Linear Temporal Logic Modulo Theories over Finite Traces

Luca Geatti , Alessandro Gianola , Nicola Gigante
Free University of Bozen-Bolzano, Italy
{geatti,gianola,gigante}@inf.unibz.it

Abstract

This paper studies Linear Temporal Logic over Finite Traces (LTLf) where proposition letters are replaced with first-order formulas interpreted over arbitrary theories, in the spirit of Satisfiability Modulo Theories. The resulting logic, called LTLf Modulo Theories (LTLf$^{\mathsf{MT}}$), is semi-decidable. Nevertheless, its high expressiveness comes useful in a number of use cases, such as model-checking of data-aware processes and data-aware planning. Despite the general undecidability of these problems, being able to solve satisfiable instances is a compromise worth studying. After motivating and describing such use cases, we provide a sound and complete semi-decision procedure for LTLf$^{\mathsf{MT}}$ based on the SMT encoding of a one-pass tree-shaped tableau system. The algorithm is implemented in the BLACK satisfiability checking tool, and an experimental evaluation shows the feasibility of the approach on novel benchmarks.

1 Introduction

Linear Temporal Logic (LTL) [Pnueli, 1977] and LTL over finite traces (LTLf) [De Giacomo and Vardi, 2013] are common languages to express temporal properties in the fields of *formal verification* and AI. In particular, LTLf has recently gained traction in AI and *business process modeling* [De Giacomo *et al.*, 2014a]. In these fields, reasoning over finite traces is more natural.

However, in the modeling of *data-aware processes* [Calvanese *et al.*, 2020; Li *et al.*, 2017], the propositional nature of LTLf is a severe limitation. These are systems whose behavior depends on and/or manipulate a persistent data storage such as a relational database. In these contexts, one would like to express actions and constraints that depend on the contents of the database, which is an inherently first-order object. A similar use case is that of *data-aware* planning, *i.e.*, planning problems [Ghallab *et al.*, 2004] where conditions of actions depend on the contents of the persistent data storage.

In order to obtain the expressiveness needed to model and reason about these scenarios, this paper introduces and studies LTLf *Modulo Theory* (LTLf$^{\mathsf{MT}}$), a logic that extends LTLf by replacing propositional symbols with first-order formulas over an arbitrary theory, in the spirit of *Satisfiability Modulo Theory* (SMT). While LTLf$^{\mathsf{MT}}$ is easily seen to be undecidable in general, for decidable theories it is *semi-decidable*, *i.e.*, a positive answer can always be obtained for satisfiable instances, while reasoning might not terminate over unsatisfiable instances. We argue that, for complex problems and scenarios like those mentioned above, where reasoning is inherently undecidable anyway, being able to solve satisfiable instances is a compromise worth studying.

In particular, we provide a semi-decision procedure for LTLf$^{\mathsf{MT}}$ satisfiability based on the one-pass tree-shaped tableau for LTL by Reynolds [2016]. In contrast to classic graph-shaped tableaux for LTL, Reynolds' builds a tree structure where a single independent pass is sufficient to decide whether to accept or reject the current branch. Here, we adapt Reynolds' tableau to LTLf$^{\mathsf{MT}}$, showing that, for decidable underlying first-order theories, LTLf$^{\mathsf{MT}}$ is semi-decidable. While first-order extensions of LTL have already been studied before, our setting is more specific: on the one hand, we are more general than, *e.g.*, [Cimatti *et al.*, 2020] by supporting quantified formulas. On the other hand, we restrict general first-order LTL [Kontchakov *et al.*, 2004; Artale *et al.*, 2019] by not allowing temporal operators inside quantifiers.

Our main contribution is an SMT-based algorithmic technique for satisfiability checking of such an expressive logic, which we implement in the BLACK tool [Geatti *et al.*, 2019; Geatti *et al.*, 2021b]. BLACK is a recent state-of-the-art satisfiability checker for LTL and LTLf based on a SAT encoding of Reynolds' tableau. Inspired by that, we provide an SMT encoding of our tableau for LTLf$^{\mathsf{MT}}$: suitably encoded SMT formulas represent the branches of the tableau up to depth k, for increasing values of k.

An experimental evaluation assesses the applicability of our approach. With a number of novel benchmarks over different underlying theories, we show that BLACK is able to reason over satisfiable instances and, in some cases, over unsatisfiable instances as well, with promising performance.

The paper is organized as follows. We introduce LTLf$^{\mathsf{MT}}$ in Section 2. Then, Section 3 discusses some relevant use cases that motivate our work. Sections 4 and 5 provide a semi-decision procedure for LTLf$^{\mathsf{MT}}$ and its SMT encoding. Section 6 evaluates the implementation of this procedure in the BLACK satisfiability checker, and Section 7 concludes. Proofs can be found in [Geatti *et al.*, 2022].

2 LTL Modulo Theories over Finite Traces

$\mathsf{LTLf}^{\mathsf{MT}}$ extends Linear Temporal Logic over Finite Traces (LTLf) by replacing proposition letters with first-order formulas over arbitrary theories, similarly to how Satisfiability Modulo Theory (SMT) extends the classical Boolean satisfiability problem. Before going into the formal details of syntax and semantics, let us show some examples.

Consider the theory of *linear integer arithmetic* (LIA). This first-order theory predicates over the set of integer numbers $\mathbb{Z}$, and interprets the $+$ function symbol and the $\leq$ relation symbol as the usual sum and comparison between integer numbers. There is no multiplication symbol. With $\mathsf{LTLf}^{\mathsf{MT}}$ over LIA, we can express formulas such as the following:

$$\mathsf{G}(x = y + y) \quad (x < y)\,\mathsf{U}\,y = 0 \quad \mathsf{G}(x > 5) \wedge \mathsf{F}(x < 0)$$

This first formula on the left states that the interpretation of the variable x must always (*i.e.*, in any time point) be the double of the variable y. The second formula states that the variable x must remain less than y until the first time y becomes equal to zero. The third one, which is unsatisfiable, demands x to be always greater than 5 but also to become less than 0 in the future. Quantifiers are allowed, but *temporal operators* cannot be nested inside them. For example, the following formula states that x has always to be even:

$$\mathsf{G}(\exists y(x = y + y))$$

While these examples are interesting, there is often the need to go further, by relating the value of variables at a given time point to their value at the following instant. This is expressed by two term constructors, $\bigcirc x$ and $\circleddash x$, which both represent, in a different way, the value of the variable x at the next state. The difference lies in how these terms behave at the end of the finite trace. When a first-order formula contains some $\bigcirc x$, the next state is required to exist, while it is not required to exist if it contains only $\circleddash x$ terms. This replicates the difference between the *tomorrow* and *weak tomorrow* temporal operators in LTLf [De Giacomo and Vardi, 2013; De Giacomo *et al.*, 2014b]. With these two term constructors in place, we can express more interesting things such as:

$$x = 0 \wedge ((\bigcirc x = x + 1)\,\mathsf{U}\,x = 42)$$
$$x = 0 \wedge \mathsf{G}(\circleddash x > x \wedge \exists y(x = y + y))$$
$$y = 1 \wedge \mathsf{G}(\circleddash y = y + 1 \wedge x = y + y)$$

The first formula states that variable x behaves as a counter that increments until reaching $x = 42$. The second one makes x represent *any* strictly increasing sequence of even numbers, while in the third formula x takes *all* the first n even numbers for any $n \geq 1$. Note that, in the second and third formula, replacing $\circleddash x$ with $\bigcirc x$ would result in an unsatisfiable formula, because each time point would require the existence of the next one, which is impossible in a finite-trace semantics.

Syntax. We are now ready to delve into the details of syntax and semantics of $\mathsf{LTLf}^{\mathsf{MT}}$. Let us start from the syntax. We work with a multi-sorted first-order signature $\Sigma = \mathcal{S} \cup \mathcal{P} \cup \mathcal{C} \cup \mathcal{F} \cup \mathcal{V} \cup \mathcal{W}$, composed of a set of sort symbols $\mathcal{S}$, a set of predicate symbols $\mathcal{P}$, a set of constant symbols $\mathcal{C}$, a set of function symbols $\mathcal{F}$, a set of variable symbols $\mathcal{V}$, and a set of *quantified* variables $\mathcal{W}$. Each constant in $\mathcal{C}$ and each variable in $\mathcal{V}$ and $\mathcal{W}$ is associated with a sort symbol $S \in \mathcal{S}$, and so are the domains of each relation symbol and the domains and ranges of each function symbol.

A Σ-*term* t is generated by the following grammar:

$$t \coloneqq x \mid y \mid c \mid f(t_1, \ldots, t_k) \mid \bigcirc x \mid \circleddash x$$

where $x \in \mathcal{V}$ and $y \in \mathcal{W}$ are variables, $c \in \mathcal{C}$ is a constant symbol, $f \in \mathcal{F}$ is a function symbol of arity k, each t_i (for any $i \in \{1, \ldots, k\}$) is a Σ-term, and $\bigcirc$ and $\circleddash$ are the *next* and *weak next* constructors. Note that the distinction between variables in $\mathcal{V}$ and in $\mathcal{W}$ is needed since it does not make sense to apply $\bigcirc x$ and $\circleddash x$ to a quantified variable (recall that $\bigcirc x$ represents the value of x at the next state). The grammar of $\mathsf{LTLf}^{\mathsf{MT}}$ formulas over Σ is the following:

$$\begin{aligned}
&\alpha \coloneqq p(t_1, \ldots, t_k)\\
&\lambda \coloneqq \alpha \mid \neg\alpha \mid \lambda_1 \vee \lambda_2 \mid \lambda_1 \wedge \lambda_2 \mid \exists x\lambda \mid \forall x\lambda\\
&\phi \coloneqq \top \mid \lambda \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \mathsf{X}\phi \mid \widetilde{\mathsf{X}}\phi \mid \phi_1\,\mathsf{U}\,\phi_2 \mid \phi_1\,\mathsf{R}\,\phi_2
\end{aligned}$$

where $x \in \mathcal{W}$, $p \in \Sigma$ is an k-ary predicate symbol, each t_i is a Σ-term, and X, $\widetilde{\mathsf{X}}$, U, and R are the *tomorrow*, *weak tomorrow*, *until*, and *release* temporal operators, respectively. In the definition of ϕ, we force λ to not have free variables from $\mathcal{W}$. Formulas of type λ as defined above are called *first-order* formulas. We assume the usual shortcuts for temporal operators, such as $\mathsf{F}\phi \equiv \top\,\mathsf{U}\,\phi$ and $\mathsf{G}\phi \equiv \bot\,\mathsf{R}\,\phi$. Note that by the grammar above, $\mathsf{LTLf}^{\mathsf{MT}}$ formulas are always in *negated normal form*, for ease of exposition. The formulas of $\mathsf{LTLf}^{\mathsf{MT}}$ are assumed to be well-typed with regard to the sorts of all the involved symbols. One may also consider *past operators*, but we omit them here to ease exposition.

Semantics. We use the standard notion of *first-order* Σ-*structure* M over the first-order (multi-sorted) signature Σ, which consists of a domain and of an interpretation s^M of all sort, predicate, constant, and function symbols $s \in \Sigma$. In particular, sort symbols are interpreted as pairwise disjoint sets, whose union $\mathrm{dom}(M)$ is the *domain* of M. In line with [Barrett *et al.*, 2009], we define a *theory* $\mathcal{T}$ as a (finite or infinite) class of Σ-structures.

Let $\Sigma = \mathcal{S} \cup \mathcal{P} \cup \mathcal{C} \cup \mathcal{F} \cup \mathcal{V} \cup \mathcal{W}$ be a signature and let $\mathcal{T}$ be a theory. A $\mathcal{T}$-*state* $s = (M, \mu)$ is a pair made of a Σ-structure $M \in \mathcal{T}$ and a variable evaluation function $\mu : \mathcal{V} \to \mathrm{dom}(M)$ assigning to each variable in $\mathcal{V}$ a value in the domain of M.

A word $\sigma = \langle (M, \mu_0), \ldots, (M, \mu_{n-1}) \rangle$ over the theory $\mathcal{T}$ is a finite sequence of $\mathcal{T}$-states over the same first-order structure. We define $|\sigma| = n$. It is worth noticing that any two states can differ only in the variable evaluation functions, and *not* in their domain nor in the interpretations of the other symbols, which are rigidly interpreted. In the context of first-order LTL, this is called *constant domain semantics* [Hodkinson *et al.*, 2000].

Given a term t, a word $\sigma = \langle (M, \mu_0), \ldots, (M, \mu_{n-1}) \rangle$, an integer $0 \leq i < n$, and a variable evaluation function $\xi : \mathcal{W} \to \mathrm{dom}(M)$ for the variables in $\mathcal{W}$, the *evaluation* of t at the instant i on the trace σ with *environment* ξ, denoted $[\![t]\!]^i_{\sigma,\xi}$, is the following:

1. $[\![x]\!]^i_{\sigma,\xi} = \begin{cases} \mu_i(x) & \text{if } x \in \mathcal{V} \\ \xi(x) & \text{if } x \in \mathcal{W} \end{cases}$

2. $[\![\bigcirc x]\!]^i_{\sigma,\xi} = [\![\circleddash x]\!]^i_{\sigma,\xi} = \mu_{i+1}(x)$
3. $[\![c]\!]^i_{\sigma,\xi} = c^M$
4. $[\![f(t_1,\dots,t_k)]\!]^i_{\sigma,\xi} = f^M([\![t_1]\!]^i_{\sigma,\xi},\dots,[\![t_k]\!]^i_{\sigma,\xi})$

Intuitively, when evaluating a term, the free variables from $\mathcal{V}$ are evaluated according to the word σ, while the bound variables from $\mathcal{W}$ are evaluated according to the environment ξ. Note that $[\![t]\!]^i_{\sigma,\xi}$ is *well-defined* for σ only if (1) $i < |\sigma| - 1$ or (2) $i = |\sigma| - 1$ and t does not contain terms of type $\bigcirc x$ or $\circleddash x$. That is, we cannot evaluate a variable beyond the end of the word. A first-order formula λ is well-defined for σ only if $[\![t]\!]_{\sigma,\xi}$ is well-defined for all the terms t appearing in ψ. Given a variable evaluation function $\xi : \mathcal{W} \to \mathrm{dom}(M)$, we denote as $\xi[x \leftarrow v]$ the function that agrees with ξ except that $\xi(x) = v$.

Given a theory $\mathcal{T}$, the *satisfaction modulo $\mathcal{T}$* of a *first-order* formula ψ over the word σ at time point $i \in \mathbb{N}$ with environment ξ, denoted with $\sigma, \xi, i \models \psi$, is inductively defined as:

1. $\sigma, \xi, i \models p(t_1,\dots,t_k)$ is defined depending on whether terms of type $\bigcirc x$ appear in $t_1,\dots,t_k$:
   (a) if $\bigcirc x$ appear in $t_1,\dots,t_k$ for some variable x, then $\sigma,\xi,i \models p(t_1,\dots,t_k)$ iff $[\![t_1]\!]^i_{\sigma,\xi},\dots,[\![t_k]\!]^i_{\sigma,\xi}$ are well-defined and $([\![t_1]\!]^i_{\sigma,\xi},\dots,[\![t_k]\!]^i_{\sigma,\xi}) \in p^M$;
   (b) otherwise, $\sigma,\xi,i \models p(t_1,\dots,t_k)$ if and only if at least one in $[\![t_1]\!]^i_{\sigma,\xi},\dots,[\![t_k]\!]^i_{\sigma,\xi}$ is not well-defined or $([\![t_1]\!]^i_{\sigma,\xi},\dots,[\![t_k]\!]^i_{\sigma,\xi}) \in p^M$;
2. $\sigma,\xi,i \models \neg p(t_1,\dots,t_k)$ iff $\sigma,\xi,i \not\models p(t_1,\dots,t_k)$;
3. $\sigma,\xi,i \models \lambda_1 \vee \lambda_2$ iff $\sigma,\xi,i \models \lambda_1$ or $\sigma,\xi,i \models \lambda_2$;
4. $\sigma,\xi,i \models \lambda_1 \wedge \lambda_2$ iff $\sigma,\xi,i \models \lambda_1$ and $\sigma,\xi,i \models \lambda_2$;
5. $\sigma,\xi,i \models \exists x.\psi$ iff there exists a value $v \in \mathrm{dom}(M)$ such that $\sigma,\xi[x \leftarrow v],i \models \psi$;
6. $\sigma,\xi,i \models \forall x.\psi$ iff for all values $v \in \mathrm{dom}(M)$ it holds that $\sigma,\xi[x \leftarrow v],i \models \psi$;

Note that Items 1a and 1b above are those where the semantics of $\bigcirc x$ and $\circleddash x$ terms differ. In the first case, the next state must exist for the atom to hold. In the second case, the atom holds by definition if the next state does not exist. Given a theory $\mathcal{T}$, the *satisfaction modulo $\mathcal{T}$* of an $\mathsf{LTLf}^{\mathsf{MT}}$ formula ϕ over the word σ at time point $i \in \mathbb{N}$, denoted as $\sigma, i \models \phi$, is inductively defined as follows:

1. $\sigma, i \models \lambda$, where λ is a first-order formula, if $\sigma,\xi,i \models \lambda$ for some ξ;
2. $\sigma, i \models \phi_1 \vee \phi_2$ iff $\sigma, i \models \phi_1$ or $\sigma, i \models \phi_2$;
3. $\sigma, i \models \phi_1 \wedge \phi_2$ iff $\sigma, i \models \phi_1$ and $\sigma, i \models \phi_2$;
4. $\sigma, i \models \mathsf{X}\phi$ iff $i < |\sigma| - 1$ and $\sigma, i+1 \models \phi$;
5. $\sigma, i \models \widetilde{\mathsf{X}}\phi$ iff $i = |\sigma| - 1$ or $\sigma, i+1 \models \phi$;
6. $\sigma, i \models \phi_1 \mathsf{U} \phi_2$ iff there exists a $j \geq i$ such that $\sigma, j \models \phi_2$ and $\sigma, k \models \phi_1$ for all $i \leq k < j$;
7. $\sigma, i \models \phi_1 \mathsf{R}\, \phi_2$ iff either $\sigma, j \models \phi_2$ for all $i \leq j < |\sigma|$, or there exists $k \geq i$ such that $\sigma, k \models \phi_1$ and $\sigma, j \models \phi_2$ for all $i \leq j \leq k$.

We say that σ *satisfies ϕ modulo $\mathcal{T}$* iff $\sigma, 0 \models \phi$. The *language modulo $\mathcal{T}$* of ϕ is the set of words σ over $\mathcal{T}$ that satisfy ϕ. A formula ϕ with no temporal operators and no $\bigcirc x$ or $\circleddash x$ terms is a *purely first-order formula*. If a word σ satisfies such a ϕ, only the first state is involved in its satisfaction. In this case we can write $s \models \phi$. An atom $p(t_1,\dots,t_k)$ is *strong* if it contains at least a $\bigcirc x$ term. It is *weak* if it contains $\circleddash x$ terms but no $\bigcirc y$ terms.

The satisfiability checking problem for $\mathsf{LTLf}^{\mathsf{MT}}$ is easily seen to be undecidable, depending on the theory. For example, with the LIA theory one can easily encode the PLANEX-$(\mathcal{C}_{p+}, \mathcal{C}_{\varnothing}, \mathcal{E}_{+1})$ *numeric planning* problem, proved to be undecidable by Helmert [Helmert, 2002]. However, for suitable theories (see Section 4), it is semi-decidable.

Expressiveness. It is interesting at this point to wonder how much expressiveness we gain by extending the classical (propositional) LTLf [De Giacomo and Vardi, 2013; De Giacomo *et al.*, 2014b] in the way we previously shown. As LTLf is a propositional logic, it is trivially subsumed by $\mathsf{LTLf}^{\mathsf{MT}}$ with a theory $\mathcal{B}$ consisting only of the equality relation and a Boolean domain ($\{0,1\}$). One may wonder, however, how expressive $\mathsf{LTLf}^{\mathsf{MT}}$ over other theories is with regards to $\mathcal{B}$ (*i.e.*, to LTLf), when we abstract its models to propositional finite words. More formally, for a signature $\Sigma = \mathcal{S} \cup \mathcal{P} \cup \mathcal{C} \cup \mathcal{F} \cup \mathcal{V} \cup \mathcal{W}$ we consider *unary* predicates from $\mathcal{P}$ as propositional symbols, and given a word $\sigma = \langle s_0,\dots,s_{n-1}\rangle$ we define its *Boolean abstraction* $\mathbb{B}(\sigma) = \langle \mathbb{B}(s_0),\dots,\mathbb{B}(s_{n-1})\rangle$ as follows: $P \in \mathbb{B}(s_i)$ iff $s_i \models \exists x P(x)$. For any theory $\mathcal{T}$, the Boolean abstraction of a language modulo $\mathcal{T}$ is the set of the Boolean abstractions of all the words of the language. We can prove the following.

Theorem 1. *There are languages definable in* $\mathsf{LTLf}^{\mathsf{MT}}$ *whose Boolean abstraction cannot be defined in* LTLf.

Proof. Consider the theory of Linear Integer Arithmetic (LIA) together with uninterpreted unary predicates. It is well-known [Wolper, 1983] that LTLf cannot express the language made of words where a proposition p appears in at least all *even* positions. Instead, such a language can be defined as the Boolean abstraction of the language modulo LIA recognized by the following $\mathsf{LTLf}^{\mathsf{MT}}$ formula:

$$x = 0 \wedge \mathsf{G}(\circleddash x = x + 1 \wedge (\exists y(x = y + y) \to \mathsf{P}(x))) \qquad \square$$

Note the essential role of the $\circleddash x$ term here. Without such terms, the satisfaction of first-order formulas in different time steps of a $\mathsf{LTLf}^{\mathsf{MT}}$ model would be completely unrelated, which would allow us to abstract them into proposition symbols. As a matter of fact, we can prove the following.

Theorem 2. *The Boolean abstraction of any language definable in* $\mathsf{LTLf}^{\mathsf{MT}}$ *without* $\bigcirc x$ *nor* $\circleddash x$ *terms can also be defined in* LTLf*, and* vice versa.

Regarding $\bigcirc x$ and $\circleddash x$ terms, it should be noted that the $\mathsf{LTLf}^{\mathsf{MT}}$ syntax only allows these term constructors to be applied to single variables. However, the syntax may be extended to arbitrary nesting of $\bigcirc$ and $\circleddash$ operators with a simple translation maintaining the equisatisfiability. For instance, the formula $x = 1 \wedge \bigcirc x = 1 \wedge \mathsf{G}(\circleddash\circleddash x = \circleddash x + x)$, expressing the Fibonacci sequence, is equisatisfiable to $x = 1 \wedge \bigcirc x = 1 \wedge y = 1 \wedge \mathsf{G}(y = \circleddash x \wedge \circleddash y = y + x)$.

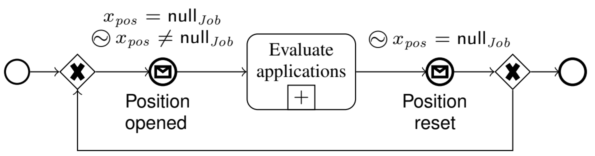

$x_{pos} = \mathsf{null}_{Job}$
$x_{pos} \neq \mathsf{null}_{Job}$
Evaluate applications
$x_{pos} = \mathsf{null}_{Job}$
Position opened
Position reset

Figure 1: Example business process model of a job application

3 Use Cases

Verification of data-aware processes. We take inspiration from verification of data-aware processes to present an interesting class of case studies. Specifically, we rely on the framework by Calvanese *et al.* [2020; 2021], who introduced a general model of transition systems interacting with relational databases, with the distinctive feature that the transitions can query the content of the persistent data storage. As shown there, the theory of *equality and uninterpreted functions* (EUF) provides an algebraic formalization of relational databases with primary and foreign key dependencies: in particular, unary functions map the primary key attribute of a relation schema to another attribute of the same relation, and implicitly represent key dependencies. If EUF is combined with an arithmetical theory such as LRA (e.g., following [Calvanese *et al.*, 2019a; Calvanese *et al.*, 2022]), complex datatype values can be injected into the database and used to model non-deterministic inputs from external users [Calvanese *et al.*, 2019b]. A symbolic transition system Sys operating over databases can be formalized by means of: (i) a sequence of variable symbols $x_1, \dots, x_n$ from $\mathcal{V}$, describing the current configuration; (ii) a purely first-order formula $I(x_1, \dots, x_n)$, describing the initial states; (iii) a weak first-order formula $Tr(x_1, \dots, x_n)$ describing the transitions from the current configuration to the next one. Given an $\mathsf{LTLf}^{\mathsf{MT}}$ formula $\psi(x_1, \dots, x_n)$, called *property*, we are interested in establishing whether $I \wedge \mathsf{G}\,(Tr) \wedge \psi$ is satisfiable modulo the theory constraining the database, i.e., EUF $\cup$ LRA. As an example, imagine a job hiring business process modeling the evaluation of the applications received for a job position: the transition system is intuitively represented, via the BPMN standard language,[1] in Figure 1. The following property states that whenever a job position is opened, eventually it is closed.

$$\mathsf{G}(x_{pos} \neq \mathsf{null}_{Job} \rightarrow \mathsf{F}(x_{status} = \mathsf{PosClosed}))$$

Data-aware planning. $\mathsf{LTLf}^{\mathsf{MT}}$ satisfiability over LRA provides a straightforward approach to *numeric planning problems*, which are similarly undecidable [Helmert, 2002] but nevertheless very relevant in practice. Even more interestingly, we can model *data-aware* planning problems, *i.e.*, *planning problems* [Ghallab *et al.*, 2004] where preconditions of actions can query a relational database. As shown by Cialdea Mayer *et al.* [2007], LTL can encode classical STRIPS-like planning problems. Similarly, data-aware planning problems can be encoded with $\mathsf{LTLf}^{\mathsf{MT}}$ formulas. The encoding of the planning domain results into a data-aware transition system similar to the one above, with the property to verify corresponding to the reachability of the goal.

[1]https://www.omg.org/spec/BPMN/

Rule	$\psi \in \Gamma$	$\Gamma_1(\psi)$	$\Gamma_2(\psi)$
DISJUNCTION	$\alpha \vee \beta$	$\{\alpha\}$	$\{\beta\}$
CONJUNCTION	$\alpha \wedge \beta$	$\{\alpha, \beta\}$	
UNTIL	$\alpha \,\mathsf{U}\, \beta$	$\{\beta\}$	$\{\alpha, \utilde{\mathsf{X}}(\alpha \,\mathsf{U}\, \beta)\}$
RELEASE	$\alpha \,\mathsf{R}\, \beta$	$\{\alpha, \beta\}$	$\{\beta, \widetilde{\mathsf{X}}(\alpha \,\mathsf{R}\, \beta)\}$

Table 1: Expansion rules for the $\mathsf{LTLf}^{\mathsf{MT}}$ tableau

4 One-Pass Tree-Shaped Tableau for $\mathsf{LTLf}^{\mathsf{MT}}$

Here we provide a semi-decision procedure for $\mathsf{LTLf}^{\mathsf{MT}}$ satisfiability based on an adaptation of the one-pass tree-shaped tableau for LTL by Reynolds [2016]. In contrast to classic graph-shaped tableaux, Reynolds' produces a purely tree-shaped structure where a single pass is sufficient to either accept or reject a given branch. Reynolds' tableau proved to be amenable to efficient implementation [Bertello *et al.*, 2016] and parallelization [McCabe-Dansted and Reynolds, 2017], and to direct extension to real-time logics [Geatti *et al.*, 2021a]. A state-of-the-art SAT encoding of Reynolds' tableau has been implemented in the BLACK satisfiability solver [Geatti *et al.*, 2019; Geatti *et al.*, 2021b]. Here, we adapt the tableau system to $\mathsf{LTLf}^{\mathsf{MT}}$, and provide an SMT encoding that we implement in the BLACK satisfiability tool.

The *closure* of a formula ϕ, denoted $\mathcal{C}(\phi)$, is the set of all the subformulas of ϕ with the addition of $\mathsf{X}(\psi_1 \,\mathsf{U}\, \psi_2)$ for any formula $\psi_1 \,\mathsf{U}\, \psi_2 \in \mathcal{C}(\phi)$, and $\widetilde{\mathsf{X}}(\psi_1 \,\mathsf{R}\, \psi_2)$ for any formula $\psi_1 \,\mathsf{R}\, \psi_2 \in \mathcal{C}(\phi)$.

A tableau for an $\mathsf{LTLf}^{\mathsf{MT}}$ formula ϕ is a rooted tree where each node u is labelled by a set of formulas $\Gamma(u) \subseteq \mathcal{C}(\phi)$. The root u_0 is labelled by $\Gamma(u_0) = \{\phi\}$. The tree is built starting from the root, until all branches have been either accepted or rejected. If at least one branch is accepted, ϕ is satisfiable. The tree is built by applying a set of *expansion rules* to the formulas in each node's label. Such rules are listed in Table 1. When a rule is applied to a formula $\psi \in \Gamma(u)$, two children u' and u'' are added to u, where $\Gamma(u') = \Gamma(u) \setminus \{\psi\} \cup \Gamma_1(u)$ and $\Gamma(u'') = \Gamma(u) \setminus \{\psi\} \cup \Gamma_2(u)$, unless $\Gamma_2(u)$ is empty, in which case only a single child is added.

When no expansion rules are applicable to a node, it is called a *poised node*. A poised node contains only *elementary* formulas, *i.e.*, only first-order, *tomorrow* ($\mathsf{X}\phi$) or *weak tomorrow* ($\widetilde{\mathsf{X}}\phi$) formulas. A poised node represents a single state in a tentative model for the formula. So let $\overline{u} = \langle u_0, \dots, u_{n-1} \rangle$ be a branch of the tableau with u_{n-1} a poised node. A temporal step can be made through the STEP rule.

STEP A new child u_n of u_{n-1} is created such that:

$$\Gamma(u_n) = \{\psi \mid \mathsf{X}\psi \in \Gamma(u_{n-1}) \text{ or } \widetilde{\mathsf{X}}\psi \in \Gamma(u_{n-1})\}$$

By alternating the STEP rule with the expansion rules we build each branch of the tree. Then, we need a way to stop the construction when a branch is ready to be accepted or rejected. Two *termination rules* are provided for this purpose. The CONTRADICTION rule *rejects* a branch when it presents some contradiction. In order to define this rule, we need some definitions. Given a branch $\overline{u} = \langle u_0, \dots, u_{n-1} \rangle$, with u_{n-1} a poised node, we define a first-order formula $\Omega(\overline{u})$ that summarizes the first-order formulas made true by the branch $\overline{u}$ over the different time points. This formula is defined over a new alphabet $\Sigma' = \mathcal{P}' \cup \mathcal{C} \cup \mathcal{F} \cup \mathcal{V}' \cup \mathcal{W}$ where $\mathcal{P}' = \mathcal{P} \cup \{\ell^i | i \in \mathbb{N}\}$, and $\mathcal{V}' = \{x^i | x \in \mathcal{V}, i \in \mathbb{N}\}$. Here, ℓ^i are fresh 0-ary EUF predicates, and $\mathcal{V}'$ is a set of *stepped* versions of the variables in $\mathcal{V}$. Given a term t, the *stepped* version of t at time i is the term t^i defined as follows:

1. $c^i = c$
2. $x^i = x$ if $x \in \mathcal{W}$;
3. $x^i = x^i$ if $x \in \mathcal{V}$;
4. $(\bigcirc x)^i = (\circleddash x)^i = x^{i+1}$
5. $(f(t_1, \ldots, t_k))^i = f((t_1)^i, \ldots, (t_k)^i)$

By extension, given a first-order formula ψ, the formula ψ^i is obtained by replacing each term t in ψ with its stepped version t^i. Given a first-order formula ϕ, the formula $L_i(\phi)$ is obtained from ϕ by replacing all *strong* atoms α with $\ell^i \wedge \alpha$, and all *weak* atoms α with $\ell^i \rightarrow \alpha$. Intuitively, if ℓ^i is true, i is not the last step of the model, hence $L_i(\alpha)$ encodes this aspect of the semantics of $\bigcirc x$ and $\circleddash x$ terms.

For the branch $\overline{u}$, let $\overline{\pi} = \langle \pi_0, \ldots, \pi_{m-1} \rangle$ be the sequence of *poised nodes* of $\overline{u}$. Let $F(\pi_i)$ be the set of *first-order* formulas of $\Gamma(\pi_i)$. Then we define the $\Omega(\overline{u})$ formula as follows:

$$\Omega(\overline{u}) = \bigwedge_{i=0}^{m-1} \bigwedge_{\psi \in F(\pi_i)} (L_i(\psi))^i \wedge \bigwedge_{i=0}^{m-2} \ell^i$$

Intuitively, the purpose of $\Omega(\overline{u})$, which is a purely first-order formula over $\mathcal{T} \cup$ EUF, is that of describing the whole tentative model represented by $\overline{u}$ in a single formula. Note that, in $\Omega(\overline{u})$, the value of ℓ^{m-1} is left unconstrained. Hence, if $\Omega(\overline{u})$ is unsatisfiable, a contradiction exists independently from the choice of closing or extending the branch. Now we can define the rule.

CONTRADICTION The branch $\overline{u}$ is rejected if $\Omega(\overline{u})$ is unsatisfiable modulo $\mathcal{T} \cup$ EUF.

Note that the CONTRADICTION rule can be easily checked by giving $\Omega(\overline{u})$ to an SMT solver over $\mathcal{T} \cup$ EUF.

The EMPTY rule, instead, *accepts* suitable branches when there is no reason to further extend the model.

EMPTY If $\Gamma(\pi_{m-1})$ does *not* contain *tomorrow* formulas and $\Omega(\overline{u}) \wedge \neg\ell^{m-1}$ is *satisfiable*, the branch is accepted.

We can prove soundness and completeness of the system.

Theorem 3 (Soundness and completeness)**.** *Given a* LTLf$^{\mathsf{MT}}$ *formula ϕ, the tableau for ϕ has an accepted branch if and only if ϕ is satisfiable.*

If $\mathcal{T} \cup$ QF_EUF is decidable, the breadth-first construction of the tableau tree provides a semi-decision procedure for LTLf$^{\mathsf{MT}}$ satisfiability, as it always finds at least an accepted branch when given a satisfiable formula.

Despite the generality of our definitions, it is clear that our approach to LTLf$^{\mathsf{MT}}$ satisfiability is applicable over decidable theories supported by the underlying SMT solvers. This is the case for most quantifier-free fragments of the supported theories and their combinations, and, in some cases (such as LRA and LIA), also in the quantified case.

Algorithm 1 BLACK's main procedure for LTLf$^{\mathsf{MT}}$

```
1: procedure BLACK($\phi$)
2:     $k \leftarrow 0$
3:     while $True$ do
4:         if $\langle\langle\phi\rangle\rangle_k$ is UNSAT then
5:             return $\phi$ is UNSAT
6:         end if
7:         if $|\phi|_k$ is SAT then
8:             return $\phi$ is SAT
9:         end if
10:        $k \leftarrow k + 1$
11:    end while
12: end procedure
```

The procedure may also terminate on some unsatisfiable formulas, such as those where the unsatisfiability comes from theory contradictions, such as $x = 3 \wedge \mathsf{G}(\exists y(x = y + y))$. It cannot terminate instead on formulas where the unsatisfiability comes from unsatisfiable temporal requests. An example of such formulas is $\mathsf{G}(x > 3) \wedge \mathsf{F}(x < 2)$, whose tableau contains an infinite branch which tries to fulfill the $\mathsf{F}(x < 2)$ eventuality at each step. In Reynolds' tableau for LTL [Reynolds, 2016], closing such branches is the purpose of the PRUNE rule, which however cannot be applied in LTLf$^{\mathsf{MT}}$ (otherwise we would get a decision procedure for an undecidable problem). It is worth noticing that the PRUNE rule can be made to work if we do not allow $\bigcirc x$ and $\circleddash x$ terms. That is, a decision procedure is possible for the fragment of LTLf$^{\mathsf{MT}}$ without $\bigcirc x$ nor $\circleddash x$ terms, which, however, as we have seen, is far less expressive.

5 The SMT Encoding

The SAT encoding of the original tableau for LTL by Reynolds has been implemented in the BLACK satisfiability checking tool as described by Geatti *et al.* [2019; 2021b]. Here, we extend it naturally, namely with an SMT encoding of the tableau system described above.

We encode the tableau for a formula ϕ as a pair of SMT formulas that represent the branches of the tree up to a depth k, for increasing values of k, as shown in Algorithm 1. The formula $\langle\langle\phi\rangle\rangle_k$ used at Line 4 is called the *k-unraveling* of ϕ, and represents the branches of the tree with at most $k + 1$ poised nodes. To define it, we encode the expansion rules of Table 1.

Definition 1 (Stepped Normal Form)**.** *Given an* LTLf$^{\mathsf{MT}}$ *formula ϕ and an $i \geq 0$, its i-th* stepped normal form*, denoted by* $\mathrm{snf}_i(\phi)$*, is defined as follows:*

$$\begin{aligned}
\mathrm{snf}_i(\lambda) &= L_i(\lambda) && \text{where } \lambda \text{ is a first-order formula}\\
\mathrm{snf}_i(\otimes \phi_1) &= \otimes \phi_1 && \text{where } \otimes \in \{\mathsf{X}, \widetilde{\mathsf{X}}\}\\
\mathrm{snf}_i(\phi_1 \otimes \phi_2) &= \mathrm{snf}_i(\phi_1) \otimes \mathrm{snf}_i(\phi_2) && \text{where } \otimes \in \{\wedge, \vee\}\\
\mathrm{snf}_i(\phi_1 \,\mathsf{U}\, \phi_2) &= \mathrm{snf}_i(\phi_2) \vee (\mathrm{snf}_i(\phi_1) \wedge \mathsf{X}(\phi_1 \,\mathsf{U}\, \phi_2))\\
\mathrm{snf}_i(\phi_1 \,\mathsf{R}\, \phi_2) &= \mathrm{snf}_i(\phi_2) \wedge (\mathrm{snf}_i(\phi_1) \vee \widetilde{\mathsf{X}}(\phi_1 \,\mathsf{R}\, \phi_2))
\end{aligned}$$

For a generic formula ψ and for $i > 0$, considering its stepped version ψ^i, we denote as ψ^i_G the result of replacing each *tomorrow* or *weak tomorrow* formula τ with a fresh 0-ary EUF predicate τ^i_G.

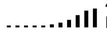
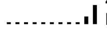
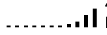
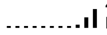
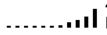
Theory	Formula	Result	Behavior
LIA	$x = 0 \wedge \mathsf{G}(\tilde{\bigcirc} x = x + 1) \wedge \mathsf{F}(x = N)$	SAT	 4min50s N=2590
	$x_0 > 0 \wedge \bigwedge_{i=0}^{N-1} \mathsf{X}^i(\bigcirc x_{i+1} > x_i) \wedge \mathsf{G}(\bigwedge_{i=0}^{N} \tilde{\bigcirc} x_i = x_i) \wedge \mathsf{G}(\sum_{i=0}^{N-1} x_i = \frac{N(N-1)}{2} - 1)$	UNSAT	 4min30s N=57
LRA	$c = 1 \wedge \mathsf{G}(\tilde{\bigcirc} c = 10c) \wedge \mathsf{X}^N(x = c \wedge \mathsf{G}(\tilde{\bigcirc} x = \frac{x}{10}) \wedge \mathsf{F}(x = 1))$	SAT	 4min43s N=234
	$c = 1 \wedge \mathsf{G}(\tilde{\bigcirc} c = 10c) \wedge e = 1 \wedge x = 0$ $\wedge \mathsf{X}^N\big(g = c \wedge \mathsf{G}(\tilde{\bigcirc} e = \frac{e}{2} \wedge \tilde{\bigcirc} x = x + e \wedge 0 \leq x < 2) \wedge \mathsf{F}(x > 2 - \frac{1}{g})\big)$	SAT	 4min46s N=113
EUF+LIA	$n = 0 \wedge c \geq 0 \wedge \mathsf{G}\begin{pmatrix} \tilde{\bigcirc} c = c \wedge \\ \tilde{\bigcirc} n = n + 1 \end{pmatrix} \wedge \mathsf{G}\begin{pmatrix} n > 1 \to f(n) = 2f(n-1) + c \wedge \\ n = 1 \to f(n) = c \end{pmatrix} \wedge \mathsf{X}^N(\widetilde{\mathsf{X}}\bot)$	SAT	 4min28s N=387

Table 2: Test formulas used in the experimental evaluation. N is the scalable parameter.

The k-unraveling of ϕ is recursively defined as follows:

$$\langle\!\langle \phi \rangle\!\rangle_0 = \mathrm{snf}_0(\phi)^0_G$$
$$\langle\!\langle \phi \rangle\!\rangle_{k+1} = \langle\!\langle \phi \rangle\!\rangle_k \wedge \ell^k \wedge \bigwedge_{\mathsf{X}\alpha \in \mathsf{XR}} \Big((\mathsf{X}\alpha)^k_G \leftrightarrow \mathrm{snf}_{k+1}(\alpha)^{k+1}_G \Big)$$
$$\wedge \bigwedge_{\widetilde{\mathsf{X}}\alpha \in \widetilde{\mathsf{X}}\mathsf{R}} \Big((\widetilde{\mathsf{X}}\alpha)^k_G \leftrightarrow \mathrm{snf}_{k+1}(\alpha)^{k+1}_G \Big)$$

where XR and $\widetilde{\mathsf{X}}\mathsf{R}$ are the sets of all the *tomorrow* and *weak tomorrow* formulas, respectively, in $\mathcal{C}(\phi)$.

If $\langle\!\langle \phi \rangle\!\rangle_k$ is unsatisfiable, it means all the branches of the tableau for ϕ are of at most $k + 1$ poised nodes and are all rejected by the CONTRADICTION rule.

The $|\phi|_k$ formula encodes the EMPTY rule:

$$|\phi|_k \equiv \langle\!\langle \phi \rangle\!\rangle_k \wedge \bigwedge_{\psi \in \mathsf{XR}} \neg\psi^k_G \wedge \neg\ell^k$$

If $|\phi|_k$ is satisfiable, the tableau for ϕ has at least a branch accepted by the EMPTY rule.

By relating the tableau branches with the models for $\langle\!\langle \phi \rangle\!\rangle_k$ and $|\phi|_k$, one can easily prove the following, taking inspiration from [Geatti *et al.*, 2021b].

Theorem 4. *Algorithm 1 answers* SAT *if and only if the tableau for ϕ has an accepted branch.*

6 Experiments

We implemented the above encoding in the BLACK satisfiability checking tool.[2] The current implementation supports the full LTLf$^{\mathsf{MT}}$ syntax limited to mono-sorted theories, but in addition to that, it supports past operators, which are handled in the tableau similarly to [Geatti *et al.*, 2021a] and are encoded similarly to [Geatti *et al.*, 2021b].

We tested BLACK on a number of crafted benchmarks intended to stress the underlying algorithm in different ways. We have a total of 990 formulas generated from 5 scalable patterns each parameterized on a value N. The scalable schemata are shown in Table 2. The table shows the theory over which the formulas are intended to be tested, the expected result of the satisfiability checking, and a qualitative plot of running times from $N = 1$ up to the maximum value of N which runs under the 5 minutes timeout. The tests were run on commodity hardware, setting BLACK to use the Z3 backend [de Moura and Bjørner, 2008].

The first LIA schema represents a set of simple counters going from 0 to N. The tableau for these formulas has an accepted branch of exponential size, but BLACK's encoding is not paying a price to that. In contrast, the explicit Boolean encoding of a counter found in BLACK's propositional LTL benchmark set [Geatti *et al.*, 2019] takes orders of magnitude longer to solve in the harder cases. This unsurprising fact shows the advantage of the explicit arithmetic reasoning done by the SMT solver over bit blasting techniques. The second LIA schema is an example of *unsatisfiable* formulas where BLACK nevertheless halts, and stresses the behavior of the solver when nontrivial arithmetic is involved.

The LRA schemata represent behaviors that are only possible in a dense domain. Both build an arbitrary big integer value used to define an arbitrary small fraction to use as the target of the computation. The combined EUF+LIA schema recursively defines a function (inspired by the computational complexity proofs of common sorting algorithms) and forces the model to compute it for the given number of steps.

We can see that these formulas stress the solver but are solved in a manageable amount of time. Globally, this gives evidence of the applicability of our approach to data-aware reasoning.

7 Conclusions

This paper presented LTLf$^{\mathsf{MT}}$, an extension of LTLf where propositional symbols are replaced by first-order formulas over arbitrary theories, *à la* SMT. We discussed how the logic can be useful to approach the model-checking problem for data-aware systems and a form of data-aware planning problems. Although LTLf$^{\mathsf{MT}}$ is easily seen to be undecidable, we provided a semi-decision procedure in the form of a sound and complete tableau system *à la* Reynolds, that we encode into an incremental SMT-based algorithm. We then implemented our approach in BLACK, a state-of-the-art LTL/LTLf satisfiability checker, proving the applicability of the approach with a number of crafted benchmark tests.

Investigating and developing the use cases discussed in Section 3 seems the most natural evolution of this work.

[2]The source code and all the tests and benchmarks are available in BLACK's GitHub repository, merged into version 0.7 of the tool.

Acknowledgements

This work was supported by the projects VERBA, MENS, TOTA and STAGE by the Free University of Bozen-Bolzano.

References

[Artale *et al.*, 2019] Alessandro Artale, Andrea Mazzullo, and Ana Ozaki. Do you need infinite time? In *Proc. of IJCAI 2019*, pages 1516–1522, 2019.

[Barrett *et al.*, 2009] Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Satisfiability*, volume 185. IOS Press, 2009.

[Bertello *et al.*, 2016] Matteo Bertello, Nicola Gigante, Angelo Montanari, and Mark Reynolds. Leviathan: A new LTL satisfiability checking tool based on a one-pass tree-shaped tableau. In *Proc. of IJCAI 2016*, pages 950–956, 2016.

[Calvanese *et al.*, 2019a] Diego Calvanese, Silvio Ghilardi, Alessandro Gianola, Marco Montali, and Andrey Rivkin. Combined covers and Beth definability (extended version). *CoRR*, abs/1911.07774, 2019.

[Calvanese *et al.*, 2019b] Diego Calvanese, Silvio Ghilardi, Alessandro Gianola, Marco Montali, and Andrey Rivkin. Formal modeling and SMT-based parameterized verification of data-aware BPMN. In *Proc. of BPM 2019*, pages 157–175, 2019.

[Calvanese *et al.*, 2020] Diego Calvanese, Silvio Ghilardi, Alessandro Gianola, Marco Montali, and Andrey Rivkin. SMT-based verification of data-aware processes: a model-theoretic approach. *Math. Struct. Comput. Sci.*, 30(3):271–313, 2020.

[Calvanese *et al.*, 2021] Diego Calvanese, Silvio Ghilardi, Alessandro Gianola, Marco Montali, and Andrey Rivkin. Model completeness, uniform interpolants and superposition calculus. *J. Autom. Reason.*, 65(7):941–969, 2021.

[Calvanese *et al.*, 2022] Diego Calvanese, Silvio Ghilardi, Alessandro Gianola, Marco Montali, and Andrey Rivkin. Combination of uniform interpolants via Beth definability. *J. Autom. Reason.*, 2022.

[Cialdea Mayer *et al.*, 2007] Marta Cialdea Mayer, Carla Limongelli, Andrea Orlandini, and Valentina Poggioni. Linear temporal logic as an executable semantics for planning languages. *J. Log. Lang. Inf.*, 16(1):63–89, 2007.

[Cimatti *et al.*, 2020] Alessandro Cimatti, Alberto Griggio, Enrico Magnago, Marco Roveri, and Stefano Tonetta. SMT-based satisfiability of first-order LTL with event freezing functions and metric operators. *Inf. Comput.*, 272:104502, 2020.

[De Giacomo and Vardi, 2013] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *Proc. of IJCAI 2013*, pages 854–860, 2013.

[De Giacomo *et al.*, 2014a] Giuseppe De Giacomo, Riccardo De Masellis, Marco Grasso, Fabrizio Maria Maggi, and Marco Montali. Monitoring business metaconstraints based on LTL and LDL for finite traces. In *Proc. of BPM 2014*, pages 1–17, 2014.

[De Giacomo *et al.*, 2014b] Giuseppe De Giacomo, Riccardo De Masellis, and Marco Montali. Reasoning on LTL on finite traces: Insensitivity to infiniteness. In *Proc. of AAAI 2014*, pages 1027–1033, 2014.

[de Moura and Bjørner, 2008] Leonardo de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In *Proc. of TACAS 2008*, pages 337–340, 2008.

[Geatti *et al.*, 2019] Luca Geatti, Nicola Gigante, and Angelo Montanari. A SAT-based encoding of the one-pass and tree-shaped tableau system for LTL. In *Proc. of TABLEAUX 2019*, pages 3–20, 2019.

[Geatti *et al.*, 2021a] Luca Geatti, Nicola Gigante, Angelo Montanari, and Mark Reynolds. One-pass and tree-shaped tableau systems for TPTL and TPTL_b+Past. *Inf. Comput.*, 278:104599, 2021.

[Geatti *et al.*, 2021b] Luca Geatti, Nicola Gigante, Angelo Montanari, and Gabriele Venturato. Past matters: Supporting LTL+Past in the BLACK satisfiability checker. In *Proc. of TIME 2021*, 2021.

[Geatti *et al.*, 2022] Luca Geatti, Alessandro Gianola, and Nicola Gigante. Linear temporal logic modulo theories over finite traces (extended version). Technical Report https://arxiv.org/abs/2204.13693, arXiv.org, 2022.

[Ghallab *et al.*, 2004] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning - theory and practice*. Elsevier, 2004.

[Helmert, 2002] Malte Helmert. Decidability and undecidability results for planning with numerical state variables. In *Proc. of ICAPS 2002*, pages 44–53, 2002.

[Hodkinson *et al.*, 2000] Ian Hodkinson, Frank Wolter, and Michael Zakharyaschev. Decidable fragments of first-order temporal logics. *Ann. Pure Appl. Log.*, 106(1-3):85–134, 2000.

[Kontchakov *et al.*, 2004] Roman Kontchakov, Carsten Lutz, Frank Wolter, and Michael Zakharyaschev. Temporalising tableaux. *Stud. Log.*, 76(1):91–134, 2004.

[Li *et al.*, 2017] Yuliang Li, Alin Deutsch, and Victor Vianu. VERIFAS: A practical verifier for artifact systems. *Proc. VLDB Endow.*, 11(3):283–296, 2017.

[McCabe-Dansted and Reynolds, 2017] John Christopher McCabe-Dansted and Mark Reynolds. A parallel linear temporal logic tableau. In *Proc. of GandALF 2017*, pages 166–179, 2017.

[Pnueli, 1977] Amir Pnueli. The temporal logic of programs. In *Proc. of FOCS 1977*, pages 46–57, 1977.

[Reynolds, 2016] Mark Reynolds. A new rule for LTL tableaux. In *Proc. of GandALF 2016*, pages 287–301, 2016.

[Wolper, 1983] Pierre Wolper. Temporal logic can be more expressive. *Information and control*, 56(1-2):72–99, 1983.