$ and $\forall k$ with $i \leq k < j$ we have $\pi, k \models \varphi_1$; |
| $\pi, i \models (\varphi_1 \,\mathsf{R}\, \varphi_2)$ | iff $\forall i \leq j < n$ it holds $\pi, j \models \varphi_2$; or $\exists j \geq i$ such that $\pi, j \models \varphi_1$ and $\forall i \leq k \leq j$ we have $\pi, k \models \varphi_2$; |

Whenever $\pi, 0 \models \varphi$ holds, then we say that $\pi$ is a *model* of $\varphi$ and we just write $\pi \models \varphi$.

**Example 2.1** Consider the LTL$_f$ formula $\varphi = (a \vee \mathsf{X}(b)) \wedge (c \,\mathsf{U}\, (d \wedge \mathsf{X_w}(a)))$ defined over the set $\mathcal{V} = \{a, b, c, d\}$ of propositional variables. The finite trace $\pi = \{a, d\}$ (consisting of one time instant only) is a model of $\varphi$; indeed, $\pi, 0 \models a \vee \mathsf{X}(b)$ holds because $a \in \pi_0$, while $\pi, 0 \models c \,\mathsf{U}\, (d \wedge \mathsf{X_w}(a))$ holds because $d \in \pi_0$ and there is no subsequent time instant (hence, $\mathsf{X_w}(a)$ holds too). Other models are $\pi' = \{c\}, \{b, d\}$, $\pi'' = \{c\}, \{c, b\}, \{d\}$, and $\pi''' = \{c\}, \{c, b\}, \{d\}, \{a\}$. Instead, note that the trace $\{b, c\}, \{a\}, \{d\}$ is not a model. $\lhd$

**Valuation structures.** A *valuation structure* $\mathtt{v}$ on $\mathcal{V}$ is a tuple $\mathtt{v} = \langle \circ, *, w \rangle$, where $w : \mathcal{V} \mapsto \mathbb{Z}$ is a function mapping each variable $x \in \mathcal{V}$ to an integer $w(x)$, and where $\circ$ and $*$ are two binary operators defined over $\mathbb{Z}$, which are moreover commutative and associative. For any set $\{x_1, ..., x_m\} \subseteq \mathcal{V}$, we define $\mathtt{v}(\{x_1, ..., x_m\})$ as the value $w(x_1) \circ \cdots \circ w(x_m)$. Moreover, for a trace $\pi = \pi_0, \pi_1, ..., \pi_{n-1}$, its $\mathtt{v}$-*weight* is defined as $\mathtt{v}(\pi) = \mathtt{v}(\pi_0) * \cdots * \mathtt{v}(\pi_{n-1})$.

**Example 2.2** Consider again the setting of Example 2.1, and the valuation structure $\mathtt{v} = \langle +, \times, w \rangle$ where $w(a) = 6$, $w(b) = 2$, $w(c) = 2$, and $w(d) = 1$. Then, we have $\mathtt{v}(\pi) = \mathtt{v}(\{a, d\}) = w(a) + w(d) = 7$. Furthermore, we have that $\mathtt{v}(\pi') = \mathtt{v}(\pi'_0) \times \mathtt{v}(\pi'_1) = 6$, $\mathtt{v}(\pi'') = \mathtt{v}(\pi''_0) \times \mathtt{v}(\pi''_1) \times \mathtt{v}(\pi''_2) = 8$, and $\mathtt{v}(\pi''') = \mathtt{v}(\pi'') \times w(a) = 48$. $\lhd$

Valuation structures are used to single out the "best" models by just looking for minimizing their associated weights. More formally, any model $\pi^*$ of $\varphi$ is called $\mathtt{v}$-*optimal* if there is no model $\pi$ of $\varphi$ with $\mathtt{v}(\pi) < \mathtt{v}(\pi^*)$.

**Example 2.3** In our running example, we have $\mathtt{v}(\pi') < \mathtt{v}(\pi)$, $\mathtt{v}(\pi') < \mathtt{v}(\pi'')$, and $\mathtt{v}(\pi') < \mathtt{v}(\pi''')$ hold. In fact, $\pi'$ is an optimal model. In particular, note that the models where $c$ holds at the first time instant, while $c$ plus $b$ hold at the second time instant have $\mathtt{v}$-weight at least $2 \times 4 = 8$. $\lhd$

**Monotone valuations.** Perhaps not surprisingly, optimal models are not guaranteed to exist, even ifad the LTL$_f$ formula given to hand is *satisfiable* (that is, if admits a model). For instance, consider the structure $\mathtt{v} = \langle +, +, -\mathbf{1} \rangle$ where $-\mathbf{1}$ is the constant function mapping every propositional variable $x \in \mathcal{V}$ to $-1$. Then, for the formula $\varphi = a$ and any model $\pi$ (of whatever length), the trace $\bar{\pi}$ obtained by appending the state $\{a\}$ to $\pi$ is again a model and $\mathtt{v}(\bar{\pi}) = \mathtt{v}(\pi) - 1 < \mathtt{v}(\pi)$.

Motivated by the above observation, we hereinafter restrict ourselves to valuation structures $\mathtt{v} = \langle \circ, *, w \rangle$, which we call *monotone*, enjoying the following properties:

**(P1)** $w(x) \geq 0$, for each $x \in \mathcal{V}$;

**(P2)** $w_1 \circ w_2 \geq \max\{w_1, w_2\}$, for each pair $w_1, w_2 \geq 0$; and

**(P3)** $w_1 * w_2 \geq \max\{w_1, w_2\}$, for each pair $w_1, w_2 \geq 0$.

The following is a simple consequence of monotonicity.

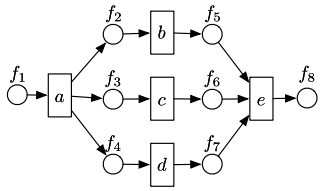
**Lemma 2.4** *Assume that* $\mathtt{v} = \langle \circ, *, w \rangle$ *is a monotone valuation structure, and let* $\pi_0, ..., \pi_{n-1}$ *be a trace. Then, for each pair* $0 \leq i < j \leq n-1$*, it holds* $\mathtt{v}(\pi_0, ..., \pi_i) \leq \mathtt{v}(\pi_0, ..., \pi_j)$.

**Theorem 2.5** *If* $\mathtt{v} = \langle \circ, *, w \rangle$ *is a monotone valuation structure and* $\varphi$ *is satisfiable, then* $\varphi$ *admits a* $\mathtt{v}$*-optimal model.*

**Proof (Sketch)**. Let $\Pi$ be the set of *irreducible* models of $\varphi$, that is, none of their prefixes is a model too. Since $\varphi$ is satisfiable, $\Pi \neq \emptyset$. Then, let $\pi^*$ be a model of minimum $\mathtt{v}$-weight over all models in $\Pi$. We claim that $\pi^*$ is $\mathtt{v}$-optimal. Indeed, assume by contradiction that $\pi = \pi_0, ..., \pi_{n-1}$ is a model with $\mathtt{v}(\pi) < \mathtt{v}(\pi^*)$. By construction of $\pi^*$, $\pi \notin \Pi$; hence, there is a time instant $i \in \{0, ..., n-2\}$ such that the prefix $\pi_0, ..., \pi_i$ belongs to $\Pi$, which implies $\mathtt{v}(\pi^*) \leq \mathtt{v}(\pi_0, ..., \pi_i)$. However, by Lemma 2.4, we know that $\mathtt{v}(\pi) \geq \mathtt{v}(\pi_0, ..., \pi_i)$, which entails $\mathtt{v}(\pi) \geq \mathtt{v}(\pi^*)$, which is impossible. $\square$

[1] WELTL, the benchmark data and details on the (theoretical/experimental) results are available as Supplementary Material.

[2] W.l.o.g., we consider formulas in *negated normal form*, i.e., negations are in front of propositional variables only.




| | $\langle +, +, c\rangle$ | $\langle +, \max, r\rangle$ | $\langle \max, +, t\rangle$ | $\langle \max, \max, t\rangle$ |
|---|---|---|---|---|
| $\{f_1, a\}, \{f_2, f_3, f_4, b, c, d\}, \{f_5, f_6, f_7, e\}, \{f_8\}$ | 5 | 4 | 3 | 1 |
| $\{f_1, a\}, \{f_2, f_3, f_4, b\}, \{f_3, f_4, f_5, c\}, \{f_4, f_5, f_6, d\}, \{f_5, f_6, f_7, e\}, \{f_8\}$ | 5 | 2 | 5 | 1 |
| $\{f_1, a\}, \{f_2, f_3, f_4, b, c\}, \{f_4, f_5, f_6, d\}, \{f_5, f_6, f_7, e\}, \{f_8\}$ | 5 | 2 | 4 | 1 |

Figure 1: The planning problem P, three models of $\varphi_{\mathrm{P}}$, and their values under the valuations structures built over "$+$" and "$\max$".

### 2.2 Examples of Monotone Valuations

Noticeable monotone valuations can be defined over "$+$" and "$\max$", e.g. to encode cost-related planning problems. Consider for instance the planning problem P in Figure 1: a set $\mathcal{F} = \{f_1, ..., f_8\}$ of fluents is given and, by starting from the scenario where $f_1$ holds, our goal is to end up with $f_8$, by applying, possibly in parallel, some of the available actions in $\mathcal{A} = \{a, b, c, d, e\}$. An example solution is the plan where we first execute $a$ ("consuming" $f_1$ and "producing" $f_2$, $f_3$, and $f_4$); then, we execute in parallel actions $b$, $c$, and $d$; and, finally, we execute $e$. It is well-known (see, e.g., [De Giacomo and Vardi, 2013]) that planning problems P can be easily encoded via LTL$_f$ formulas $\varphi_{\mathrm{P}}$ over propositional variables in $\mathcal{A} \cup \mathcal{F}$ so that models for $\varphi_{\mathrm{P}}$ one-to-one correspond with plans for P—at each time instant, we just keep trace of the fluents that hold and of the actions that are executed (see again Figure 1, reporting the model corresponding to the plan discussed above, plus two further models). We next enrich $\varphi_{\mathrm{P}}$ with valuations suited to encode cost-related variants.

**$\langle +, +, c\rangle$.** Assume that each action $x \in \mathcal{A}$ is associated with a cost $c(x)$, and consider the structure $\langle +, +, c\rangle$, where $c(y) = 0$ holds, for each fluent $y \in \mathcal{F}$. Then, $\langle +, +, c\rangle$-optimal models correspond to plans whose total cost is minimized over all the possible plans. For instance, by assuming that all actions cost 1, every model of $\varphi_{\mathrm{P}}$ is also an optimal one, since all plans reaching the goal have the same cost (namely, 5)—for they need to execute all given actions.

**$\langle +, \max, r\rangle$.** Suppose that each action $x \in \mathcal{A}$ is associated with the number $r(x)$ of employees required for its execution. If it is necessary, in time of a pandemic, to limit the number of employees that are simultaneously in the office, then we can consider the valuation structure $\langle +, \max, r\rangle$ (with $r(y) = 0$, for each $y \in \mathcal{F}$). For instance, in the setting of Figure 1, by assuming that all actions require 1 employee only, except $d$ requiring 2 employees, we get that $\langle +, \max, r\rangle$-optimal models of $\varphi_{\mathrm{P}}$ correspond to plans where $d$ is executed alone.

**$\langle \max, +, t\rangle$.** Consider a setting where each action $x \in \mathcal{A}$ is associated with the time $t(x)$ required for its execution. If we are interested in minimizing the total time for plan execution, then we can consider the valuation structure $\langle \max, +, t\rangle$ (with $t(y) = 0$, for each $y \in \mathcal{F}$)—in particular, note that at each step, for the actions that are executed in parallel, we just consider the maximum time over all of them. In our running example, by assuming that every action requires a unit of time, the only $\langle \max, +, t\rangle$-optimal model of $\varphi_{\mathrm{P}}$ corresponds to the plan where $b$, $c$, and $d$ are executed in parallel.

**$\langle \max, \max, t\rangle$.** Assume, finally, that we want to fix the duration of the work shift, by computing the minimum amount of time needed to complete each step. This can be modeled via the structure $\langle \max, \max, t\rangle$, so that the total cost is now the maximum completion time over all the steps. For instance, by assuming again that every action requires a unit of time, every model is an optimal one and its cost is 1.

## 3 Complexity of LTL$_f$ on Weighted Traces

In this section, we study the computational complexity of reasoning about LTL$_f$ on weighted traces. Formally, we consider the $\exists$MODEL problem that, given as input an integer $\tau \geq 0$, an LTL$_f$ formula $\varphi$, and a structure $\mathtt{v} = \langle \circ, *, w\rangle$, asks whether there is a model $\pi$ of $\varphi$ such that $\mathtt{v}(\pi) \leq \tau$.

In the analysis, we assume a standard encoding for the input; in particular, v is given by explicitly listing all elements in its domain with their associated values, while $\circ$ and $*$ are polynomial-time computable and produce outputs having size polynomial w.r.t. the input size. Hereinafter, the size of $\varphi$, i.e., the total number of temporal operators, Boolean connectives and propositional variables it contains, is denoted by $||\varphi||$.

### 3.1 Basic Results: Weighted Transition Graph

To prove our basic complexity results, we introduce the notion of *weighted transition graph*, which we formalize in the spirt of the transition system defined by [Li *et al.*, 2020].

An LTL$_f$ formula is in *neXt Normal Form* (XNF) if all occurrences of the temporal operators $\mathsf{R}$ and $\mathsf{U}$ are nested within the scope of some $\mathsf{X}$ or $\mathsf{X_w}$ operator. Note that there is a linear-time conversion from any formula $\varphi$ into an equivalent XNF one, denoted as $\mathtt{xnf}(\varphi)$; indeed, we can build $\mathtt{xnf}(\varphi)$ from $\varphi$ by just applying the following transformations:

- $\varphi_1 \,\mathsf{U}\, \varphi_2$ is replaced by $\varphi_2 \vee (\varphi_1 \wedge \mathsf{X}(\varphi_1 \,\mathsf{U}\, \varphi_2))$;
- $\varphi_1 \,\mathsf{R}\, \varphi_2$ is replaced by $\varphi_2 \wedge (\varphi_1 \vee \mathsf{X_w}(\varphi_1 \,\mathsf{R}\, \varphi_2))$.

Given $\mathtt{xnf}(\varphi)$, we then define $\mathcal{V}_{\mathsf{X}}(\varphi)$ (resp., $\mathcal{V}_{\mathsf{X_w}}(\varphi)$) as the set of all its $\mathsf{X}$-subformulas (resp., $\mathsf{X_w}$-subformulas) that are not nested within the scope of some other $\mathsf{X}$ or $\mathsf{X_w}$ operator; moreover, let $\mathtt{xnf}(\varphi)^p$ denote $\mathtt{xnf}(\varphi)$ viewed as a propositional formula over $\mathcal{V} \cup \mathcal{V}_{\mathsf{X}}(\varphi) \cup \mathcal{V}_{\mathsf{X_w}}(\varphi)$.

**Example 3.1** For the formula $\varphi$ of Example 2.1, we have

$$\mathtt{xnf}(\varphi) = (a \vee \mathsf{X}(b)) \wedge ((d \wedge \mathsf{X_w}(a)) \vee (c \wedge \mathsf{X}(c \,\mathsf{U}\, (d \wedge \mathsf{X_w}(a))))).$$

So, $\mathcal{V}_{\mathsf{X}}(\varphi) = \{\mathsf{X}(b), \mathsf{X}(c \,\mathsf{U}\, (d \wedge \mathsf{X_w}(a)))\}$ and $\mathcal{V}_{\mathsf{X_w}}(\varphi) = \{\mathsf{X_w}(a)\}$. A model[3] of $\mathtt{xnf}(\varphi)^p$ is $\sigma = \{\mathsf{X}(b), d, \mathsf{X_w}(a)\}$. $\lhd$

Let $\sigma$ be a model of $\mathtt{xnf}(\varphi)^p$. If $\sigma \cap \mathcal{V}_{\mathsf{X}}(\varphi) = \emptyset$, then we define $\sigma|_{\mathsf{X}} = \mathtt{end}$, where end is a special symbol not occurring in $\mathcal{V}$; otherwise, we define:

[3] Models of the propositional formula $\mathtt{xnf}(\phi)^p$ are sets of propositional variables (or, equivalently, traces with one time instant).

$$\sigma|_{\mathsf{x}} = \mathtt{xnf}\Big(\bigwedge_{\mathsf{X}(\psi)\in\sigma\cap\mathcal{V}_{\mathsf{X}}(\varphi)} \psi \wedge \bigwedge_{\mathsf{X}_{\mathsf{w}}(\psi)\in\sigma\cap\mathcal{V}_{\mathsf{X}_{\mathsf{w}}}(\varphi)} \psi\Big).$$

Intuitively, $\sigma|_{\mathsf{x}}$ is the XNF formula that is required to hold in the next time instant (actually being $\mathtt{end}$, if no further time instant is required), when $\sigma$ holds at the present instant; in fact, $\mathsf{X}_{\mathsf{w}}$ subformulas are only considered if a subsequent time instant is required to exist due to some $\mathsf{X}$ subformula. Then, the weighted transition graph just unfolds the formula $\varphi$, via XNF transformations, by looking for convergence to $\mathtt{end}$.

**Definition 3.2** The weighted transition graph $\mathtt{wtg}(\varphi, \mathtt{v})$ is the tuple $\langle N, E, \lambda_E, \mathtt{v}_E\rangle$ where: $(N, E)$ is a directed multigraph over XNF formulas; $\lambda_E : E \mapsto 2^{\mathcal{V}}$; and $\mathtt{v}_E : E \mapsto \mathbb{Z}$. In particular: $\mathtt{xnf}(\varphi)$ is in $N$; for each $\bar{\varphi} \in N \setminus \{\mathtt{end}\}$ and *minimal* model $\sigma$ for $\bar{\varphi}^p$, $\sigma|_{\mathsf{x}}$ is in $N$ and the edge $e = (\bar{\varphi}, \sigma|_{\mathsf{x}})$ is in $E$, with $\lambda_E(e) = \sigma \cap \mathcal{V}$ and $\mathtt{v}_E(e) = \mathtt{v}(\sigma \cap \mathcal{V})$; no further node (resp., edge) is in $N$ (resp., $E$). □

As an example, consider Figure 2 and note that the minimum-weighted path from $\varphi$ to $\mathtt{end}$ corresponds to the $\mathtt{v}$-optimal model $\pi'$ in Example 2.3. This is not by chance. Indeed, by letting $\Sigma_i$ be the set of all minimal models of $\varphi_i^p$, the following lemma derives by observing that $\varphi_i$ can be rewritten as $\bigvee_{\sigma\in\Sigma_i}\left(\bigwedge_{x\in\sigma\cap\mathcal{V}} x \wedge f(\sigma|_{\mathsf{x}})\right)$, where $f$ is such that $f(\mathtt{end}) = \mathtt{true}$ and $f(\Psi) = \mathsf{X}(\Psi)$, for each $\Psi \neq \mathtt{end}$.

**Lemma 3.3** *Let $\mathtt{wtg}(\varphi, \mathtt{v})$ be the weighted transition graph associated with $\varphi$ and $\mathtt{v} = \langle \circ, *, w\rangle$. Then,*

*(1) If there is a path $\varphi_0, \varphi_1, ..., \varphi_n$ with $\varphi_n = \mathtt{end}$, then $\lambda_E(\varphi_0, \varphi_1), ..., \lambda_E(\varphi_{n-1}, \varphi_n)$ is a model of $\varphi_0$;*

*(2) If $\pi^*$ is a $\mathtt{v}$-optimal model of $\varphi$, then there is a simple path $\varphi_0, \varphi_1, ..., \varphi_n$, such that $\varphi_0 = \varphi$, $\varphi_n = \mathtt{end}$, and*

*(2.a) $\lambda_E(\varphi_0, \varphi_1), ..., \lambda_E(\varphi_{n-1}, \varphi_n) = \pi^*$;*

*(2.b) $\mathtt{v}_E(\varphi_0, \varphi_1) * \cdots * \mathtt{v}_E(\varphi_{n-1}, \varphi_n) = \mathtt{v}(\pi^*)$.*

Now, it is routine that we can check in PSPACE whether there is a path from $\varphi$ to $\mathtt{end}$ corresponding to a model whose $\mathtt{v}$-weight is $\tau$ at most: we can build the path nondeterministically (recall that NPSPACE = PSPACE), and we identify a failure when either the weight of the current path exceeds $\tau$, or its length exceeds the overall number of nodes (with this number being at most exponential in $||\varphi||$ and, therefore, requiring polynomial space for its encoding).

**Theorem 3.4** ∃MODEL *is in* PSPACE.

However, the bound can be improved if we impose syntactic restrictions on the allowed operators. Formally, for any set $T \subseteq \{\mathsf{X}, \mathsf{X}_{\mathsf{w}}, \mathsf{U}, \mathsf{R}\}$, we define $\langle T\rangle$-LTL$_f$ as the class of all LTL$_f$ formulas where only operators in $T$ can be used.

**Theorem 3.5** ∃MODEL *is in* NP*, if restricted on* $\langle T\rangle$-LTL$_f$ *formulas, with* $T \in \{\{\mathsf{X}, \mathsf{X}_{\mathsf{w}}\}, \{\mathsf{X}_{\mathsf{w}}, \mathsf{R}\}\}$.

**Proof (Sketch)**. For a formula $\varphi \in \langle\{\mathsf{X}, \mathsf{X}_{\mathsf{w}}\}\rangle$-LTL$_f$, it is immediate to check that the length of the longest simple path in $\mathtt{wtg}(\varphi, \mathtt{v})$ from $\varphi$ to $\mathtt{end}$ is bounded by the maximum number of nested $\mathsf{X}$ operators, which is clearly polynomial in $||\varphi||$. For $\langle\{\mathsf{X}_{\mathsf{w}}, \mathsf{R}\}\rangle$-LTL$_f$ formulas, instead, just check that all edges outgoing from $\varphi$ are converging into $\mathtt{end}$. Therefore, on these fragments (and with monotone valuations $\mathtt{v}$), $\mathtt{v}$-optimal models have polynomial length. Thus, we can solve ∃MODEL by guessing a trace with a nondeterministic Turing machine and by then checking in polynomial time whether its $\mathtt{v}$-value is below the given threshold. □

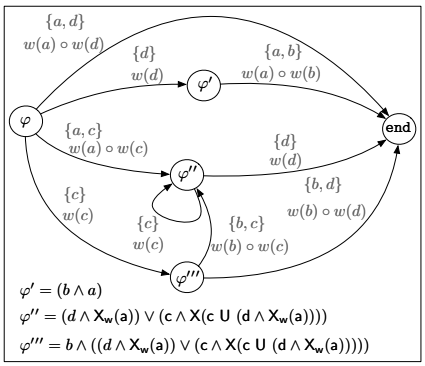



Figure 2: The weighted transition graph $\mathtt{wtg}(\varphi, \mathtt{v})$ built for the formula $\varphi = (a \vee X(b)) \wedge (c\,\mathsf{U}\,(d \wedge \mathsf{X}_{\mathsf{w}}(a)))$ and $\mathtt{v} = \langle\circ, *, w\rangle$.

## 3.2 A Closer Look at LTL$_f$ Fragments

According to Theorem 3.4, it seems that the expressiveness of the weighted framework comes at no cost (satisfiability for LTL$_f$ is PSPACE-complete [De Giacomo and Vardi, 2013]). We next embark on a finer-grained complexity analysis, and we show that actually this is not the case. In particular, we focus on valuation structures built over the operators "+" and "max", and we consider formulas with arbitrary Boolean connectives as well as *negation-free* formulas. A summary of our complexity results is depicted in Figure 3.

**Theorem 3.6** *All results in Figure 3 hold.*

**Proof (Idea).** For the case of arbitrary formulas, all hardness results that are known to hold for LTL$_f$ are clearly inherited by our more expressive setting. The corresponding membership results come from Theorem 3.4 and Theorem 3.5.

For negation-free formulas, we distinguish three cases. For the polynomial-time results, we define $\mathcal{V}' = \{x \in \mathcal{V} \mid w(x) \leq \tau\}$, and we can observe that ∃MODEL has a solution if, and only if, $\varphi$ admits a model over $\mathcal{V}'$—with this latter task being feasible in polynomial time after the results for negation-free formulas (without valuation structures).

For the various NP-hardness results, we can exhibit (different kinds of) reductions from the well-known *vertex cover* problem on undirected graphs $G = (N, E)$. For instance, for the case $\langle +, *, w\rangle$ (even without temporal operators), we can view each node in $N$ as a propositional variable with a unitary $\mathtt{v}$-weight, and define the formula $\varphi = \bigwedge_{(x,y)\in E}(x \vee y)$. Then, there is a vertex cover with cardinality $k$ at most if, and only if, there is a model for $\varphi$ whose $\mathtt{v}$-weight is $k$ at most.

Finally, PSPACE-hardness results can be shown by exploiting the valuation structures to simulate the possibility of negating some variables. As an example, let $\varphi$ be an arbitrary $\langle T\rangle$-LTL$_f$ formula, with $\{\mathsf{U}\} \subseteq T$ or $\{\mathsf{X}, \mathsf{R}\} \subseteq T$. Consider $\psi = \bigwedge_{x\in\mathcal{V}}(x \vee \bar{x})$, and the formula $\bar{\varphi}$ derived from $\varphi$ by: (1)

| OPERATORS | ARBITRARY FORMULAS | | NEGATION-FREE FORMULAS | | | | |
|---|---|---|---|---|---|---|---|
| | PLAIN | $\circ, *$ | PLAIN | $+, +$ | $+, \max$ | $\max, +$ | $\max, \max$ |
| $T \subseteq \{\mathsf{X_w}, \mathsf{R}\}$ | NP-c | NP-c | in P | NP-c | NP-c | in P | in P |
| $\{\mathsf{X}\} \subseteq T \subseteq \{\mathsf{X_w}, \mathsf{X}\}$ | NP-c | NP-c | in P | NP-c | NP-c | NP-c | in P |
| $\{\mathsf{U}\} \subseteq T \subseteq \{\mathsf{X_w}, \mathsf{U}, \mathsf{R}\}$ | PSPACE-c | PSPACE-c | in P | PSPACE-c | PSPACE-c | in P | in P |
| $\{\mathsf{X}, \mathsf{U}\} \subseteq T$ or $\{\mathsf{X}, \mathsf{R}\} \subseteq T$ | PSPACE-c | PSPACE-c | in P | PSPACE-c | PSPACE-c | PSPACE-c | in P |

Figure 3: Summary of complexity results in Section 3 for weighted LTL$_f$—the column PLAIN reports the results that were already known to hold for LTL$_f$ without weights (see [Fionda and Greco, 2018]).

replacing every occurrence of $\neg x$, for each $x \in \mathcal{V}$, with $\bar{x}$; and (2) replacing every subformula $\varphi_s$ with $\varphi_s \wedge \psi$. Consider then the valuation $\mathtt{v} = \langle +, \max, w \rangle$ where $w(x) = w(\bar{x}) = 1$, for each $x \in \mathcal{V}$. It can be checked that $\varphi$ is satisfiable if, and only if, $\bar{\varphi}$ admits a model $\pi$ such that $\mathtt{v}(\pi) \leq |\mathcal{V}|$. □

## 4 System Design and Implementation

The notion of weighted transition graph and its properties (cf. Lemma 3.3) provided us with an immediate approach to implement a prototype for LTL$_f$ over weighted traces, which we named WELTL. At a high level, it consists of two modules:

WELTL$_{df}$: The module builds in a depth-first manner the weighted transition graph, by checking for a model/path whose weight is below the desired threshold. Moreover, as we deal with monotone structures, it avoids exploring paths whose current weight exceeds that threshold.

WELTL$_{bf}$: The module uses a complementary breadth-first approach, which is better suited to be supported via a mechanism for implicit navigation, that we implemented in terms of a rewriting in *Answer Set Programming*.

In our experiments, we have found that WELTL$_{df}$ is more effective to identify unsatisfiable instances, while WELTL$_{bf}$ can be more practical to compute models of satisfiable formulas. With this observation to hand, we have coupled the two approaches. Since the design of WELTL$_{df}$ poses no special challenges (indeed, we adapted the reasoner presented by [Li *et al.*, 2020]), we next focus on the description of WELTL$_{bf}$, by subsequently presenting results of experimental activity.

### 4.1 WELTL$_{bf}$ (and ASP in a Nutshell)

The architecture of WELTL$_{bf}$ is depicted in Figure 4. Given a formula $\varphi$, it starts by computing a model $\pi$ of minimum length, or immediately checks that the formula is not satisfiable—this task is delegated to the model checker made available by [Fionda and Greco, 2018]. If $\mathtt{v}(\pi) \leq \tau$, then we are done. Otherwise, WELTL$_{bf}$ starts its (implicit) breadth-first exploration by looking for a model whose length $\ell$ coincides with the length of $\pi$ and such that the corresponding value is $\tau$ at most. This is carried out by leveraging a rewriting[4] in terms of an Answer Set Program $\mathtt{P}(\varphi, \ell, w)$.

For the sake of completeness, we recall that an ASP program [Brewka *et al.*, 2011] is made of (a combination of): *(i)* facts of the form "$head$."; *(ii)* normal rules of the form "$head$ :– $body$."; *(iii)* choice rules of the form "$\{head\}$ :– $body$."; *(iv)* constraints of the form " :– $body$.", where $head$ is an atom and $body$ is a set of (possibly negated)

[4]Details on the encoding are in the Supplementary Material.

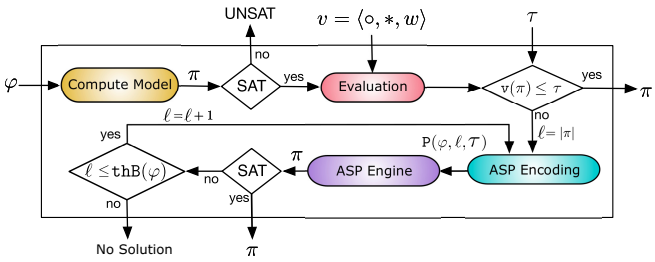



Figure 4: Conceptual architecture of WELTL$_{bf}$.

atoms, also including aggregate functions, such as *#sum* or *#max* [Faber *et al.*, 2011]. Atoms can be made over variables (strings starting with uppercase letter) or constants (non-negative integers or strings starting with lowercase letters). The semantics is given in terms of its *answer sets*, that is, sets $A$ of ground atoms such that: *(i)* $head$ is in $A$; *(ii)* whenever the $body$ is true, $head$ is in $A$; *(iii)* $head$ *can* be in $A$ whenever the $body$ is true; *(iv)* the $body$ must be false.

The ASP rewriting $\mathtt{P}(\varphi, \ell, \tau)$ uses the parse tree of $\varphi$: a fact $\mathrm{var}(v_i)$ is included, for each node $v$ and instant $i \in \{0, ..., \ell - 1\}$. Then, a choice rule ($\{\mathrm{selected}(X)\}$:-$\mathrm{var}(X)$.) is used to guess the subformulas of $\varphi$ (represented by the nodes in the parse tree) that hold at $i$; and, to check that such formulas actually hold, the definitions in Section 2 are straightforwardly translated in terms of constraints. Eventually, the ASP paradigm supports cost constraints that can encode the binary operators used in the valuation structures (in particular, they support "+" and "$\max$"), so that the program can also enforce that answer set programs are associated with traces whose v-weight is $\tau$ at most. This is next exemplified.

**Example 4.1** For an illustration of the rewriting, consider the formula $\varphi = a \wedge \mathsf{X}(b)$, and the structure $\mathtt{v} = \langle +, +, w \rangle$ where $w(a) = w(b) = 3$. Then, the ASP program $\mathtt{P}(\varphi, 2, \tau)$ is shown in Figure 5. The parse tree has 4 nodes, denoted as $\mathrm{and}$, $\mathrm{a}$, $\mathrm{x}$, and $\mathrm{b}$. Then, rule (1) guesses the atoms corresponding to these nodes and to the time instants 0 and 1. Rule (2) forces that $\mathrm{selected}(\mathrm{and}_0)$ holds, i.e., the whole formula holds at the initial time instant. Rules (3) and (4) define the semantics

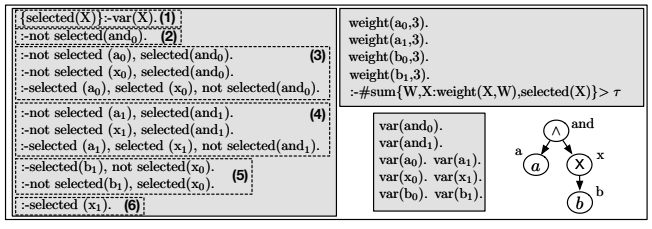



Figure 5: ASP encoding of Example 4.1.

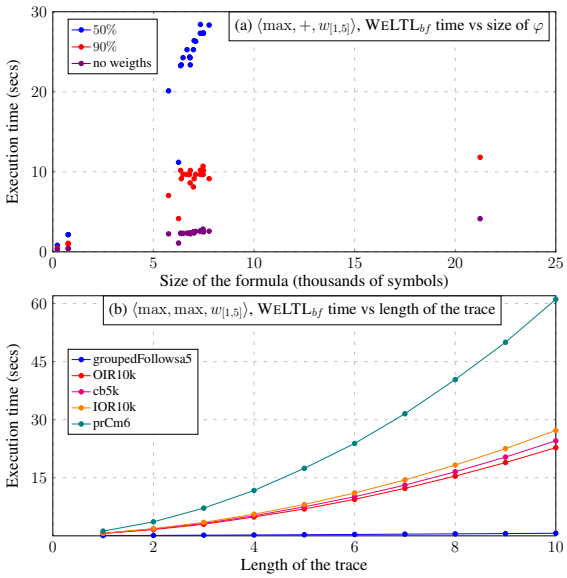



Figure 6: Execution times of over the benchmarks.

of the $\wedge$ connective. Rule (5) defines the semantics of the X operator; in particular, note that when $\mathrm{x}_0$ is selected, then $\mathrm{b}_1$ must be selected too. Finally, rule (6) prevents that $\mathrm{x}_1$ is selected, because we are considering $\ell = 2$ (and, hence, there is no subsequent time instant). The gray right box reports the rules encoding cost constraints. An atom $\mathrm{weight}(\alpha_i, w(\alpha))$ is included, for each propositional variable $\alpha$ and time instant $i$. A cost constraint states that the sum of the weights must not exceeded $\tau$. It is immediate to check that there is no answer set (hence, trace) satisfying the above rules for $\tau$<6. $\triangleleft$

Finally, the ASP engine *clingo* [Gebser *et al.*, 2018] is executed on the rewriting $\mathtt{P}(\varphi, \ell, w)$ (see Figure 4). If no answer set with the desired length $\ell$ and weight is found, then we explore the successive level. The process ends if either we actually find a solution, or if $\ell$ exceeds the theoretical length $\mathtt{thB}(\varphi)$ of the longest path in the weighted transition graph, which is in general exponential w.r.t. $||\varphi||$, and polynomially bounded in some syntactic fragments (cf. Theorem 3.5).

## 4.2 Experimental Results

For the evaluation, we used a benchmark taken from the business process domain, namely a dataset of 109 satisfiable DECLARE formulas [van der Aalst *et al.*, 2009] obtained by mining all DECLARE constraints that hold on a number of logs made available by the IEEE Task-Force on Process Mining (http://datacentrum.3tu.nl). The DECLARE constraints are then translated into their corresponding LTL$_f$ formulas [van der Aalst *et al.*, 2009]. For each formula, we randomly generated the variable weights in the intervals $[1, 5]$ and $[1, 10]$ and considered all combinations of $\max$ and $+$ operators, thus ending up with a dataset of 872 weighted settings. Moreover, for each weighted setting, we considered four different cost bounds obtained as follows: *(i)* we compute an arbitrary model $\pi$; *(ii)* we build a trace $\pi'$ whose length $n$ is the same as $\pi$ and such that $\pi'_i$=$\mathcal{V}$ for each $i \in \{0, ..., n-1\}$; *(iii)* we set the cost bounds to the 25%, 50%, 75% and 90% of $\mathrm{v}(\pi')$. Clearly enough, for the reduced weight bounds, formulas quickly became unsatisfiable. Experiments have been carried out on a Intel CPU 2,4 GHz with 16GB RAM. Time and memory were limited to 600 seconds and 15GB, respectively. In particular, we first run WELTL$_{bf}$ (with 300 seconds of timeout); and if no solution has been returned, then we run WELTL$_{df}$ (again with the same timeout).

| $\langle \circ, *, w \rangle$ | Solved formulas | | | |
|---|---|---|---|---|
| | 25% | 50% | 75% | 90% |
| $\langle \max, \max, w_{[1,5]} \rangle$ | 24 (0) | 24 (0) | 24 (0) | 33 (14) |
| $\langle \max, \max, w_{[1,10]} \rangle$ | 24 (0) | 24 (0) | 25 (2) | 42 (25) |
| $\langle \max, +, w_{[1,5]} \rangle$ | 25 (2) | 96 (87) | 109 (105) | 109 (105) |
| $\langle \max, +, w_{[1,10]} \rangle$ | 26 (3) | 104 (94) | 109 (105) | 109 (107) |
| $\langle +, \max, w_{[1,5]} \rangle$ | 109 (106) | 109 (109) | 109 (109) | 109 (109) |
| $\langle +, \max, w_{[1,10]} \rangle$ | 109 (103) | 109 (109) | 109 (109) | 109 (109) |
| $\langle +, +, w_{[1,5]} \rangle$ | 109 (108) | 109 (109) | 109 (109) | 109 (109) |
| $\langle +, +, w_{[1,10]} \rangle$ | 109 (108) | 109 (109) | 109 (109) | 109 (109) |

Table 1: Summary of results. Columns "Solved formulas" report the number of formulas for which WELTL computes a solution. For the remaining formulas, WELTL terminates in timeout.

Table 1 reports a summary of the obtained results. For each valuation structure and each cost bound we report the number of formulas (out of 109) that are solved by WELTL within the time limit of 600 seconds. Each cell of the table reports inside the parenthesis the number of instances solved by WELTL$_{bf}$ during the first 300 seconds. All the instances solved by WELTL$_{bf}$ are satisfiable (i.e, a trace satisfying the cost bound has been found), while all the instances solved by WELTL$_{df}$ during the last 300 seconds are unsatisfiable.

To get further insights on WELTL$_{bf}$, we report in Figure 6 the running times obtained on some instances. In particular, Figure 6.(a) reports the time to solve some satisfiable formulas when no weights are considered (i.e., standard satisfiability) and according to the valuation structure $\langle \max, +, w_{[1,5]} \rangle$ and the weight bounds 50% and 90%, respectively. The 22 formulas reported in the chart are those for which the model found at 90% did not meet the weight requirement at 50% and required to search for additional models. It can be noted that the time grows with the size of the formula and by lowering the cost bound. Figure 6.(b) reports the time required by WELTL$_{bf}$ to check the existence of models for $\langle \max, \max, w_{[1,5]} \rangle$ and the bound 25% for 5 formulas, and by fixing different lengths $\ell$ in the encoding. The theoretical exponential dependency is nicely handled by the ASP engine.

# 5 Conclusion

A framework to express LTL$_f$ properties in quantitative settings has been proposed and studied, based on the concept of valuation structures as mechanisms to equip each trace with a weight. While our approach departs from most of the quantitative LTL (and LTL$_f$) extensions proposed in the literature, a perspective similar to ours has been taken previously by [Lahijanian *et al.*, 2015]—though in the specific context of temporal planning with partial satisfaction (and without providing any complexity analysis and system prototype). In particular, a clear picture of the complexity issues arising when reasoning over weighted finite traces has been depicted and an efficient reasoning engine has been made available. On the theoretical level, instead, it remains open to study LTL$_f$ over valuation structures that are not monotone, by characterizing its computational properties and by defining and implementing computation algorithms.

## References


[Almagor *et al.*, 2016] Shaull Almagor, Udi Boker, and Orna Kupferman. Formally reasoning about quality. *Journal of the ACM*, 63(3):24:1–24:56, 2016.

[Alur *et al.*, 2001] Rajeev Alur, Kousha Etessami, Salvatore La Torre, and Doron A. Peled. Parametric temporal logic for "model measuring". *ACM Trans. on Computional Logics*, 2(3):388–407, 2001.

[Baier *et al.*, 2014] Christel Baier, Joachim Klein, Sascha Klüppelholz, and Sascha Wunderlich. Weight monitoring with linear temporal logic: complexity and decidability. In *Proc. of CSL-LICS*, pages 11:1–11:10. ACM, 2014.

[Bouyer *et al.*, 2014] Patricia Bouyer, Nicolas Markey, and Raj Mohan Matteplackel. Averaging in LTL. In *Proc. of CONCUR*, pages 266–280, 2014.

[Brewka *et al.*, 2011] Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, 2011.

[Camacho *et al.*, 2018b] Alberto Camacho, Meghyn Bienvenu, and Sheila A. McIlraith. Finite LTL synthesis with environment assumptions and quality measures. In *Proc. of KR*, pages 454–463, 2018.

[de Alfaro *et al.*, 2004] Luca de Alfaro, Marco Faella, and Mariëlle Stoelinga. Linear and branching metrics for quantitative transition systems. In *Proc. of ICALP*, pages 97–109, 2004.

[de Alfaro *et al.*, 2005] Luca de Alfaro, Marco Faella, Thomas A. Henzinger, Rupak Majumdar, and Mariëlle Stoelinga. Model checking discounted temporal properties. *Theor. Comput. Sci.*, 345(1):139–170, 2005.

[De Giacomo and Vardi, 2013] Giuseppe De Giacomo and M. Y. Vardi. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *Proc. of IJCAI*, pages 854–860.

[De Giacomo *et al.*, 2014a] Giuseppe De Giacomo, Riccardo De Masellis, Marco Grasso, Fabrizio M. Maggi, and Marco Montali. Monitoring Business Metaconstraints Based on LTL and LDL for Finite Traces. In *Proc. of BPM*, pages 1–14, 2014.

[De Giacomo *et al.*, 2014b] Giuseppe De Giacomo, Riccardi De Masellis, and Marco Montali. Reasoning on LTL on Finite Traces: Insensitivity to Infiniteness. In *Proc. of AAAI*, pages 1027–1033, 2014.

[De Giacomo and Rubin, 2018] Giuseppe De Giacomo and Sasha Rubin. Automata-theoretic foundations of FOND planning for LTLf and LDLf goals. In *Proc. of IJCAI*, pages 4729–4735, 2018.

[De Giacomo and Vardi, 2015] Giuseppe De Giacomo and Moshe Y. Vardi. Synthesis for LTL and LDL on finite traces. In *Proc. of IJCAI*, pages 1558–1564, 2015.

[Droste and Rahonis, 2016] Manfred Droste and George Rahonis. Weighted linear dynamic logic. In *Proc. of EPTCS*, volume 226, pages 149–163, 2016.

[Droste and Vogler, 2012] Manfred Droste and Heiko Vogler. Weighted automata and multi-valued logics over arbitrary bounded lattices. *TCS*, 418:14–36, 2012.

[Faber *et al.*, 2011] Wolfgang Faber, Gerald Pfeifer, and Nicola Leone. Semantics and complexity of recursive aggregates in answer set programming. *Artif. Intell.*, 175(1):278–298, 2011.

[Faella *et al.*, 2008] Marco Faella, Axel Legay, and Mariëlle Stoelinga. Model checking quantitative linear time logic. *E. N. in Theor. Comput. Sci.*, 220(3):61–77, 2008.

[Fionda and Greco, 2018] Valeria Fionda and Gianluigi Greco. LTL on finite and process traces: Complexity results and a practical reasoner. *J. Artif. Intell. Res.*, 63:557–623, 2018.

[Gebser *et al.*, 2018] Martin Gebser, Nicola Leone, Marco Maratea, Simona Perri, Francesco Ricca, and Torsten Schaub. Evaluation techniques and systems for answer set programming: a survey. In *Proc. of IJCAI'18*, pages 5450–5456, 2018.

[Kuperberg, 2014] Denis Kuperberg. Linear temporal logic for regular cost functions. *Log. Methods Comput. Sci.*, 10(1), 2014.

[Kupferman and Lustig, 2007] Orna Kupferman and Yoad Lustig. Lattice automata. In *Proc. of VMCAI*, pages 199–213, 2007.

[Kupferman *et al.*, 2009] Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. From liveness to promptness. *Formal Methods in System Design*, 34(2):83–103, 2009.

[Lahijanian *et al.*, 2015] Morteza Lahijanian, Shaull Almagor, Dror Fried, Lydia E. Kavraki, and Moshe Y. Vardi. This time the robot settles for a cost: A quantitative approach to temporal logic planning with partial satisfaction. In *Proc. of AAAI*, pages 3664–3671, 2015.

[Li *et al.*, 2020] Jianwen Li, Geguang Pu, Yueling Zhang, Moshe Y. Vardi, and Kristin Y. Rozier. Sat-based explicit ltlf satisfiability checking. *Artif. Intell.*, 289:103369, 2020.

[Mandrali, 2012] Eleni Mandrali. Weighted LTL with discounting. In *Proc. of CIIA'12*, pages 353–360, 2012.

[Montali *et al.*, 2010] Marco Montali, Maja Pesic, Wil M. P. van der Aalst, Federico Chesani, Paola Mello, and Sergio Storari. Declarative specification and verification of service choreographiess. *ACM Trans. on the Web*, 4(1):3:1–3:62, 2010.

[Pnueli, 1977] Amir Pnueli. The Temporal Logic of Programs. In *Proc. of FOCS*, pages 46–57, 1977.

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

[Xiao *et al.*, 2021] Shengping Xiao, Jianwen Li, Shufang Zhu, Yingying Shi, Geguang Pu, and Moshe Y. Vardi. On-the-fly synthesis for LTL over finite traces. In *Proc. of AAAI*, pages 6530–6537, 2021.

[Zhu *et al.*, 2020] Shufang Zhu, Giuseppe De Giacomo, Geguang Pu, and Moshe Y. Vardi. Ltl$f$ synthesis with fairness and stability assumptions. In *Proc. of AAAI*, pages 3088–3095, 2020.