Temporal Logics Over Finite Traces with Uncertainty

Fabrizio M. Maggi[*]
University of Tartu
f.m.maggi@ut.ee

Marco Montali[†]
Free University of Bozen-Bolzano
montali@inf.unibz.it

Rafael Peñaloza
University of Milano-Bicocca
rafael.penaloza@unimib.it

Abstract

Temporal logics over finite traces have recently seen wide application in a number of areas, from business process modelling, monitoring, and mining to planning and decision making. However, real-life dynamic systems contain a degree of uncertainty which cannot be handled with classical logics. We thus propose a new probabilistic temporal logic over finite traces using superposition semantics, where all possible evolutions are possible, until observed. We study the properties of the logic and provide automata-based mechanisms for deriving probabilistic inferences from its formulas. We then study a fragment of the logic with better computational properties. Notably, formulas in this fragment can be discovered from event log data using off-the-shelf existing declarative process discovery techniques.

Introduction

Linear temporal logic (LTL) is one of the most important formalisms to declaratively specify and reason about the evolution of systems and processes (Baier and Katoen 2008). Traditionally, LTL adopts a linear, infinite model of time where formulas are interpreted over infinite traces. In recent years, increasing attention has been given to a different version of the logic, *LTL over finite traces* or LTL_f (De Giacomo and Vardi 2013), which adopts instead a finite-trace semantics. From the modelling point of view, LTL_f matches settings where each execution of the system is eventually expected to end (even though there is no bound on the number of steps required to reach the termination point). From the reasoning point of view, the automata-theoretic characterisation of LTL_f relies on classical finite-state automata (De Giacomo and Vardi 2013; De Giacomo, De Masellis, and Montali 2014), which are easier to manipulate and pave the way for the development of robust and efficient reasoning techniques. In fact, LTL_f and extensions thereof have been widely employed in a number of application domains relevant for AI: from declarative business process modelling (Pesic, Schonenberg, and van der Aalst 2007; Montali 2010), monitoring (Maggi et al. 2011; De Giacomo et al. 2014), and mining (Maggi, Chandra Bose, and van der Aalst 2012; De Giacomo et al. 2017), to planning (Gerevini et al. 2009; De Giacomo and Rubin 2018) and decision making (Brafman, De Giacomo, and Patrizi 2018).

When considering real-world processes, uncertainty (which is inexpressible in classical logics) is unavoidable. For example, some pieces may be defective, external events may delay a service, and the loan may lead to an outcome depending on various implicit factors. Handling these scenarios requires a logical formalism capable of expressing uncertainty. Surprisingly, to the best of our knowledge no probabilistic extension of LTL_f has been considered so far. Although several probabilistic variants of infinite-time temporal logics exist (Ognjanovic 2006; Morão 2011; Konur 2010; Kovtunova and Peñaloza 2018; Woltzenlogel Paleo 2016), the complex interaction of probabilities and time usually requires syntactic or semantic restrictions in the logic, and does not directly carry over the finite-trace setting.

To overcome both challenges at once, we propose a new probabilistic extension of LTL_f, called PLTL_f, that essentially predicates over the possible evolutions of a trace. The main novelty of PLTL_f lies in its *superposition* semantics, where every evolution is possible (with different probabilities) until it is observed. This semantics accommodates a seamless interaction of probabilities and time that was not possible in previous formalisms, and elegantly fits over finite traces. PLTL_f is a direct generalisation of LTL_f: PLTL_f formulas without probabilistic constructors are in fact LTL_f formulas. PLTL_f adequately describes probabilistic temporal or dynamic properties of process executions; e.g., it can express that a shipped package will eventually reach its destination with probability 0.95, or that a machine will fail in the next 100 timepoints with probability below 0.001.

Our second main contribution is an investigation of the logical and computational properties of PLTL_f, introducing automata-based algorithms for deciding satisfiability of PLTL_f formulas and for computing the most likely executions of a system described in this logic. These core reasoning services provide the basis for sophisticated, domain-specific tasks such as a probabilistic version of conformance checking (Carmona et al. 2018) and (prefix) monitoring

[*]Supported by the Estonian Research Council Grant IUT20-55.
[†]Partially supported by the UNIBZ CRC project REKAP.

(Maggi et al. 2011). Unsurprisingly, due to the intertwined connection of temporal and probabilistic constructors, handling PLTL$_f$ formulas becomes EXPTIME-hard. This leads us to our third contribution: a study of a fragment of PLTL$_f$, called PLTL$_f^0$, where the complexity of reasoning falls to PSPACE in the length of the formula, matching the classical LTL$_f$ case. Notably, formulas in this fragment can be discovered from event log data using off-the-shelf existing declarative process discovery techniques (Maggi, Chandra Bose, and van der Aalst 2012).

Full proofs and an example covering the main notions are available at (Maggi, Montali, and Peñaloza 2019).

Preliminaries

We briefly introduce tree and weighted string automata, assuming basic formal language knowledge. For more details, see (Comon et al. 2007; Droste, Kuich, and Vogler 2009).

Tree Automata. A *tree* is a set of words of natural numbers $T \subseteq \mathbb{N}^*$, which is closed under prefixes and preceding siblings; i.e., if $wi \in T$, then $w \in T$, and $wj \in T$ for all $1 \leq j \leq i$. A tree is *finite* if its cardinality is finite. Each finite tree T has a maximum number $k \in \mathbb{N}$ (its *width*) s.t. $wk \in T$ for some $w \in \mathbb{N}^*$. The empty word ε is the *root*, and a *leaf* is a node $w \in T$ s.t. $w1 \notin T$. A *labelling* of T on a set Σ is a mapping $T \to \Sigma$. A tree with a labelling is a *labelled tree*. A *branch* of the tree T is a sequence $w_1, \ldots, w_n$ of nodes of T such that $w_1 = \varepsilon$, w_n is a leaf node, and for every $i, 1 \leq i < n$, $w_{i+1} = w_i m$ for $m \in \mathbb{N}$. If T is labelled, we call branch also the sequence of labels of a branch.

A k-ary *tree automaton* is a tuple $\mathcal{A} = (\mathcal{Q}, \Delta, I, F)$ where $\mathcal{Q}$ is a finite set of *states*, $I, F \subseteq \mathcal{Q}$ are the *initial* and *final* states, respectively, and $\Delta \subseteq \mathcal{Q} \times \bigcup_{i \leq k} \mathcal{Q}^k$ is the *transition relation*. A *run* of $\mathcal{A}$ on the tree T is a labelling $\rho : T \to \mathcal{Q}$ s.t. $\rho(\varepsilon) \in I$ and for every $w \in T$, if $wn \in T$ but $w(n{+}1) \notin T$, then $(\rho(w), \rho(w1), \ldots, \rho(wn)) \in \Delta$. It is *successful* if for every leaf node $w \in T$, $\rho(w) \in F$. The *language accepted* by $\mathcal{A}$ is the set $\mathcal{L}(\mathcal{A})$ of finite trees for which there is a successful run of $\mathcal{A}$. The *emptiness problem* asks whether $\mathcal{L}(\mathcal{A}) = \emptyset$.

The emptiness problem of k-ary tree automata is decidable in time $\mathcal{O}(|\mathcal{Q}|^{k+2})$ (Vardi and Wolper 1986) by computing *good states*; i.e., those that appear in a successful run. States $q \in F$ without transitions are good. The set of good states is iteratively extended, adding any state that has a transition leading to only good states. This iteration reaches a fixpoint after checking the transitions of each state at most $|\mathcal{Q}|$ times. As there are at most $|\mathcal{Q}|^{k+1}$ transitions, the set of good states is computable in time $\mathcal{O}(|\mathcal{Q}|^{k+2})$. $\mathcal{A}$ is not empty iff at least one initial state is good. The *reduced automaton* $\ddot{\mathcal{A}}$ of $\mathcal{A}$ is obtained by removing all bad (i.e., non-good) states from $\mathcal{A}$. $\mathcal{A}$ and $\ddot{\mathcal{A}}$ accept the same language, and $\mathcal{L}(\ddot{\mathcal{A}}) \neq \emptyset$ iff $\ddot{\mathcal{A}}$ contains at least one initial state.

An alternative emptiness test constructs a successful run top-down. It guesses an initial state to label the root node, and iteratively guesses transitions for every node not labelled with a final state. Through a depth-first construction, the algorithm preserves in memory only one branch at a time, together with the information of the chosen transitions. Since every branch can be restricted to depth $|\mathcal{Q}|$, the process requires $\mathcal{O}(|\mathcal{Q}|{\cdot}k)$ space (Baader, Hladik, and Peñaloza 2008).

Weighted Automata. Consider the *probabilistic semiring* $A = ([0,1], \max, \times)$ with the usual $\max$ and product on $[0,1]$. A *weighted automaton* is a tuple $\mathcal{A} = (\mathcal{Q}, \mathsf{in}, \mathsf{wt}, F)$ where $\mathcal{Q}$ is a finite set of *states*, $F \subseteq \mathcal{Q}$ are the *final states*, $\mathsf{in} : \mathcal{Q} \to [0,1]$ is the *initialization function* and $\mathsf{wt} : \mathcal{Q} \times \mathcal{Q} \to [0,1]$ is the *weight function*. A *run* of $\mathcal{A}$ is a finite sequence $\rho = q_0, q_1, \ldots, q_n$ with $q_n \in F$; $\mathsf{run}(\mathcal{A})$ is the set of all runs of $\mathcal{A}$. The weight of $\rho = q_0. \ldots, q_n \in \mathsf{run}(\mathcal{A})$ is $\mathsf{wt}(\rho) := \prod_{i=0}^{n-1} \mathsf{wt}(q_i, q_{i+1})$. The *behaviour* of $\mathcal{A}$ is $\|\mathcal{A}\| := \max_{\rho \in \mathsf{run}(\mathcal{A})} \mathsf{in}(q_0) \cdot \mathsf{wt}(\rho)$. To compute the behaviour of the weighted automaton $\mathcal{A}$, we adapt the emptiness test for unweighted automata to consider the weights of the transitions computing a function $\mathsf{w} : \mathcal{Q} \to [0,1]$, where $\mathsf{w}(q)$ is the maximum weight of all runs starting in q. Initialize $\mathsf{w}_0(q) = 1$ if $q \in F$ and $\mathsf{w}_0(q) = 0$ if $q \notin F$. Iteratively compute $\mathsf{w}_{i+1}(q) := \max_{q' \in \mathcal{Q}} \mathsf{wt}(q, q')\mathsf{w}_i(q)$. After polynomially many iterations, we reach a fixpoint where $\mathsf{w}_i \equiv \mathsf{w}_{i+1}$. If $\mathsf{w} := \mathsf{w}_i$, then $\|\mathcal{A}\| = \max_{q \in \mathcal{Q}} \mathsf{in}(q)\mathsf{w}(q)$.

The Probabilistic Temporal Logic PLTL$_f$

PLTL$_f$ extends the linear temporal logic on finite traces LTL$_f$ (De Giacomo and Vardi 2013), with a probabilistic constructor expressing uncertainty about the evolution of traces. The only syntactic difference between LTL$_f$ and PLTL$_f$ is this new constructor. Formally, PLTL$_f$ formulas are built by the following syntactic rule where a is a propositional variable, $p \in [0,1]$, and $\bowtie \in \{\leq, \geq, <, >\}$:

$$\varphi ::= a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi U \varphi \mid \circledcirc_{\bowtie p}\varphi.$$

Intuitively, $\circledcirc_{\bowtie p}\varphi$ means that, at the next point in time, φ holds with probability $\bowtie p$. To formalise this, we use tree-shaped interpretations providing a class of alternatives branching into the future in a new *superposition semantics*. A *PLTL$_f$ interpretation* is a triple $I = (T, \cdot^I, P)$, where T is a finite tree, $\cdot^I$ is a labelling of T on the set of propositional valuations,[1] and $P : T \setminus \{\varepsilon\} \to [0,1]$ is s.t. for all $w \in T$, $\sum_{wi \in T} P(wi) = 1$. Satisfiability of a formula in a tree node is defined inductively, extending the LTL$_f$ semantics. For an interpretation $I = (T, \cdot^I, P)$ and $w \in T$:

- $I, w \models a$ iff $a \in w^I$
- $I, w \models \neg\varphi$ iff $I, w \not\models \varphi$
- $I, w \models \varphi \wedge \psi$ iff $I, w \models \varphi$ and $I, w \models \psi$
- $I, w \models \bigcirc\varphi$ iff w is not a leaf node and for all $i \in \mathbb{N}$, if $wi \in T$ then $I, wi \models \varphi$
- $I, w \models \varphi U \psi$ iff either (i) $I, w \models \psi$ or (ii) $I, w \models \varphi$ and for all $i \in \mathbb{N}$, if $wi \in T$ then $I, wi \models \varphi U \psi$
- $I, w \models \circledcirc_{\bowtie p}\varphi$ iff $\sum_{wi \in T; I, wi \models \varphi} P(wi) \bowtie p$.

I is a *model* of ϕ if $I, \varepsilon \models \phi$. ϕ is *satisfiable* if it has a model.

Example 1. Figure 1 shows three models of the formula $\phi_0 := \circledcirc_{\leq 0.5} a \wedge \circledcirc_{\geq 0.6} \bigcirc b$. In (a), a and $\bigcirc b$ are observed at the next timepoint with probability 0 and 1, respectively; (c) depicts the other extreme, where a and $\bigcirc b$ are observed

[1]As usual, we describe a propositional valuation by the set of variables it makes true.

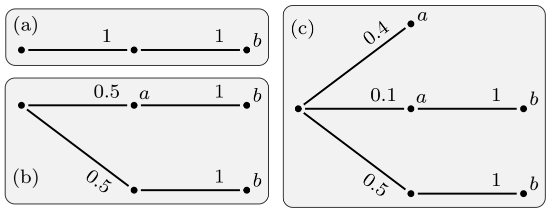

Figure 1: Three models of the formula ϕ_0 from Example 1.

with probability 0.5 and 0.6, respectively. Many intermediate models exist. The formula $\phi_1 := \circledcirc_{\geq 0.5} a \wedge \circledcirc_{\geq 0.6} \neg a$ is unsatisfiable: no model allows a and $\neg a$ to hold with probability 0.5 and 0.6, respectively.

As usual, the main reasoning task for a PLTL$_f$ formula φ is checking satisfiability. This problem becomes harder than in LTL$_f$ by the need to verify the compatibility of the potential future steps w.r.t. their probabilities. Thus, we build a tree automaton accepting a class of models of ϕ. For satisfiability, we can ignore the specific probabilities used, as long as they are compatible. Thus an unweighted automaton suffices. Building this automaton requires some definitions.

For a PLTL$_f$ formula ϕ, $\mathsf{csub}(\phi)$ is the smallest set of PLTL$_f$ formulas containing all subformulas of ϕ, is closed under negation (modulo double negations), and s.t. $\psi_1 U \psi_2 \in \mathsf{csub}(\phi)$ implies $\bigcirc \psi_1 U \psi_2 \in \mathsf{csub}(\phi)$. An *atom* is a subset $\mathbf{a} \subseteq \mathsf{csub}(\phi)$ s.t. (i) for every $\psi \in \mathsf{csub}(\phi)$, $\{\psi, \neg\psi\} \cap \mathbf{a} \neq \emptyset$ and $\{\psi, \neg\psi\} \not\subseteq \mathbf{a}$; (ii) for every formula $\psi_1 \wedge \psi_2 \in \mathsf{csub}(\phi)$, $\psi_1 \wedge \psi_2 \in \mathbf{a}$ iff $\{\psi_1, \psi_2\} \subseteq \mathbf{a}$; and (iii) for all $\psi_1 U \psi_2 \in \mathsf{csub}(\phi)$, $\psi_1 U \psi_2 \in \mathbf{a}$ iff either $\psi_2 \in \mathbf{a}$ or $\bigcirc \psi_1 U \psi_2 \in \mathbf{a}$. Atoms are maximally consistent subsets of $\mathsf{csub}(\phi)$ that also verify the satisfiability of the until operator. The set of all atoms is denoted by $\mathsf{At}(\phi)$. For brevity, we equate $\neg \circledcirc_{\bowtie p} \equiv \circledcirc_{\bowtie^- p}$, where $\bowtie^-$ is the inverse relation of $\bowtie$ and assume that probabilistic formulas are never negated in csub; e.g., $\neg \circledcirc_{\geq 0.5} a$ is replaced by $\circledcirc_{<0.5} a$.

Atoms define the states of the automaton. To define the transitions, we identify the combinations of probabilistic subformulas that can appear together under the uncertainty constraints; e.g., if an atom contains $\circledcirc_{\leq 0.3} \psi_1$ and $\circledcirc_{\leq 0.4} \psi_2$, then transitions must contain a successor with $\neg\psi_1$ and $\neg\psi_2$. Let $\mathcal{P}(\mathbf{a}) = \{\circledcirc_{\bowtie p} \psi \in \mathbf{a}\}$ be the set of all probabilistic formulas in the atom $\mathbf{a}$. For every subset $S \subseteq 2^{\mathcal{P}(\mathbf{a})}$ define the system of inequalities

$$\mathfrak{I}(S) := \{ \sum_{\circledcirc_{\bowtie p_i} \psi_i \in Q, Q \in S} x_Q \bowtie p_i \mid \circledcirc_{\bowtie p_i} \psi_i \in \mathcal{P}(a) \} \cup$$
$$\{x_Q \geq 0 \mid Q \in S\} \cup \{ \sum_{Q \in S} x_Q = 1 \}.$$

$\mathcal{S}(\mathbf{a})$ is the set of all $S \subseteq 2^{\mathcal{P}(\mathbf{a})}$ where $\mathfrak{I}(S)$ has a solution.

Example 2. One atom of the formula ϕ_0 from Example 1 is $\mathbf{a}_0 = \{\phi_0, \circledcirc_{\leq 0.5} a, \circledcirc_{\geq 0.6} \bigcirc b, \neg \bigcirc b, \neg a, \neg b\}$; for which $\mathcal{P}(\mathbf{a}_0) = \{\circledcirc_{\leq 0.5} a, \circledcirc_{\geq 0.6} \bigcirc b\}$. For brevity, call the elements of $\mathcal{P}(\mathbf{a}_0)$ 1 and 2, respectively. The system of inequalities for $S_0 = \{\{1\}, \{2\}, \{1,2\}\}$

$$x_{\{1\}} + x_{\{1,2\}} \leq 0.5$$
$$x_{\{2\}} + x_{\{1,2\}} \geq 0.6$$
$$x_Q \geq 0 \qquad Q \in S_0$$
$$x_{\{1\}} + x_{\{2\}} + x_{\{1,2\}} = 1,$$

has a solution; e.g., $x_{\{1\}} = 0.4$, $x_{\{2\}} = 0.5$, $x_{\{1,2\}} = 0.1$. As shown later, this means that from the atom $\mathbf{a}_0$, it is possible to branch in three scenarios to satisfy the probabilities (Figure 1 (c)). For $S_1 = \{\emptyset, \{1\}, \{1,2\}\}$, the system $\mathfrak{I}(S_1)$

$$x_{\{1\}} + x_{\{1,2\}} \leq 0.5, x_{\{1,2\}} \geq 0.6, x_\emptyset + x_{\{1\}} + x_{\{1,2\}} = 1$$

has no non-negative solution: $x_{\{1,2\}}$ needs to be ≤ 0.5 and ≥ 0.6 simultaneously. Hence $S_0 \in \mathcal{S}(\mathbf{a}_0)$ but $S_1 \notin \mathcal{S}(\mathbf{a}_0)$. The latter means that to satisfy $\mathbf{a}_0$, one must find a transition where the formula $\bigcirc b$ is satisfied, but a is not.

If $\mathcal{P}(\mathbf{a}) = \emptyset$, then $2^{\mathcal{P}(\mathbf{a})} = \{\emptyset\}$, and $\mathfrak{I}(\{\emptyset\}) = \{1 = x_\emptyset\}$, which has a trivial solution; i.e., if an atom $\mathbf{a}$ contains no probabilistic subformulas, $\mathcal{S}(\mathbf{a})$ contains only one element, and the construction reduces to classical LTL$_f$. Each element in $\mathcal{S}(\mathbf{a})$ defines a set of tuples of atoms yielding the transition relation of the automaton. Assume w.l.o.g. that the elements of each $S \in \mathcal{S}(\mathbf{a})$ are ordered as $Q_1, \ldots, Q_{|S|}$. $T_S(\mathbf{a})$ is the set of $|S|$-tuples of atoms $(\mathbf{a}_1, \ldots, \mathbf{a}_{|S|})$ s.t. for all $\bigcirc\psi, \circledcirc_{\bowtie p}\psi \in \mathsf{csub}(\phi)$: (i) $\bigcirc\psi \in \mathbf{a}$ iff $\psi \in \mathbf{a}_i$ for all i, and (ii) for every $i, 1 \leq i \leq |S|$, $\circledcirc_{\bowtie p}\psi \in Q_i$ iff $\psi \in \mathbf{a}_i$.

Example 3. From Example 2, $S_0 \in \mathcal{S}(\mathbf{a})$ defines 3-tuples of atoms s.t. the first two elements contain one of the probabilistic subformulas each, and the last contains both probabilistic subformulas. Hence, the tuple $(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ formed by the following atoms belongs to $T_{S_0}(\mathbf{a})$:[2]

$$\mathbf{a}_1 = \{\neg\phi_0, \circledcirc_{>0.5} a, \circledcirc_{<0.6} \bigcirc b, \neg \bigcirc b, a, \neg b\}$$
$$\mathbf{a}_2 = \{\neg\phi_0, \circledcirc_{>0.5} a, \circledcirc_{<0.6} \bigcirc b, \bigcirc b, \neg a, \neg b\}$$
$$\mathbf{a}_3 = \{\neg\phi_0, \circledcirc_{>0.5} a, \circledcirc_{<0.6} \bigcirc b, \bigcirc b, a, \neg b\}$$

We define an automaton which can decide satisfiability of PLTL$_f$ formulas.

Definition 4. The tree automaton $\mathcal{A}_\phi = (\mathcal{Q}, \Delta, I, F)$ is given by $\mathcal{Q} = \mathsf{At}(\phi)$, $\Delta = \{\{\mathbf{a}\} \times \bigcup_{S \in \mathcal{S}(\mathbf{a})} T_S(\mathbf{a}) \mid \mathbf{a} \in \mathcal{Q}\}$, $I = \{\mathbf{a} \in \mathcal{Q} \mid \varphi \in \mathbf{a}\}$, and F the set of all atoms not containing formulas of the form $\bigcirc\psi$, $\circledcirc_{>p}\psi$, or $\circledcirc_{\geq p'}\psi, p' > 0$.

$\mathcal{A}_\phi$ naturally generalises the automata-based approach for satisfiability of LTL$_f$ formulas: if ϕ has no probabilistic constructor (i.e., it is an LTL$_f$ formula), $\mathcal{A}_\phi$ is the standard automaton for this setting (De Giacomo, De Masellis, and Montali 2014). $\mathcal{A}_\phi$ accepts a class of well-structured quasi-models of the formula ϕ, merging redundant branches. The only missing element to have a model are the probabilistic values attached to each successor of a node. These are found solving the system of inequalities built from each transition.

Theorem 5. *The formula ϕ is satisfiable iff $\mathcal{L}(\mathcal{A}_\phi) \neq \emptyset$.*

Automata emptiness is decidable in time $\mathcal{O}(|\mathcal{Q}|^{k+2})$, where k is the rank of the automaton. In this case, the rank of $\mathcal{A}_\phi$ depends on the size of the formula ϕ; specifically, on the number of probabilistic subformulas that it contains: if

[2]Recall that we equate $\neg \circledcirc_{\leq 0.5} \phi \equiv \circledcirc_{>0.5} \phi$.

$\mathsf{csub}(\phi)$ has n probabilistic subformulas, the rank of $\mathcal{A}_\phi$ is bounded by 2^n; i.e., emptiness of $\mathcal{A}_\phi$ runs in time $\mathcal{O}(|\mathcal{Q}|^{2^n})$. There is also a non-deterministic algorithm that uses space $\mathcal{O}(|\mathcal{Q}| \cdot 2^n)$. As the number of states is bounded exponentially on the length of ϕ, Savitch's theorem (Savitch 1970) yields the following result.

Theorem 6. *PLTL$_f$ satisfiability is decidable in exponential space in the number of probabilistic formulas, but* only *exponential time in the size of the formula.*

If the total number of probabilistic formulas is bounded by some constant, or if we parameterise the problem over the number of probabilistic subformulas (Downey and Fellows 2012), then satisfiability of PLTL$_f$ formulas is in EXPTIME. Conversely, satisfiability is EXPTIME-hard on the length of the input formula ϕ. The proof of this fact is based on a reduction from the *intersection non-emptiness* problem for deterministic tree automata (Comon et al. 2007; Seidl 1994).

Theorem 7. *PLTL$_f$ satisfiability is* EXPTIME*-hard.*

Probabilistic Entailment

We have focused in a decision problem considering only the existence of a model of the PLTL$_f$ formula ϕ, disregarding the probabilistic information included in ϕ. We now consider reasoning problems dealing with the likelihood of different traces. A basic probabilistic reasoning problem on PLTL$_f$ is computing the *most likely* trace, along with its probability. To handle the multiplicity of models, we use an optimistic approach, which selects the model maximising the likelihood of observing a given trace. One could have chosen a *pessimistic* approach minimising the likelihood instead. Such a case can be handled analogously by changing all relevant maxima for minima in the following.

Definition 8. A *trace* is a finite sequence of propositional valuations. The interpretation $I = (T, \cdot^I, P)$ *contains* the trace t if there is a branch b of T such that $\mathsf{b}^I = t$. The *probability* of t in I is $P_I(t) = \prod_{w \in \mathsf{b}} P(w)$. The *probability* of t w.r.t. the PLTL$_f$ formula ϕ is $P_\phi(t) = \max_{I \models \phi} P_I(t)$.

Example 9. Using the formula ϕ_0 from Example 1, let I_0 and I_1 be the models (b) and (c) from Figure 1, respectively. I_1 contains the trace $(\emptyset, \{a\})$, but I_0 does not. Both models contain the trace $t = (\emptyset, \{a\}, \{b\})$; $P_{I_0}(t) = 0.5$; and $P_{I_1}(t) = 0.1$. $P_{\phi_0}(t) = 0.5$ as witnessed by the model I_0.

We want to find the traces with a maximal probability. Formally, t is a *most likely trace* (mlt) w.r.t. ϕ iff for every trace t', $P_\phi(t') \leq P_\phi(t)$. In our running example, a most likely trace is $(\emptyset, \emptyset, \{b\})$, which has probability 1 (Figure 1 (a)); however, mlts are not necessarily unique; indeed, $(\emptyset, \emptyset, \{a, b\})$ and $(\emptyset, \emptyset, \{b\}, \emptyset)$ are also mlts w.r.t. ϕ_0. To find the mlts w.r.t. ϕ, we transform the tree automaton $\mathcal{A}_\phi$ into a weighted string automaton $\mathcal{B}_\phi$ which keeps track of the most likely transitions available from a given state of $\mathcal{A}_\phi$. For brevity, we do not distinguish between the valuation forming a model, and the atom (containing the valuation) of the run of the automaton. Using the probabilistic semiring, the behaviour of $\mathcal{B}_\phi$ yields the probability of the mlts. We later show how to use this information to extract the actual traces.

Recall that the reduced automaton $\dddot{\mathcal{A}}_\phi$ of the emptiness test excludes the bad states from $\mathcal{A}_\phi$ and accepts the same language as $\mathcal{A}_\phi$, but from every state in $\dddot{\mathcal{A}}_\phi$ one can build a successful run. Deleting bad states also removes all transitions (produced by the different combinations of the probabilistic subformulas that appear in an atom) which cannot be used due to semantic incompatibilities. Let $(\mathbf{a}, \mathbf{a}_1, \ldots, \mathbf{a}_n) \in \dddot{\Delta}$; i.e., a transition from $\dddot{\mathcal{A}}_\phi$. There exists an $S \in \mathcal{S}(\mathbf{a})$ such that $(\mathbf{a}_1, \ldots, \mathbf{a}_n) \in T_S$. For each $Q \in S$, we solve the optimisation problem

$$\text{maximize } x_Q \qquad\qquad \text{subject to } \mathfrak{I}(S).$$

Intuitively, we compute the largest probability that a branch satisfying the probabilistic constraints in Q can obtain, given the other branches defined by S. Call the result of this optimisation problem $\mathsf{m}_{S,\mathbf{a}}(Q)$. As each optimisation problem is solved independently, the maxima may not add 1; e.g., for $\mathbf{a}_0, S_0$ from Example 2, $\mathsf{m}_{S_0,\mathbf{a}_0}(\{2\}) = 1$, $\mathsf{m}_{S_0,\mathbf{a}_0}(\{1,2\}) = 0.5$ and $\mathsf{m}_{S_0,\mathbf{a}_0}(\emptyset) = \mathsf{m}_{S_0,\mathbf{a}_0}(\{1\}) = 0.4$. This is intended; we try to identify the largest probability that can be assigned to a path in a model; i.e., a trace.

For an atom $\mathbf{a}$ and a set $Q \subseteq \mathcal{P}(\mathbf{a})$, let now

$$\mathsf{m}_{\mathbf{a}}(Q) := \max_{S \in \mathcal{S}(\mathbf{a}), Q \in S} \mathsf{m}_{S,\mathbf{a}}(Q).$$

We obtain the automaton $\mathcal{B}_\phi$ by *flattening* $\dddot{\mathcal{A}}_\phi$ into a string automaton, and weighting every transition according to m.

Definition 10. $\mathcal{B}_\phi = (\dddot{\mathcal{Q}}, \mathsf{in}, \mathsf{wt}, \dddot{F})$ is the weighted automaton where $\dddot{\mathcal{Q}}$ and $\dddot{F}$ are obtained from $\dddot{\mathcal{A}}_\phi$, $\mathsf{in}(\mathbf{a}) = 1$ iff $\mathbf{a} \in \dddot{I}$ (0 o.w.), and $\mathsf{wt}(\mathbf{a}, \mathbf{a}') = \mathsf{m}_{\mathbf{a}}(Q)$, where $Q \in \mathbf{a}'$.

Importantly, $\mathcal{B}_\phi$ is constructed from $\dddot{\mathcal{A}}_\phi$ and not from $\mathcal{A}_\phi$. This ensures that branches defined by unsatisfiable constraints are ignored. For example, if $\{\circledcirc_{\leq p} a, \circledcirc_{\leq q} \neg a\} \subseteq \mathbf{a}$, with $p + q < 1$, $\mathcal{S}(\mathbf{a}) \neq \emptyset$, and $\mathsf{m}_{\mathbf{a}}(\{\circledcirc_{\leq p} a\}) = p$. However, if $\mathbf{a}$ is not a final state, but a bad state: $\mathbf{a}$ has no transition. Constructing $\mathcal{B}_\phi$ from $\mathcal{A}_\phi$, $\mathbf{a}$ would have a transition to an atom $\mathbf{a}'$ containing a (with weight p), which is incorrect.

Theorem 11. *Let $\mathcal{B}_\phi$ be the weighted automaton constructed from ϕ by Definition 10. The probability of the mlt w.r.t. ϕ is $\|\mathcal{B}_\phi\|$.*

The behaviour of $\mathcal{B}_\phi$ is computable in polynomial time on the number of states, i.e., exponential on the length of the formula, but is not affected by the number of probabilistic subformulas appearing in ϕ. Yet, to build $\mathcal{B}_\phi$, we need first to construct and manipulate $\mathcal{A}_\phi$, which may be doubly-exponential on the number of probabilistic subformulas. Indeed, there is a trace with positive probability iff ϕ is satisfiable. Thus, deciding whether the probability of the most likely trace is higher than some bound has the same complexity as satisfiability.

Corollary 12. *The probability of the mlts is computable in exponential space in the number of probabilistic formulas, but exponential time in the size of the formula. Deciding if it is greater than 0 is* EXPTIME*-hard.*

To find the mlts (and not just their probability), we adapt the computation process for the behaviour of $\mathcal{B}_\phi$. At each iteration of the computation, associate each state with the maximum probability that can be derived from a trace starting from it. Together with this number, we also store the successor states yielding that maximum probability, getting an automaton that accepts all the most likely traces.

Definition 13. Given a PLTL$_f$ formula ϕ, its weighted automaton $\mathcal{B}_\phi = (\mathcal{Q}, \mathsf{in}, \mathsf{wt}, F)$, and a state $\mathbf{a} \in \mathcal{Q}$, let $\mathsf{w}(\mathbf{a})$ be obtained through the computation of the behaviour of $\mathcal{B}_\phi$. The unweighted automaton $\overline{\mathcal{B}_\phi} = (\mathcal{Q}, I, \Delta, F)$ is given by

$$I = \{\mathbf{a} \in \mathcal{Q} \mid \mathsf{in}(\mathbf{a}) \cdot \mathsf{w}(\mathbf{a}) = \|\mathcal{B}_\phi\|\},$$
$$\Delta = \{(\mathbf{a}, \mathbf{a}') \in \mathcal{Q} \times \mathcal{Q} \mid \mathsf{wt}(\mathbf{a}, \mathbf{a}') \cdot \mathsf{w}(\mathbf{a}') = \mathsf{w}(\mathbf{a})\}.$$

Theorem 14. *$\overline{\mathcal{B}_\phi}$ accepts the most likely traces.*

While it is important to understand the mlts, it is often more useful to compute the likelihood of observing a specific trace or an element from a set of traces; e.g., to verify that an unwanted outcome is unlikely. This problem can be reduced to that of computing the most likely traces, as long as the set of desired traces is a recognisable language.

Definition 15. Let L be a recognisable set of finite traces and ϕ a PLTL$_f$ formula. The *probability* of L w.r.t. ϕ is

$$P_\phi(L) = \max_{t \in L} P_\phi(t).$$

A trace $t \in L$ is a *most likely trace of* L w.r.t. ϕ if it holds that $P_\phi(t) = P_\phi(L)$.

Since L is recognisable, there exists an automaton $\mathcal{A}_L$ that accepts exactly the traces in L. We can obviously see this automaton as a very simple weighted automaton, whose weights are all in $\{0, 1\}$. To find the mlts of L and their corresponding probability, we intersect $\mathcal{A}_L$ with $\mathcal{B}_\phi$ and $\overline{\mathcal{B}_\phi}$, respectively, and compute $\|\mathcal{A}_L \cap \mathcal{B}_\phi\|$ and the language accepted by $\mathcal{A}_L \cap \overline{\mathcal{B}_\phi}$, respectively.

Consider now the same probabilistic problems but in relation to an observed prefix. Given a sequence s of propositional valuations, we want to analyse only traces t that extend s. Formally, if ϕ is a PLTL$_f$ formula, and s is a finite sequence of propositional valuations, a *most likely trace extending* s is a trace $t = s \cdot u$ such that for every trace $t' = s \cdot u'$, $P_\phi(t') \leq P_\phi(t)$. We want to find all the most likely traces extending s, and their probability. Note that the set of all finite words over the alphabet of propositional valuations which extend s is recognisable; indeed, a simple concatenation of the universal automaton to the automaton that accepts only s recognises this language. Hence, our previous results answer this question.

The PLTL$_f^0$ Fragment of PLTL$_f$

We have seen that even the basic task of satisfiability is, in the PLTL$_f$ case, EXPTIME-hard on the length of the formula. To mitigate this complexity, we now focus on the fragment of PLTL$_f$ where probabilities can only appear as the top-most temporal constructor of a conjunctive formula. We call this fragment PLTL$_f^0$. Formally, a PLTL$_f^0$ formula is a finite set of expressions of the form $\circledcirc_{\bowtie p}\varphi$, where φ is a classical LTL$_f$ formula, $\bowtie \in \{\leq, \geq, <, >\}$, and $p \in [0, 1]$.[3] In terms of processes, $\circledcirc_{\bowtie p}\varphi$ expresses that the proportion of traces of the process that satisfy φ is p. The set of formulas is interpreted as a conjunction of the probabilistic formulas appearing in it; that is, a PLTL$_f^0$ formula is a conjunction of probabilistic constraints. Note that in this restricted setting, the probabilistic constructor $\circledcirc$ refers only to the probability of observing a specific LTL$_f$ formula, without a reference to the next point in time. We preserve the same notation, to keep consistent with the general logic PLTL$_f$.

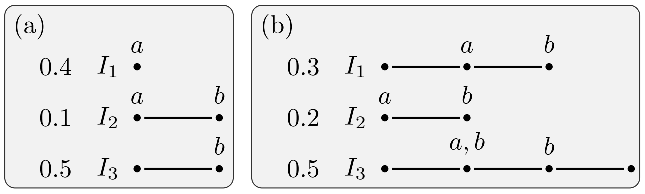

Figure 2: A probabilistic model $\mathcal{P} = (\{I_1, I_2, I_3\}, P)$ of Φ_0. The probability of each interpretation appears on the left.

This logic is interesting for two reasons. On the one hand, reasoning about PLTL$_f^0$ falls down to PSPACE, matching the complexity of the classical LTL$_f$ case (without probabilities). On the other hand, PLTL$_f^0$ is suited for describing declarative constraints mined from historical log data of business process executions. More specifically, some PLTL$_f^0$ patterns can be readily mined from log data using existing declarative process discovery techniques.

Reasoning in PLTL$_f^0$

The superposition semantics of PLTL$_f$ collapse in PLTL$_f^0$ to the more standard multiple-world semantics. To simplify the presentation, we define a *probabilistic interpretation* as a pair $\mathcal{P} = (\mathcal{I}, P_\mathcal{I})$, where $\mathcal{I}$ is a finite set of LTL$_f$ interpretations and $P_\mathcal{I}$ is a discrete probability distribution over $\mathcal{I}$. Satisfiability of an LTL$_f$ formula by an LTL$_f$ interpretation is defined as usual (De Giacomo and Vardi 2013). The probabilistic interpretation $\mathcal{P} = (\mathcal{I}, P_\mathcal{I})$ is a *model* of the PLTL$_f^0$ formula $\Phi = \{\circledcirc_{\bowtie p_i}\varphi_i \mid 1 \leq i \leq n\}$ iff for every $i, 1 \leq i \leq n$ it holds that

$$P_\mathcal{I}(\{I \in \mathcal{I} \mid I \models \varphi_i\}) \bowtie p_i;$$

that is, if the probability of all the models of φ_i is $\bowtie p_i$. For example, the formula ϕ_0 from Example 1 is equivalent to the PLTL$_f^0$ formula $\Phi_0 := \{\circledcirc_{\leq 0.5}a, \circledcirc_{\geq 0.6} \bigcirc b\}$. Two models of this formula are depicted in Figure 2.

Briefly, the uncertainty of PLTL$_f^0$ formulas appears only at the beginning of the process, after which the execution follows a regular LTL$_f$ execution. Thus, there is no need to branch within the superposition semantics at later times; a model becomes a degenerate tree which only branches at the root. Moreover, as all formulas start with the constructor $\circledcirc$, the root node serves only as an anchor for the PLTL$_f$

[3]We consider all the standard abbreviations from LTL$_f$. In particular, $\Diamond\varphi \equiv \top U \varphi$, where $\top$ stands for any tautology, and $\Box\varphi \equiv \neg\Diamond\neg\varphi$.

semantics. Hence, a model can be represented as a sequence of classical LTL$_f$ interpretations. Compare the model in Figure 1 (c) with Figure 2 (a). Interestingly, restricting to PLTL$_f^0$ reduces the complexity of dealing with probabilistic formulas, and allows for simpler algorithms. Consider first the case of deciding whether the PLTL$_f^0$ formula Φ is satisfiable. In practice, this corresponds to verifying whether the class of all possible traces can be divided in such a way that the proportions required by the probabilistic constraints are satisfied. To solve this problem, we may proceed as follows.

Given $\Phi = \{\circledcirc_{\bowtie p_i}\varphi_i \mid 1 \leq i \leq n\}$, analyse the 2^n possible scenarios of a trace, depending which of the constraints are satisfied or violated. More precisely, consider the 2^n sets of constraints in the Cartesian product $\prod_{i=1}^{n}\{\phi_i, \neg\phi_i\}$; i.e., each set chooses for every formula whether it will be satisfied or violated. If each of these sets, seen as the conjunction of the formulas that it contains, is satisfiable, then the input PLTL$_f^0$ formula is satisfiable as well. On the other hand, if any of these sets is unsatisfiable, it means that it is impossible to build a trace that satisfies that combination of formulas; hence that scenario should be assigned probability 0.

To verify that probabilities for the remaining branches can still be assigned consistently with the values in Φ, we build a system of inequalities whose solution space is precisely the valid probability assignments. We consider one variable for each case. For readability, we name these variables $x_0, \ldots, x_{2^n-1}$ using a binary subindex which indicates satisfaction or violation of constraints assuming w.l.o.g. that the constraints are linearly ordered. That is, the subindex is a chain of length n of 0s and 1s; a 0 or a 1 at position i means that $\neg\phi_i$ or ϕ_i is satisfied, respectively. We use the same subindex convention to refer to the sets of constraints S_i; e.g., if Φ contains three formulas, then x_{010} is the variable corresponding to the set $S_{010} = \{\neg\phi_1, \phi_2, \neg\phi_3\}$. Using this information, $\mathcal{L}_\Phi$ is the system of inequalities

$$\begin{aligned}
x_i &\geq 0 && 0 \leq i < 2^n \\
\sum_{i=0}^{2^n-1} x_i &= 1 && \\
\sum_{j\text{th position is 1}} x_i &\bowtie p_j && 0 \leq j < n \\
x_i &= 0 && \text{if } S_i \text{ is unsatisfiable}
\end{aligned}$$

The first two lines guarantee that we assign a non-negative value to each variable, and that their sum is one; we can see these assignments as probabilities. The third line verifies the probability associated to each constraint in Φ: all the variables that correspond to cases making ϕ_i true should add to be $\bowtie p_i$. The last line ensures that the unsatisfiable cases are never assigned a positive probability. This system of inequalities has a solution iff the PLTL$_f^0$ formula is satisfiable.

Example 16. Let $\Phi_1 := \{\circledcirc_{\leq 0.8}\Diamond a, \circledcirc_{\leq 0.7}\Box(a \rightarrow \Diamond b)\}$. Since Φ_1 has two probabilistic constraints, we build four variables and sets

$$\begin{aligned}
S_{00} &:= \{\neg\Diamond a, \neg\Box(a \rightarrow \Diamond b)\}, & S_{01} &:= \{\neg\Diamond a, \Box(a \rightarrow \Diamond b)\}, \\
S_{10} &:= \{\Diamond a, \neg\Box(a \rightarrow \Diamond b)\}, & S_{11} &:= \{\Diamond a, \Box(a \rightarrow \Diamond b)\}.
\end{aligned}$$

S_{00} is clearly unsatisfiable, but the remaining three sets are satisfiable. The system of inequalities must enforce that x_{00} is 0. Specifically, the system $\mathcal{L}_{\Phi_1}$ is

$$\begin{gathered}
x_{00} = 0 \quad x_{01} \geq 0 \quad x_{10} \geq 0 \quad x_{11} \geq 0 \\
x_{00} + x_{01} + x_{10} + x_{11} = 1 \\
x_{10} + x_{11} \leq 0.8 \qquad\qquad x_{01} + x_{11} \leq 0.7
\end{gathered}$$

A solution of $\mathcal{L}_{\Phi_1}$ is $x_{00} = 0$, $x_{01} = 0.2$, $x_{10} = 0.3$, and $x_{11} = 0.5$, which yields a probabilistic model of Φ_1 consisting of three interpretations, I_1, I_2, I_3; each interpretation I_i satisfies the constraints in the set S_i and is assigned the probability x_i. An interpretation for S_0 is not needed because these constraints are unsatisfiable (and the combination of formulas is assigned probability 0).

Theorem 17. *The PLTL$_f^0$ formula Φ is satisfiable iff $\mathcal{L}_\Phi$ has a solution.*

To construct $\mathcal{L}_\Phi$, one must solve 2^n LTL$_f$ satisfiability tests, each requiring polynomial space (Sistla and Clarke 1985). Solving this system of inequalities requires polynomial time on the number of variables; i.e., exponential on n. Overall, it needs exponential time on n, but only polynomial space on the length of Φ.

Theorem 18. *PLTL$_f^0$ satisfiability is decidable in exponential time on the number of probabilistic formulas, but in polynomial space on their total length.*

In particular, if the number of formulas in Φ is bounded, satisfiability is PSPACE-complete, improving the EXPTIME lower bound for general PLTL$_f$ (Theorem 7). We are more interested in deducing (probabilistic) guarantees of a process that follows the constraints in a formula; and more importantly of traces being observed over them. Recall that in our semantics any execution is possible as long as there is no evidence to the contrary. In PLTL$_f^0$ the uncertainty is stated at the beginning; that is, we do not know which of the formulas $\phi_1, \ldots, \phi_n$ are satisfied, but once a trace has chosen its path, it remains in it without further uncertainty arising later on.

As before, we follow an optimistic approach and try to find the most likely scenarios and traces that fit the constraints in the formula. Recall that the solution space of $\mathcal{L}_\Phi$ yields the probability assignments that can be consistently given to the LTL$_f$ interpretations appearing in a probabilistic model according to the constraints that it satisfies. Thus, maximising the value of a variable x_i yields the maximum probability that the set S_i may be assigned in a model.

For each $i, 0 \leq i < 2^n$, let $\max_i$ be the solution of maximising x_i subject to $\mathcal{L}_\Phi$. Note that each variable is maximised independently of the others, and hence the values $\max_i$ do not form a probability distribution *per se*; their sum may be greater than 1. The values $\max_i$ express the *optimistic* position of assigning the highest possible probability to the traces satisfying S_i. For the formula Φ_1 from Example 16, the answers to the maximisation problems for the different variables correspond precisely to the values in the solution presented; namely, $\max_{00} = 0$, $\max_{01} = 0.2$, $\max_{10} = 0.3$, and $\max_{11} = 0.5$.

The question of finding the mlts or simply the *most likely scenario* can be answered using the values $\max_i$. Take the

Algorithm 1: Most likely scenario for t over Φ.

```
Data: $\Phi{=}\{\circledcirc_{\bowtie p_i}\varphi_i | 1{\leq}i{\leq}n\}$ PLTL$_f^0$ formula, $t$ prefix
Result: Index of the most likely scenario for $t$ in $\Phi$
$mls \leftarrow -1$
for $0 \leq i < 2^n$ do
    compute $\max_i$
    if $S_i \Vdash t$ and $\max_i > \max_{mls}$ then
        $mls \leftarrow i$
Return $mls$
```

indices j where $\max_j$ is the largest among all variables; i.e., $j \in \text{argmax}\{\max_i \mid 0{\leq}i{<}2^n\}$. Each S_j is a most likely scenario, and every trace satisfying S_j is an mlt, with probability $\max_j$. In our example, S_{11} is the most likely scenario; i.e, we expect to observe a trace satisfying $\Diamond a$ and $\Box(a \rightarrow \Diamond b)$.

If Φ represents a process, the mlts are those that we would expect to observe in an execution of the process in the absence of other information. In general, it is more interesting to predict the future behaviour of the process, given some observation of its first steps. Given a (partial) trace t, corresponding to the prefix of a full process, we want to find the most likely scenario where t can happen, and predict the potential future evolution of the trace. Given a set S of LTL$_f$ formulas and a prefix t, S *accepts* t (denoted by $S \Vdash t$) iff there is a suffix s such that $S \models t \cdot s$. In words, S accepts a prefix if it can be extended into a trace that satisfies all the conditions in S. To find the most likely scenario accepting a prefix, and a suffix extending it to a successful trace, we generalise the idea described for the case without prefix.

Let $\mathsf{acc}(t) := \{i \mid S_i \Vdash t\}$ be the set of indices j s.t. S_j accepts t, and let $j \in \text{argmax}_{i \in \mathsf{acc}(t)}\{\max_i\}$ be an index with the maximum value in the set of solutions from the maximisation problems of $\mathcal{L}_\Phi$. Then, S_j is a most likely scenario given t, and any trace extending t accepted by S_j is an mlt. For Φ_1 in Example 16, the most likely scenario for any finite prefix t is always S_{11}, and $t \cdot ab$ is always a trace accepted by this set. In general, however, the most likely scenario may change as a prefix grows.

Example 19. Let $\Psi_1 := \{\circledcirc_{\leq 0.5}\Diamond a, \circledcirc_{\leq 0.6}\Box(a \rightarrow \Diamond b)\}$, which is very similar to Φ_1 (Example 16), but with different probabilities. We get $\max_{00} = 0$, $\max_{01} = 0.5$, $\max_{10} = 0.4$, and $\max_{11} = 0.1$. For the empty prefix ε and the prefix $\neg a$, the most likely scenario is S_{01}, which holds with probability 0.5, and a most likely continuation would append them with a finite number of observations of $\neg a$. If at the second point in time we observe a (making the prefix $\neg a \cdot a$) then S_{01} does not accept this prefix anymore, and the most likely scenario becomes S_{10}: we will eventually observe an a after which b will never be observed anymore.

The method for finding the most likely scenario is formalised in Algorithm 1, where $\max_{-1} := 0$. Note that an expensive part of this algorithm is the computation of the values $\max_i$. However, this computation can be made offline, as a preprocessing step before the algorithm is called, as these values remain invariant for any call. A second point to consider is that the set J monotonically decreases as the trace t grows. More precisely, for every t, s, if $S \Vdash t \cdot s$, then $S \Vdash t$. Hence, if we are monitoring the evolution of a process, trying to find out the most likely continuation of the currently observed trace, then after every newly observed step, we only need to update the set J to remove those S_is not accepting the prefix anymore. Finally, to avoid unnecessary tests, we can exclude from the **for** loop all indices i where $\max_i = 0$: $\max_i = 0$ means that the system should not observe any trace satisfying S_i. If the most likely scenario is one that has probability 0, then the observed prefix is violating the conditions described by Φ.

Proposition 20. *Algorithm 1 returns the index of the most likely scenario, given a prefix.*

Interestingly, assuming that all the values $\max_i$ have been computed before, Algorithm 1 can be executed to use only polynomial space. The information to control the **for** loop requires at most n bits, and the two tests within this loop require polynomial space.

Note that finding the probability of the most likely scenario (and trace) is akin to monitoring agreement with a model. Analogously, one can extend the task to monitoring a complex PLTL$_f$ property ψ. For the maximum likelihood of accepting ψ, we use Algorithm 1, but now considering whether $S_i \cup \{\psi\} \Vdash t$; i.e., finding the scenarios where ψ may still be satisfied given the knowledge of t. This is analogous to the notion of *eventual satisfaction* in monitoring. Other notions like *current satisfaction*, or *permanent satisfaction* can be dealt with accordingly, modifying the notion of acceptability of a trace w.r.t. a set of LTL$_f$ constraints (De Giacomo et al. 2014).

Discovering PLTL$_f^0$ Patterns from Event Log Data

PLTL$_f^0$ formulas can be automatically mined from event log data using state-of-the-art *declarative process discovery* algorithms within *process mining*. Process mining focuses on the continuous improvement of business processes based on factual data (van der Aalst 2016). Such data are stored in a so-called *event log*, where each event refers to an *activity* (a well-defined step in a process) and is related to a *case* (a *process instance*). Events in a case are *ordered* and seen as an execution (or *trace*) of the process. A core process mining task is *process discovery*, which learns a process model that reproduces the traces contained in the log. In declarative process discovery, the target model is specified using rules/constraints, like the LTL$_f$ patterns adopted by the Declare process modelling language (Pesic, Schonenberg, and van der Aalst 2007).

Maggi, Chandra Bose, and van der Aalst (2012) developed a two-phase method to automatically infer Declare constraints from event logs. In the first phase, candidate constraints to be mined are generated by an algorithm called Apriori. This algorithm returns frequent activity sets indicating a high correlation between activities involved in an activity set. Highly correlated sets are used to instantiate, in any possible ways, the Declare patterns. For example, considering the frequent activity set $\{a, b\}$ and the Declare pattern of `response`, the two LTL$_f$ constraints $\Box(a \rightarrow \Diamond b)$ and $\Box(b \rightarrow \Diamond a)$ are generated. In the second phase, the set of so-generated constraints is filtered by retaining only "relevant" constraints, where relevance is measured using metrics such as that of *support*: the proportion of traces satisfying the constraint.

What makes Apriori interesing in our setting is that support can be interpreted as the constraint probability: the discovery of LTL_f formula φ with support p (that is, appearing in $100p\%$ of the traces) can be interpreted as the discovery of the PLTL_f^0 formula $\circledcirc_{=p}\varphi$.

Conclusions

We have introduced a new probabilistic temporal logic PLTL_f based on a novel superposition semantics, and its sublogic PLTL_f^0 where this semantics collapses to the standard multiple-world approach. These logics are specifically crafted for predicating about uncertainty in dynamic systems whose executions eventually finish. We studied the main properties of the logics, and provided automata-theoretic methods for extracting relevant information from them.

In future work, we plan to implement the algorithms for PLTL_f^0 and apply them to the declarative modelling and analysis of business processes, considering in particular monitoring and conformance checking.

References

Baader, F.; Hladik, J.; and Peñaloza, R. 2008. Automata can show pspace results for description logics. *Inf. Comput.* 206(9-10):1045–1056.

Baier, C., and Katoen, J.-P. 2008. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press.

Brafman, R. I.; De Giacomo, G.; and Patrizi, F. 2018. Ltlf/ldlf non-markovian rewards. In *Proc. of AAAI*, 1771–1778. AAAI Press.

Carmona, J.; van Dongen, B. F.; Solti, A.; and Weidlich, M. 2018. *Conformance Checking - Relating Processes and Models*. Springer.

Comon, H.; Dauchet, M.; Gilleron, R.; Löding, C.; Jacquemard, F.; Lugiez, D.; Tison, S.; and Tommasi, M. 2007. Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata. release October, 12th 2007.

De Giacomo, G., and Rubin, S. 2018. Automata-theoretic foundations of FOND planning for ltlf and ldlf goals. 4729–4735. ijcai.org.

De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In *Proc. of IJCAI*, 854–860. AAAI Press.

De Giacomo, G.; De Masellis, R.; Grasso, M.; Maggi, F. M.; and Montali, M. 2014. Monitoring business metaconstraints based on LTL & LDL for finite traces. In *Proc. of BPM*, volume 8659 of *LNCS*, 1–17. Springer.

De Giacomo, G.; Maggi, F. M.; Marrella, A.; and Patrizi, F. 2017. On the disruptive effectiveness of automated planning for ltl*f*-based trace alignment. In *Proc. of AAAI*, 3555–3561. AAAI Press.

De Giacomo, G.; De Masellis, R.; and Montali, M. 2014. Reasoning on LTL on finite traces: Insensitivity to infiniteness. In *Proc. of AAAI*, 1027–1033. AAAI Press.

Downey, R. G., and Fellows, M. R. 2012. *Parameterized Complexity*. Springer.

Droste, M.; Kuich, W.; and Vogler, H. 2009. *Handbook of Weighted Automata*. EATCS. Springer, 1st edition.

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: Pddl3 and experimental evaluation of the planners. *Artif. Intell.* 173(5-6):619–668.

Konur, S. 2010. Real-time and probabilistic temporal logics: An overview. *CoRR* abs/1005.3200.

Kovtunova, A., and Peñaloza, R. 2018. Cutting diamonds: A temporal logic with probabilistic distributions. In *Proc. of KR*, 561–570. AAAI Press.

Maggi, F. M.; Montali, M.; Westergaard, M.; and van der Aalst, W. M. P. 2011. Monitoring business constraints with linear temporal logic: An approach based on colored automata. LNCS, 132–147. Springer.

Maggi, F. M.; Chandra Bose, R. P. J.; and van der Aalst, W. M. P. 2012. Efficient discovery of understandable declarative process models from event logs. In *Proc. of CAiSE*, LNCS, 270–285.

Maggi, F. M.; Montali, M.; and Peñaloza, R. 2019. Temporal logics over finite traces with uncertainty (technical report).

Montali, M. 2010. *Specification and Verification of Declarative Open Interaction Models: a Logic-Based Approach*, volume 56 of *LNBIP*. Springer.

Morão, L. 2011. Probabilistic distributed temporal logic. Master thesis, Universidade Técnica de Lisboa, Portugal.

Ognjanovic, Z. 2006. Discrete linear-time probabilistic logics: Completeness, decidability and complexity. *J. Log. Comput.* 16(2):257–285.

Pesic, M.; Schonenberg, H.; and van der Aalst, W. M. P. 2007. Declare: Full support for loosely-structured processes. In *Proc. of EDOC*, 287–300. IEEE Computer Society.

Savitch, W. J. 1970. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences* 4(2):177 – 192.

Seidl, H. 1994. Haskell overloading is dexptime-complete. *Information Processing Letters* 52(2):57 – 60.

Sistla, A. P., and Clarke, E. M. 1985. The complexity of propositional linear temporal logics. *J. ACM* 32(3):733–749.

van der Aalst, W. M. P. 2016. *Process Mining - Data Science in Action, Second Edition*. Springer.

Vardi, M. Y., and Wolper, P. 1986. Automata-theoretic techniques for modal logics of programs. *J. Comput. Syst. Sci.* 32(2):183–221.

Woltzenlogel Paleo, B. 2016. An expressive probabilistic temporal logic. *CoRR* abs/1603.07453.