Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Probabilistic Synthesis and Verification for LTL on Finite Traces

Benjamin Aminof[1] , **Linus Cooper**[2] , **Sasha Rubin**[2] , **Moshe Y. Vardi**[3] , **Florian Zuleger**[4]
[1]University of Rome "La Sapienza"
[2]University of Sydney
[3]Rice University
[4]Technical University of Vienna

Abstract

We study synthesis and verification of probabilistic models and specifications over finite traces. Probabilistic models are formalized in this work as Markov Chains and Markov Decisions Processes. Motivated by the recent attention given to, and importance of, finite-trace specifications in AI, we use linear-temporal logic on finite traces as a specification formalism for properties of traces with finite but unbounded time horizons. Since there is no bound on the time horizon, our Markov chains generate infinite traces, and we consider two possible semantics: "existential (resp. universal) prefix-semantics" which says that the finite-trace property holds on some (resp. every) finite prefix of the trace. For both types of semantics, we study two computational problems: the verification problem — "does a given Markov chain satisfy the specification with probability one?"; and the synthesis problem — "find a strategy (if there is one) that ensures the Markov decision process satisfies the specification with probability one". We provide optimal algorithms that follow an automata-theoretic approach, and prove that the complexity of the synthesis problem is 2EXPTIME-complete for both semantics, and that for the verification problem it is PSPACE-complete for the universal-prefix semantics, but EXPSPACE-complete for the existential-prefix semantics.

1 Introduction

Linear-temporal logic on finite traces (LTLf) is a specification formalism for properties of traces with finite but unbounded time horizons (De Giacomo and Vardi 2013). It, and its variants, are extensively used in AI and CS, e.g., in Planning to represent trajectory constraints (Bacchus and Kabanza 2000; Baier and McIlraith 2006; Gerevini et al. 2009), and in Business Process Management to specify services and processes (Pesic and van der Aalst 2006; Montali et al. 2010; De Giacomo et al. 2014; Geatti, Montali, and Rivkin 2024). Moreover, interpreting temporal logics not only on traces but also on state-transition systems allows one to model and reason about planning domains and business processes (rather than just plans or logs).

In this work, we study computational problems of LTLf on discrete-time probabilistic state-transition systems: Markov decision processes (MDPs), which model an agent operating strategically in a probabilistic environment; and Markov chains, which model an agent's policy executing in a probabilistic environment. In the probabilistic setting, one associates a probability to each specification property, i.e., the probability that an execution of the system (consistent with a given policy in the MDP case) satisfies the property.

In this work we focus on the probabilistic analogue of satisfaction, i.e., that this probability is equal to one. In contrast, the standard notion of satisfaction says that *every* system execution satisfies the specification.

Since we are interested in systems that generate executions that are of finite but unbounded length, our probabilistic systems should generate traces of arbitrary finite length. This can be done in a number of natural ways: (i) for a parameter $L \in \mathbb{N}$, have the system generate a trace of length exactly L (and consider increasing values of L); (ii) require that the Markov chains are absorbing, and condition on the event that the trace reaches an absorbing state (which happens with probability 1); (iii) have the system only generate infinite traces, and interpret the logical formula on prefixes of the trace. In this work we follow the third approach because we feel it provides a rich and natural setting that does not constrain the system as (ii) does, and does not initially bound the length of the traces as (i) does. (Note that (i) or (ii) may also be appropriate design choices, depending on the application).

To implement (iii) we must define what it means for an infinite execution to satisfy an LTLf formula, whose usual semantics is given only for finite traces. We use the "existential prefix semantics" that says that the property holds on some finite prefix of the execution, and its dual — "universal prefix semantics" — that says that the property holds on every finite prefix of the execution. Note that the existential (resp. universal) prefix semantics intuitively corresponds to the agent (resp. environment) using a 'stop' action.

We study two foundational problems: synthesis and verification. The synthesis problem with LTLf objectives is a form of planning with temporally extended goals, and has recently been applied to synthesizing controllers for robots, e.g., manipulation domains (Wells et al. 2021) and motion planning and control (Lahijanian et al. 2010); and to model stochastic actions in the form of the trembling-hand problem for LTLf planning (Yu et al. 2024). Verification is a counterpart to synthesis: if a procedure (e.g., that uses machine learning) returns a policy claimed to solve a given problem, verification can be used to check that it indeed does.

Specifically, the *probabilistic verification* problem is to

decide, given a Markov chain M and an LTLf formula F, if the probability that some (resp. every) finite prefix of a random trace generated by M satisfies F. The *probabilistic synthesis* problem is, given a Markov Decision Process D and an LTLf formula F, to find a strategy S (if one exists) such that the probability that some (resp. every) finite prefix of a random trace generated by M and S satisfies F. The corresponding decision problem, called *realizability*, simply asks to decide if there is such a strategy.

We provide algorithms for all these problems, following an automata-theoretic approach that first compiles, in linear time, the given LTLf formula into an equivalent alternating finite automaton (AFA). The verification algorithms rely on characterizations of when the probability that some (resp. every) finite prefix of a random trace is accepted by the AFA; the case of universal semantics is straightforward, while the existential semantics is quite more subtle. The synthesis algorithms first compile the AFA into equivalent deterministic finite automata (DFA), take a product of the MDP with the DFA, and solve the synthesis problem for the simpler reachability objective on the product.

To show that our algorithms are optimal, we provide matching lower bounds. For verification, once again the universal semantics is straightforward, while the existential semantics is subtle and reduces from the acceptance problem for exponential-space Turing machines. This is done by adapting the encoding from (Vardi and Stockmeyer 1985), also found in (Kupferman and Vardi 2000; Bansal et al. 2023), to the probabilistic setting. Intuitively, those techniques show how to produce a polynomial sized formula (in the size of the input word to the Turing machine) expressing the existence of an accepting run. Unfortunately, *a single trace has probability zero* and thus, the challenge is to construct things in such a way that a single accepting run yields a positive probability of satisfying the formula, while maintaining a zero probability for satisfying the formula if there is no accepting run. For synthesis, we reduce from the LTLf synthesis problem, known to be 2EXPTIME-complete (De Giacomo and Vardi 2015). This problem has a natural universal semantics (is there a strategy such that every finite trace consistent with it satisfies the given formula). Thus, once again, the universal semantics is straightforward, while the existential semantics is subtle since it requires simulating the universal semantics by the existential semantics.

Related Work Probabilistic model-checking (Baier et al. 2018) traditionally focuses on infinite-trace properties, e.g., those described in LTL (Courcoubetis and Yannakakis 1995) or even those expressible in probabilistic variants of branching time logics (Bianco and de Alfaro 1995). The same goes for strategic logics such as Probabilistic Strategy Logic (Aminof et al. 2019) in which one can naturally express different forms of synthesis.

In planning for reachability goals, so-called "strong-cyclic solutions" are strategies that ensure, with probability 1, that the goal state is reached (Cimatti et al. 2003; D'Ippolito, Rodríguez, and Sardiña 2018), i.e., probabilistic synthesis for the LTLf formula F goal under existential semantics. While this problem is equivalent to synthesis under state-action fairness, this equivalence breaks down for general LTLf formulas (Aminof, Giacomo, and Rubin 2020). Stochastic best-effort synthesis is a refinement of strong-cyclic planning that more fully exploits stochasticity, and has been studied under existential semantics (Aminof et al. 2022) and for LTL (Aminof et al. 2023).

Algorithms for verification of LTL properties have been treated both syntactically on the structure of the formula (Courcoubetis and Yannakakis 1995), and more generally with alternating Büchi automata (Bustan, Rubin, and Vardi 2004). We also take an automata-theoretic approach, but only need to compile the given LTLf formula into the simpler alternating automata on finite words (using classic subset constructions), and handle the prefix-quantification separately. We remark that "existential prefix semantics" introduces an alternation of quantifiers that is not present in the case of ordinary LTL (with infinite-trace semantics). Practical algorithms for solving the probabilistic synthesis problem for LTLf specifications under existential semantics are explored in (Wells et al. 2020).

Motivated by bringing the power of LTLf as a declarative specification language to bear on probabilistic processes, Alman et al. 2022 study probabilistic declarative process mining. That work considers different computational problems than we do, i.e., constraint discovery, monitoring, and conformance checking, and uses probabilities slightly differently: it gives a probabilistic interpretation to a log (finite multi-set of finite traces), interpreting multiplicities as frequencies. In particular, its sample spaces of interest are finite, while ours are infinite. Indeed, in our setting we are given access to the underlying probabilistic process, allowing a single process to generate infinitely many traces. Similarly, Maggi, Montali, and Penaloza 2020 define a probabilistic logic (PLTLf) whose models are finite trees and study satisfiability and the computation of the most likely trace. In contrast to us, their sample spaces are again finite.

Model-checking and formal reasoning of nondeterministic systems, i.e., systems with non-probabilistic uncertainty, have a long history (Clarke et al. 2018). Of particular relevance to our EXPSPACE result are techniques that handle satisfiability of branching-time logic on finite traces (Vardi and Stockmeyer 1985; Kupferman and Vardi 2000) (since, intuitively, path quantifiers are related to prefix quantification) as well as model-checking nondeterministic systems for LTLf with existential semantics (Bansal et al. 2023). The latter work, motivated by the problem of model-checking finite-state history-dependent policies (aka, strategies) against LTLf specifications, proves that the verification problem for nondeterministic systems against LTLf specifications with existential-prefix semantics is EXPSPACE-complete. Our EXPSPACE upper bound, however, requires more sophistication since we are not checking that all traces satisfy the property, only that almost-all do. The EXPSPACE lower bounds here and in (Bansal et al. 2023) build on lower-bound techniques from (Vardi and Stockmeyer 1985; Kupferman and Vardi 2000). Bansal et al. also consider "terminating semantics" in which the system only generates finite executions ending in a given set of "terminating states", akin to (ii) mentioned in the introduction.

2 Preliminaries

In this section we recall the definitions of Linear-temporal logic on finite traces (LTLf), alternating, non-deterministic, and deterministic automata on finite words (AFA, NFA, DFA), Markov chains (MC), and Markov Decision Processes (MDP).

Sequences Let Σ denote a finite alphabet. The set of finite (resp. infinite) sequences over Σ is denoted Σ^* (resp. Σ^ω). The empty sequence is denoted ε. Indexing will start with 0, so we write $x = x_0 x_1 \cdots$. The length of a sequence x is denoted $|x| \in \mathbb{N} \cup \{\infty\}$. For $i < |x|$, the prefix $x_0 x_1 \cdots x_i$ of x is denoted $x_{\leq i}$ (note that it has length $i+1$). For a finite sequence $h \in \Sigma^*$ let $last(h) \in \Sigma$ denote its last element. If h is a prefix of h' we say that h' *extends* h; if, in addition, $h \neq h'$, then we say that h is a *proper prefix* of h' and that h' is a *proper extension* of h. For a subset L of X, the complement of L in X may simply be denoted $\overline{L}$ if X is clear from the context. If $x \in \Sigma^*$ we let *cone*$(x) \subseteq \Sigma^\omega$ denote the set of infinite sequences that extend x. Sequences will also be called *words*. If $\Sigma = 2^{AP}$ for a set AP of atoms (aka atomic predicates), then sequences over Σ are called *traces*. If L is a set of traces and $\tau \in L$ then we say that τ *satisfies* L.

Probability Distributions For a finite set X, let $Dbn(X)$ denote the set of *distributions* over X, i.e., functions $d : X \to [0,1]$ such that $\sum_{x \in X} d(x) = 1$. If $d(x) > 0$, we will write $x \in d$, and say that x is in the *support* of d.

Linear temporal logic on finite traces (LTLf) Fix a set AP of atomic propositions. The formulas of LTLf over AP are defined by the following BNF (where $p \in AP$):

$$\varphi ::= p \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi$$

Here X is called the *next* operator, and U the *until* operator. We use the usual abbreviations, $\varphi \supset \varphi' \doteq \neg\varphi \vee \varphi'$, $\mathsf{true} \doteq p \vee \neg p$, $\mathsf{F}\,\varphi \doteq \mathsf{true}\,\mathsf{U}\,\varphi$ (read *eventually*), $\mathsf{G}\,\varphi \doteq \neg\,\mathsf{F}\neg\varphi$ (read *always*), $\tilde{\mathsf{X}}\,\varphi \doteq \neg\,\mathsf{X}\neg\varphi$ (read *weak next*, which says that if there is a next step, then φ holds in the next step), and $\varphi_1\,\mathsf{W}\,\varphi_2 \doteq \mathsf{G}\,\varphi_1 \vee (\varphi_1\,\mathsf{U}\,\varphi_2)$ (read *weak until*). The *size* $|\varphi|$ of a formula φ is the number of symbols in it.

Given a finite trace τ, an integer n with $0 \leq n < |\tau|$, and an LTLf formula φ, the satisfaction relation $(\tau, n) \models \varphi$, stating that φ holds at step n of τ, is defined as follows:

- $(\tau, n) \models p$ iff $p \in \tau_n$;
- $(\tau, n) \models \varphi_1 \vee \varphi_2$ iff $(\tau, n) \models \varphi_1$ or $(\tau, n) \models \varphi_2$;
- $(\tau, n) \models \neg\varphi$ iff it is not the case that $(\tau, n) \models \varphi$;
- $(\tau, n) \models \mathsf{X}\,\varphi$ iff $n + 1 < |\tau|$ and $(\tau, n+1) \models \varphi$;
- $(\tau, n) \models \varphi_1\,\mathsf{U}\,\varphi_2$ iff $(\tau, m) \models \varphi_2$ for some m with $n \leq m < |\tau|$, and $(\tau, j) \models \varphi_1$ for all j with $n \leq j < m$.

Write $\tau \models \varphi$ if $(\tau, 0) \models \varphi$, read τ *satisfies* φ. The set of all traces that satisfy φ is denoted $L(\varphi)$.

Markov Chains A *labeled Markov chain (MC)* $M = (T, t_0, P, AP, lab)$ consists of: a finite set T of *states*, an *initial state* $t_0 \in T$, a *probability measure* $P : T \to Dbn(T)$, a finite set of *atomic propositions* AP, and a *labeling function* $lab : T \to 2^{AP}$. We will also write $P(i,j)$ instead of $P(i)(j)$ (this suggestive notation reflects the idea that there is a transition from i to j). A Markov chain induces a probability space in the usual way (Puterman 2014): the sample space is T^ω, and the probability function, written $Pr_M(-)$, is determined by its value on the cones, i.e., for a word $x = x_0 x_1 \cdots x_n$, define $Pr_M(cone(x))$ to be $\prod_{i=0}^{n-1} P(x_i, x_{i+1})$ if $x_0 = t_0$, and otherwise define it to be zero [1]. Finite/infinite words over T are called *histories/plays*. A history x is *generated by* M if $Pr_M(cone(x)) > 0$, i.e., if $x_0 = t_0$ and $P(x_i, x_{i+1}) > 0$ for all $i < n$. A play x is *generated by* M if all its proper prefixes are. We extend *lab* to histories/plays component-wise, i.e., $lab(x_0 x_1 \cdots) = lab(x_0) \cdot lab(x_1) \cdots$. We say a (finite or infinite) trace w is *generated* by M, if there is some history/play x generated by M such that $w = lab(x)$. For a set of infinite traces $L \subseteq (2^{AP})^\omega$, we define $Pr_M(L) \doteq Pr_M(X_L)$ for the set X_L that consists of all plays x with $lab(x) \in L$; we will only use this notation where X_L is measurable. If $Pr_M(L) = 1$ we say that the Markov chain M *almost-surely models* L.

Markov Decision Processes A *labeled Markov Decision Process (MDP)* $D = (St, s_0, Act, \Delta, AP, lab)$ consists of:

- a finite set St of *states*,
- an *initial* state $s_0 \in St$,
- a finite set Act of agent actions,
- a *transition function* $\Delta : St \times Act \to Dbn(St)$,
- a finite set of *atomic propositions* AP,
- a *labeling function* $lab : St \to 2^{AP}$.

Intuitively, the value $\Delta(s,a)(s')$ is the probability of a transition from s to s' when action a is selected. If s' is in the support of $\Delta(s,a)$ we will write $s' \in \Delta(s,a)$. We will also write $\Delta(s,a,s')$ instead of $\Delta(s,a)(s')$. Finite/infinite words over St are called *histories/plays*. The set of all histories is denoted by $Hist$. A *strategy* is a function $\sigma : Hist \to Act$; intuitively, this tells the agent what to do given what has happened before. A history/play x is *consistent* with a strategy σ if $x_0 = s_0$ and for every $i \geq 0$ with $i + 1 < |x|$ we have that $x_{i+1} \in \Delta(x_i, \sigma(x_{\leq i}))$. In this case, we say that x is a *σ-play* if x is infinite, and a *σ-history* if x is finite. A Markov decision process D and a strategy σ induce a probability space in the usual way (Puterman 2014): the sample space is St^ω, and the probability measure $Pr_{D,\sigma}(-)$ is determined by its value on the cones, i.e., for a finite sequence $h = h_0 h_1 \cdots h_n$ of states, if h is a σ-history then define $Pr_{D,\sigma}(cone(h)) = \prod_{0 \leq i < n} \Delta(h_i, \sigma(h_{\leq i}), h_{i+1})$, and otherwise define it to be zero. We extend the labeling function to plays/histories as follows: $lab(x) = lab(x_0) lab(x_1) \cdots$. For a set of infinite traces $L \subseteq (2^{AP})^\omega$, we define $Pr_{D,\sigma}(L) \doteq Pr_{D,\sigma}(X_L)$ for the set X_L of all plays x with $lab(x) \in L$; we will only use this notation where X_L is measurable. If $Pr_{D,\sigma}(L) = 1$ we say that the strategy σ *almost-surely enforces* L.

[1] As usual, the associated event space is the σ-algebra generated by the cones *cone*$(x) \subseteq T^\omega$ (for words $x \in T^*$).

Alternating Automata For a set S, let $B^+(S)$ denote the set of positive Boolean formulas over S; these are constructed from elements in S, the logical constants $\top, \bot$, conjunctions, and disjunctions. Elements of S are thus also called *atoms*. A function $f : S \to \{0,1\}$ is called a *valuation*, and we will also view it as the set of atoms $\{s \in S : f(s) = 1\}$. Thus, we may talk about a set $X \subseteq S$ satisfying a formula $\varphi \in B^+(S)$. For instance, if $\varphi = (s_1 \wedge s_2) \vee s_3$ then the set $\{s_1, s_3\}$ satisfies φ while the set $\{s_1\}$ does not. An *alternating finite-word automaton (AFA)* $A = (\Sigma, S, \delta_0, \delta, F)$ consists of:

- a finite *alphabet* Σ of symbols,
- a finite set S of *states*,
- an *initial formula* $\delta_0 \in B^+(S)$,
- a *transition function* $\delta : S \times \Sigma \to B^+(S)$.
- a set $F \subseteq S$ of *final states*,

To define acceptance we first define $\mathrm{Acc} : \Sigma^* \to 2^Q$ recursively as follows: $\mathrm{Acc}(\varepsilon) = F$; $\mathrm{Acc}(aw)$ is the set of states p such that for some $X \models \delta(p, a)$ we have $X \subseteq Acc(w)$. A word w is *accepted* by A if there exists $X \models \delta_0$ such that $X \subseteq \mathrm{Acc}(w)$. The *language of* A, denoted $L(A)$, is the set of words for which A has an accepting run. If $L(A) = L(B)$ then we call A and B *equivalent*. The size of an AFA, $|A|$, is defined as $|\delta_0| + \sum_{q \in S, x \in \Sigma} |\delta(q, x)|$. An AFA is called a *nondeterministic finite-word automaton (NFA)* if δ_0 and each formula $\delta(s, a)$ is a disjunction of atoms; in this case we also view each such formula as the set of its atoms (and thus, e.g., we can form the union of such sets). An AFA is called a *deterministic finite-word automaton (DFA)* if $\delta_0 \in S$ and $\delta(s, a) \in S$ (for every s, a), i.e., if each formula is an atom. Two representations of the same language are called *equivalent*. For example, we call an LTLf formula φ and an AFA A equivalent, if $L(\varphi) = L(A)$. Moreover, if $\tau \in L(A)$ we say that τ *satisfies* A. We recall the following conversions that apply subset constructions:

Lemma 1. *(Vardi 1995) 1. An AFA can be converted into an equivalent NFA in exponential time. 2. An NFA can be converted into an equivalent DFA in exponential time. 3. An AFA can be converted into an equivalent DFA in double exponential time (by combining the two constructions above).*

The importance of AFA is that they serve as a general specification formalism for properties expressible in LTLf (as well as other formalism such as LDLf — linear dynamic logic on finite traces):

Lemma 2. *(De Giacomo and Vardi 2013) For every LTLf formula φ one can build an equivalent AFA A_φ of size linear in φ.*

We will use that AFA are efficiently closed under complementation, i.e., one can convert an AFA into an AFA for the complement language in linear time, see e.g., (Vardi 1995).

3 Statement of Problems and Results

As discussed in the introduction, since we are interested in finite but unbounded time horizons, we consider Markov chains that generate infinite executions. As a consequence, we must extend the semantics of LTLf, which is given for finite traces, to infinite traces. A natural way to do so is to say that an infinite trace satisfies an LTLf formula if some finite prefix of it does (De Giacomo and Vardi 2015; Bansal et al. 2023); we call this the *existential-prefix semantics of LTLf on infinite traces*, which we shorten to $\exists$-*semantics*. This corresponds to the agent having a 'stop' action. For symmetry, and inspired by the duality between *safety* and *guarantee* properties in the hierarchy of (Manna and Pnueli 1990), we also consider the case that the environment determines when to 'stop'; we call this the *universal-prefix semantics of LTLf on infinite traces*, which we shorten to $\forall$-*semantics*. We give a general definition:

Definition 1. *For a set $L \subseteq (2^{AP})^+$ of finite traces, and an infinite trace $\tau \in (2^{AP})^\omega$, we say that:*

1. *τ satisfies L with $\forall$-semantics if every finite prefix of τ is in L,*
2. *τ satisfies L with $\exists$-semantics if some finite prefix of τ is in L.*

For $\alpha \in \{\forall, \exists\}$, write $[L]_\alpha$ for the set of infinite traces τ that satisfy L with the α-semantics. We will use the following shorthand: write $[\varphi]_\alpha$ instead of $[L(\varphi)]_\varphi$ and $[A]_\alpha$ instead of $[L(A)]_\alpha$.

We have the following duality between the semantics:

$$\overline{[L]_\exists} = [\overline{L}]_\forall \tag{1}$$

Intuitively, $Pr_M([L]_\forall)$ (resp. $Pr_M([L]_\exists)$) is the probability that every prefix (resp. some prefix) of an infinite trace of M is in L.[2] From (1) we immediately get:

Lemma 3. $Pr_M([L]_\exists) + Pr_M([\overline{L}]_\forall) = 1$.

As the following example shows, it is not the case, in general, that $Pr_M([L]_\exists) + Pr_M([\overline{L}]_\exists) = 1$ (note that we complemented the language but kept the existential semantics).

Example 1. Consider a Markov chain that captures repeated flipping of a fair coin, with the coin initially showing Heads, and an LTLf formula φ that captures that the coin always shows Heads. Then the probability that some prefix satisfies φ is 1 (since every trace has a prefix that does). On the other hand, $\neg\varphi$ says that the coin eventually doesn't show Heads, and this has probability 1. Formally, let $AP = \{H\}$, and let M be the following Markov chain: the state set is $\{s_0, s_1\}$, the initial state is s_1, the label of s_0 is $\emptyset$, the label of s_1 is $\{H\}$, and the transition from s_i to s_j (for $i, j \in \{0, 1\}$) is equal to $\frac{1}{2}$. Let φ be the LTLf formula $\mathsf{G}\, H$. Then $Pr_M([\varphi]_\exists) = 1$ since indeed every trace has some prefix that satisfies φ since H is true in the initial state s_1; but also $Pr_M([\neg\varphi]_\exists) = \sum_i (\frac{1}{2})^i = 1$.

3.1 Probabilistic Verification

We now define two qualitative verification problems for LTLf specifications, and state our results.

[2] The set of plays in M labeled by words in $[L]_\alpha$ is a measurable set, i.e., is a set in the event space. The reason for this is that $[L]_\exists$ is a union of cones, and $[L]_\forall$ is the complement of a union of cones.

Definition 2. *Fix $\alpha \in \{\exists, \forall\}$. The* probabilistic verification problem for LTLf under α-semantics *is to decide, given a Markov chain M and an LTLf formula φ, whether $Pr_M([\varphi]_\alpha) = 1$.*

We prove the following:

Theorem 1. *1. The probabilistic verification problem for LTLf specifications under $\forall$-semantics is PSPACE-complete.*

2. The probabilistic verification problem for LTLf specifications under $\exists$-semantics is EXPSPACE-complete.

Remark 1. Following (Vardi 1985), the probabilistic verification problem "$Pr_M(-) = 1$?" may also be called the *probabilistic universality* problem. Dually, the *probabilistic non-emptiness* problem is to decide ""$Pr_M(-) > 0$?". In these terms, our work also solves the probabilistic non-emptiness problem for LTLf with α-semantics. Indeed, by Lemma 3 we have that $Pr_M([\varphi]_\exists) > 0$ iff $Pr_M([\neg\varphi]_\forall) < 1$; and $Pr_M([\varphi]_\forall) > 0$ iff $Pr_M([\neg\varphi]_\exists) < 1$. Since LTLf is closed under negation, and deterministic complexity classes are closed under complementation, we can rephrase Theorem 1 as follows:

1. The probabilistic non-emptiness problem for LTLf under $\exists$-semantics is PSPACE-complete.
2. The probabilistic non-emptiness problem for LTLf under $\forall$-semantics is EXPSPACE-complete.

3.2 Probabilistic synthesis

We now define two qualitative synthesis problems for LTLf specifications, and state our results.

Definition 3. *Fix $\alpha \in \{\exists, \forall\}$. The* probabilistic synthesis problem for α-semantics *is to decide, given a Markov Decision Process D and an LTLf formula φ, whether there exists a strategy σ such that $Pr_{D,\sigma}([\varphi]_\alpha) = 1$.*

We prove the following:

Theorem 2. *1. The probabilistic synthesis problem for LTLf specifications under $\forall$-semantics is 2EXPTIME-complete.*

2. The probabilistic synthesis problem for LTLf specifications under $\exists$-semantics is 2EXPTIME-complete.

4 Probabilistic Verification: Upper Bounds

In this section we prove the upper bounds in Theorem 1. Let x be a history generated by M. For a language $L \subseteq (2^{AP})^*$, and $\alpha \in \{\exists, \forall\}$, write $cone(x) \subseteq [L]_\alpha$ to mean that every play x' generated by M that extends x satisfies $lab(x') \in [L]_\alpha$. In this case we say that *the cone of M at x is contained in L under α-semantics*. Obviously, if there is such an x then $Pr_M([L]_\alpha) > 0$ since $Pr_M(cone(x)) > 0$. The converse also holds:

Proposition 1. *Given a Markov chain M, a regular language $L \subseteq (2^{AP})^*$, and $\alpha \in \{\exists, \forall\}$, we have that $Pr_M([L]_\alpha) > 0$ iff some cone of M is contained in L under α-semantics.*[3]

[3]We remark that for $\alpha = \exists$ this holds even if L is not regular.

Proof. The forward direction is immediate because every cone has positive probability. For the reverse direction, assume that $Pr_M([L]_\alpha) > 0$. We will show that some cone of M is contained in L under α-semantics.

For the case that $\alpha = \exists$, observe that $Pr_M([L]_\alpha) > 0$ implies that there is some play y generated by M such that $lab(y) \in [L]_\exists$. Let x be a prefix of y such that $lab(y) \in L$, and note that $cone(x) \subseteq [L]_\exists$.

For the case that $\alpha = \forall$, we will use the assumption that L is regular, take a DFA A with $L(A) = L$. and form the product Markov chain $M' = A \times M$. Intuitively, the product Markov chain runs both A and M at the same time: if the current state of $A \times M$ is (s, t) then A deterministically updates its state on input letter $lab(t)$, and M randomly generates its next state. Formally, given a DFA $A = (2^{AP}, S, \delta_0, \delta, F)$, and an MC $M = (T, t_0, P, AP, lab)$, the *product* $A \times M$ is the MC (T', t'_0, P', AP, lab') where $T' = S \times T$; and $t'_0 = (\delta_0, t_0)$; and $P'((s_1, t_1), (s_2, t_2)) = P(t_1, t_2)$ if $\delta(s_1, lab(t_1)) = s_2$, and is equal to 0 otherwise; and $lab'((s, t)) = lab(t)$.

If the play $t = t_0 t_1 \cdots$ is generated by M then, writing $s_0 s_1 \ldots$ for the (unique) run of the DFA A on $lab(t)$, we have that M' generates the play $t' = (s_0, t_0)(s_1, t_1) \cdots$. Say that t' is induced by t. For $X \subseteq T^\omega$ let $X' \subseteq (T')^\omega$ consist of the set of induced plays t' for $t \in X$. Note that $Pr_M(X) = Pr_{M'}(X)$ (intuitively this is because $A \times M$ is like M with states of M simply annotated by states of A).

View M' as a directed graph $G' = (T', E')$, where $(t'_1, t'_2) \in E'$ iff $P'(t'_1, t'_2) > 0$, by ignoring the exact values of non-zero probabilities. Consider the strongly-connected components (SCCs) of G'. As usual, there is a natural partial order on the SCCs, i.e., $C_1 \geq C_2$ if there is some path in the graph from a vertex in C_1 to a vertex in C_2. A SCC is *bottom* if it is a minimal element in this partial order.

Let R be the event in M' consisting of the plays generated by M' that start in t'_0, reach some bottom SCC C, and visit every vertex in C. It is well known that $Pr_{M'}(R) = 1$. Recall that we assumed that $Pr_M([L]_\forall) > 0$ and that A is equivalent to L, and conclude that $Pr_{M'}([A]_\forall | R) = Pr_{M'}([A]_\forall) > 0$. Thus, there is a play $t' = (s_0, t_0)(s_1, t_1), \cdots$ generated by M' that reaches a bottom SCC, say C, visits every vertex in C, and has the property that it only visits final states of F (i.e., $s_i \in F$ for all i). Let $x' = t'_{\leq k}$ be a prefix of t' that reaches C, and conclude that (since C is a bottom SCC) that every play generated by M' that extends x also only ever visits final states of F. It follows that $cone(t_0 t_1 \cdots t_k) \subseteq [L]_\forall$, and we are done. □

The following will be used for solving the verification problem for the existential semantics:

Corollary 1. *Given a Markov chain M, and a regular language $L \subseteq (2^{AP})^*$, we have that $Pr_M([L]_\exists) = 1$ iff for every history x generated by M there is a history y generated by M, that is either a prefix or an extension of x, with $lab(y) \in L$.*

4.1 Universal Semantics

Instead of proving the upper bound for item 1 as stated in Theorem 1, we prove the upper bound for the rephrased ver-

sion of item 1 given in Remark 1. I.e., we show that the probabilistic non-emptiness problem under $\exists$-semantics is in PSPACE, as follows: given M, φ, build (using Lemma 2) an AFA A of linear size equivalent to φ, and then apply the following proposition.

Proposition 2. *For an AFA A, deciding whether $Pr_M([A]_\exists) > 0$ can be done in polynomial space.*

Proof. We will provide a nondeterministic algorithm that uses polynomial space.[4] Since NPSPACE = PSPACE (Savitch 1970), the result follows.

Algorithm. Given an AFA $A = (\Sigma, S, \delta_0, \delta, F)$ with $\Sigma = 2^{AP}$, and a Markov chain $M = (T, t_0, P, AP, lab)$. The algorithm proceeds in rounds, and has a variable for a state of M (written q_i in round i) and a variable for a set of states of A (written S_i in round i).

1. Initialize q_0 to be t_0 (the initial state of M) and nondeterministically select S_0 such that $S_0 \models \delta_0$.
2. At step $i \geq 0$:
   (a) Nondeterministically select a state q_{i+1} of M such that $P(q_i, q_{i+1}) > 0$.
   (b) Nondeterministically select S_{i+1} such that $S_{i+1} \models \bigwedge_{s \in S_i} \delta(s, lab(q_i))$. If $S_{i+1} \subseteq F$ then accept.

Correctness. This is immediate: the algorithm has an accepting computation-branch iff M generates some word $x = t_0 t_1 \cdots t_k$ with $lab(x) \in L(A)$; and by Proposition 1 this holds iff $Pr_M([A]_\exists) > 0$.

Complexity. The space requirement is polynomial: as q_{i+1} and S_{i+1} only depend on q_i and S_i, the algorithm only needs to store at most two q_i and two S_i at any point (the current and the next); the binary representation of q_i (resp. S_i) is logarithmic (resp. linear) in the input. □

We remark that the algorithm can be thought of as doing the following: derive from M a Kripke Structure $K = (T, t_0, \delta, AP, lab)$ where $\delta \subseteq T \times T$ is a transition relation with $(t, t') \in \delta$ iff $P(t, t') > 0$. Then, take the product of A and K, to get a new input-free AFA, and check in PSPACE the non-emptiness of this AFA.

4.2 Existential Semantics

We can solve the probabilistic verification problem for existential semantics for M, φ by first applying Lemma 2 to build an AFA A of linear size for φ, and then apply the following proposition:

Proposition 3. *For a Markov chain M, and an AFA A, deciding if $Pr_M([A]_\exists) = 1$ can be done in exponential space.*

Proof. We provide an alternating algorithm that uses exponential space and a single alternation. Since the number of alternations is fixed, the problem is in EXPSPACE (Chandra, Kozen, and Stockmeyer 1981, Theorem 4.2).

[4] Our algorithm will be allowed to diverge. Such an algorithm can be simulated by one that never diverges and still runs in polynomial space by incrementing a counter at every step, and rejecting the computation-branch if the counter ever exceeds the original number of configurations of the possibly-diverging algorithm.

Algorithm. Given an AFA A with $\Sigma = 2^{AP}$ and a Markov chain $M = (T, t_0, P, AP, lab)$, start by converting the AFA to an equivalent NFA $N = (\Sigma, S, \delta_0, \delta, F)$ — this can be done in exponential time by Lemma 1. We view $\delta_0 \subseteq S$ as a set of states, and assume w.l.o.g. that N rejects the empty word. The algorithm proceeds in rounds, and has two phases: a universal phase followed by an existential phase. It has a variable for a state of M (written q_i in round i) and a variable for a set of states of N (written S_i in round i).

1. In the first phase:
   (a) Initialize q_0 to be t_0 (the initial state of M) and set $S_0 = \delta_0$ (the set of initial states of N).
   (b) At step $i \geq 0$, if $S_i \cap F \neq \emptyset$ then this computation-branch accepts; otherwise:
      i. Universally, do two things: continue with this first phase and switch to the second phase.
      ii. Universally consider all states s_{i+1} of M such that $P(q_i, q_{i+1}) > 0$.
      iii. Let $S_{i+1} \subseteq S$ be the set obtained by applying N's transition function to S_i. That is, let $S_{i+1} = \cup_{s \in S_i} \delta(s, lab(q_i))$.
2. In the second phase, at step i, if $S_i \cap F \neq \emptyset$ then this computation-branch accepts; otherwise:
   (a) Nondeterministically select a state q_{i+1} of M such that $P(q_i, q_{i+1}) > 0$.
   (b) Let S_{i+1} be the set obtained by applying N's transition function to S_i. I.e., define $S_{i+1} = \cup_{s \in S_i} \delta(s, lab(q_i))$.

Correctness. By Corollary 1, $Pr_M([A]_\exists) = 1$ iff for every history x generated by M there is some history y generated by M that is either a prefix of x or an extension of x and such that $lab(y)$ is accepted by A. The first phase of the algorithm universally selects such a history x, and accepts if already some prefix is accepted by A, and the second phase, if reached, existentially selects an extension y and accepts if y is accepted by A.

Complexity. For the space analysis, note that the algorithm only needs to store at most two q_i and two S_i at any point (the current and the next). Moreover, the binary representation of q_i (resp. S_i) is logarithmic (resp. linear) in the size of N. The complexity follows by recalling that the size of N is exponential in the input size. □

We remark that this proof builds the NFA explicitly and simulates the equivalent DFA (formed by the standard subset construction) "on the fly". The reason it cannot simply simulate the NFA is that the universal phase cannot also guess runs of the NFA.

Remark 2. A simple algorithm for deciding $Pr_M([\varphi]_\exists) = 1$ can be obtained by building a DFA A (with final states F) for the LTLf formula φ (Lemma 2), forming the product $M' = A \times M$ and $F' = F \times T$, and then applying the linear-time algorithm for determining if $Pr_{M'}(\Diamond F') = 1$ from (Baier and Katoen 2008, pg 777) where $\Diamond F'$ is the event consisting of plays that visit F'. However, this simple algorithm runs in doubly-exponential time in the size of φ.

5 Probabilistic Verification: Lower Bounds

In this section we prove the lower-bounds in Theorem 1.

5.1 Universal Semantics

We start by showing that the probabilistic verification problem for LTLf under $\forall$-semantics is PSPACE-hard. By Remark 1, this is equivalent to showing that the probabilistic non-emptiness problem for LTLf under $\exists$-semantics is PSPACE-hard. To do this, we reduce from the satisfiability problem for LTLf, which is known to be PSPACE-complete for a fixed set of atoms (De Giacomo and Vardi 2013): given an LTLf formula φ (over a fixed set AP of atomic propositions), we build, in polynomial time, a Markov chain M and a formula ψ such that $Pr_M([\psi]_\exists) > 0$ iff φ is satisfiable.

Define $\psi = \mathsf{X}\varphi$, and define a "complete" Markov chain $M = (T, t_0, P, AP, \mathit{lab})$ where $T = 2^{AP}$; $t_0 \in T$ is arbitrary; for all $s, t \in T$ we have that $P(s,t) = \frac{1}{2^{|AP|}}$; and $\mathit{lab}(s) = s$. Note that every $x \in \mathit{lab}(t_0)(2^{AP})^*$ is generated by M.[5] Clearly, φ is satisfiable iff some word generated by M satisfies $\psi = \mathsf{X}\varphi$, and by Proposition 1, this is iff $Pr_M([\psi]_\exists) > 0$, as required.

5.2 Existential Semantics

We now show that the probabilistic verification problem for LTLf under $\exists$-semantics is EXPSPACE-hard. By Remark 1, this is equivalent to showing that the probabilistic non-emptiness problem for LTLf under $\forall$-semantics is EXPSPACE-hard.

Recall that EXPSPACE is the class of languages accepted by deterministic Turing Machines (TM) with space bound 2^{cn} for some constant $c > 0$. Let T be such a Turing Machine, say with tape alphabet Γ and state set Q. Let x be an input word, say of length n. We will construct, in polynomial time in n, a Markov chain M and an LTLf formula φ, such that the TM T accepts x iff $Pr_M([\varphi]_\forall) > 0$. Recall that for such a reduction one considers the set AP of atoms to be fixed, and the size of the TM as a constant, and only consider the size of the constructed M and φ with respect to the size of the input word x. The reduction we give is a variation of an encoding from (Vardi and Stockmeyer 1985), also found in (Kupferman and Vardi 2000; Bansal et al. 2023). It is, however, longer and more intricate than the reductions in (Vardi and Stockmeyer 1985; Kupferman and Vardi 2000; Bansal et al. 2023) — it has to be carefully crafted to make sure that if T accepts x then $[\varphi]_\forall$ is satisfied not just on at least one infinite trace, but on a set with positive probability. As usual, we assume that the TM halts on every input, in either the halt-reject state or the halt-accept state. Furthermore, in order to avoid an edge-case, we assume w.l.o.g. that the TM does not halt when the head is over the first or the second tape cell (this can be assured by adding two more states to the TM that allow the TM to move right twice before halting).

Define a new alphabet $\Gamma' = \Gamma \cup (\Gamma \times Q) \cup \{0,1\} \cup \{\#, \$\}$. We encode a cell (of the TM's tape) by a word of length $cn+2$ in $\{0,1\}^{cn}(\Gamma \cup (\Gamma \times Q))\{\#, \$\}$: the first cn characters encode in binary (least significant bit first) the position of the cell within the tape (called the *index* of the cell); the next character encodes the contents of the cell (called the *contents* of the cell), including whether or not the TM head is there, and if so, also the current state; and the final character is the cell-delimiter $\#$ if the index of the cell is less than $2^{cn}-1$, and is the configuration-delimiter \$ otherwise. A configuration is encoded by a concatenation of 2^{cn} cell encodings. A run, or a partial run, is encoded by \$ followed by a concatenation of configuration-encodings.

The Markov chain M we construct is the "complete" one which generates every word over Γ' that starts with \$. Formally, $M = (\Gamma', \$, P, \Gamma', \mathit{lab})$ where for all $s, t \in \Gamma'$ we have that $P(s,t) = \frac{1}{|\Gamma'|}$, and $\mathit{lab}(s) = s$.

The formula φ we build will be such that if the TM rejects x then no trace will satisfy $[\varphi]_\forall$, and thus $Pr_M([\varphi]_\forall) = 0$; and if it accepts x then every infinite trace that extends $\tau' v$ — where τ' is the encoding of the (finite) accepting run of the TM on x, and v is the encoding of the final configuration of this run — will satisfy $[\varphi]_\forall$ and thus, $\mathit{cone}(\tau' v) \subseteq [\varphi]_\forall$, and thus $Pr_M([\varphi]_\forall) > 0$.

We now describe φ, mainly in words — translating this description into an LTLf formula is straightforward and standard, e.g., (Kupferman and Vardi 2000; Bansal et al. 2023). The formula φ will be of the form $\varphi_1 \wedge (\varphi_2 \,\mathsf{W}\, \text{halt-state}) \wedge \varphi_3$ where φ_1 says "the first symbol is \$, and if there are at least $n(cn+2)$ more symbols, then the first \$ is followed by an encoding of n cells, with their indices, with x written on them without a TM state, except for the first cell that also has the TM's initial state" conjuncted with "if there are at least two \$'s in the word then all the cells after the first n cells until a \$ encode empty TM cells, i.e., some unknown index followed by the empty-cell symbol followed by a delimiter"; φ_3 says "if a halt-state appears then the first halt-state is halt-accept". Now, φ_2 is a conjunction of the following:

1. "If the current symbol is a delimiter, and there are at least $cn+2$ more symbols, then the next $cn+2$ symbols are of the from $\{0,1\}^{cn}(\Gamma \cup (\Gamma \times Q))\{\#, \$\}$, and if the next cn symbols encode the index 0 or the index 1, then that cell does not contain a halting state (this verifies that the encoding satisfies our assumption that the TM does not halt with the head over one of the first two tape cells)."
2. "If the current symbol is a delimiter, and there is another delimiter in the future, then the current cell ends with \$ if its index is $2^{cn}-1$, and with $\#$ otherwise."
3. "If the current symbol is a delimiter, and if there are two delimiters in the future, then the index of the current cell plus 1 modulo 2^{cn} is the index of the next cell";
4. "If the current symbol is a delimiter, and $3(cn+2)$ symbols before the end of the word there is a delimiter, and there is only one \$ between these two delimiters, and the index of the third cell from the end of the word is equal to the index of the current cell, then the first triple of cell contents constrains the last triple of cell contents according to the TM's transition relation".

We remark that one can expresses that there are at least j

[5] If our Markov chains were allowed to have initial distributions, then we would not need to prefix by a fixed atom as we do here.

more symbols before the end of the word (as needed for example by the first part of φ_2) by the formula $\mathsf{X}^j\,\mathsf{true}$, where X^j is a shorthand for j repetitions of the X operator. One can check equality (as in the last part of φ_2) of an index starting at the current position, with an index starting m positions before the word's end, by the formula:

$$\bigwedge_{i\in[0,cn-1]} (\mathsf{X}^i\,1) \leftrightarrow \mathsf{F}(1 \wedge \mathsf{X}^{m-i}\,last).$$

Here $last = \neg\,\mathsf{X}\,\mathsf{true}$ is a formula that is true exactly at the end of a word.

We now sketch the correctness of this construction. Let τ' be the encoding of the run of the TM on x.

Suppose the TM rejects x. Let τ be any infinite trace. If τ does not start with τ' then, for some prefix of τ, either φ_1 fails (because τ doesn't correctly encode the basic structure of the initial configuration) or $\varphi_2\,\mathsf{W}$ halt-state fails (because τ doesn't encode the run of the TM on x). In particular, our assumption that the TM does not halt with the head over the first or the second cell of the tape (which is verified by the first conjunct of φ_2) assures that φ_2 will have a chance (using prefixes of τ of suitable length) to detect a violation in any of the cells of the configurations up to (and including) the first cell with a halting state (after which the right-hand side of the W holds, effectively disabling the checks in φ_2). If τ does start with τ' then, since the TM rejects, φ_3 fails on the prefix τ' of τ. Thus, $[\varphi]_\forall = \emptyset$.

For the other direction, assume that the TM accepts x. Let τ be a trace that starts with $\tau' v$, where v is the suffix of τ' that encodes the last configuration. It is easy to see that φ_1 and φ_3 hold on every prefix of τ. Observe that all the conjuncts of φ_2 "look" from their start point into the future at most somewhere into the next configuration, but not beyond (i.e., their truth value does not depend on anything beyond the next configuration). Therefore, by including v after τ', we are assured that φ_2 holds on every substring of τ starting anywhere within τ'. Since the last configuration of τ' contains a cell with a halt state, we get that every prefix of τ satisfies $\varphi_2\,\mathsf{W}$ halt-state. We can thus conclude that $cone(\tau' v) \subseteq [\varphi]_\forall$.

6 Probabilistic Synthesis: Upper Bounds

In this section we prove the upper bounds in Theorem 2. Our synthesis algorithms work as follows:

1. Construct a DFA A equivalent to the given LTLf formula φ. This can be achieved — in doubly exponential time in the size of φ — by using Lemma 2 to obtain an equivalent AFA, and then by Lemma 1 to get an equivalent DFA.
2. Construct the product MDP $A \times D$ obtained from D by keeping a copy of the current state of A "on the side".
3. Using simple graph algorithms, for the $\exists$-semantics (resp. $\forall$-semantics) search for a strategy over $A \times D$ such that $Pr_{A\times D,\sigma}(\Diamond F') = 1$ (resp. $Pr_{A\times D,\sigma}(\Box F') = 1$); where $\Diamond F'$ denotes the goal of reaching a final state of A, and $\Box F'$ denotes the goal of never leaving A's final states.

Definition 4. *Fix AP. Given a DFA $A = (2^{AP}, S, \delta_0, \delta, F)$, and an MDP $D = (St, s_0, Act, \Delta, AP, lab)$, define the* product *MDP $A \times D = (St', s_0', Act, \Delta', AP, \_)$ by*

- $St' = S \times St$,
- $s_0' = (\delta(\delta_0, lab(s_0)), s_0)$,
- $\Delta'((q_1, s_1), a, (q_2, s_2))$ *is equal to* $\Delta(s_1, a, s_2)$ *if* $\delta(q_1, lab(s_2)) = q_2$, *and is equal to* 0 *otherwise.*

The product MDP runs both A and D at the same time: from a state (q_1, s_1) and an action a, the second component proceeds probabilistically to a state s_2 exactly as D would, and the first component deterministically updates its state using the transition of A from q_1 on the letter $lab(s_2)$. Note that the labeling function of $A \times D$ is left blank (i.e., $\_$) because it is not used later on.

A history $h = s_0 \cdots s_i$ generated by D *induces* a corresponding history $h' = (q_0, s_0)(q_1, s_1) \cdots (q_i, s_i)$ generated by $A \times D$, where $q_0 = \delta(\delta_0, lab(s_0))$ and $q_{j+1} = \delta(q_j, lab(s_j))$ for every $0 \le j < i$. Conversely, a history h' generated by $A \times D$ induces — simply by projecting its states on their second component — a history h generated by D. Consequently, every strategy σ' over $A \times D$ *induces* a strategy σ over D, and vice-versa, by taking $\sigma(h) = \sigma'(h')$.

Let $F' = F \times St$; given a strategy σ' over $A \times D$, denote by $\Diamond F'$ (resp. $\Box F'$) the set of σ'-plays $t_0 t_1 \cdots$ generated by $A \times D$ for which $t_i \in X$ for some i (resp. for all i). Observe that if σ is the strategy induced by σ', then $Pr_{D,\sigma}([A]_\exists) = Pr_{A\times D,\sigma'}(\Diamond F')$ and $Pr_{D,\sigma}([A]_\forall) = Pr_{A\times D,\sigma'}(\Box F')$. Recall that the DFA A is equivalent to φ and thus, to solve the synthesis problem for $\exists$-semantics (resp. $\forall$-semantics) it remains to show how to decide if there is a strategy σ' such that $Pr_{A\times D,\sigma'}(\Diamond F') = 1$ (resp. $Pr_{A\times D,\sigma'}(\Box F') = 1$). Deciding these two problems can be done, in time polynomial in the size of $A \times D$, by simple graph-theoretic algorithms[6]: in (Baier and Katoen 2008), the algorithm for $\Diamond$ is stated as Algorithm 45 (see also Lemma 10.108), and the algorithm for $\Box$ is easier and not stated explicitly (but sketched after the proof of Lemma 10.111 for the computation of $Pr^{\mathsf{max}}(t \models \Box\neg B) = 1$).

Finally, since the size of $A \times D$ is polynomial in the sizes of A and D, and recalling that A is of size doubly exponential in the LTLf formula φ, we get that the overall complexity of our algorithm for both semantics is 2EXPTIME.

7 Probabilistic Synthesis: Lower Bounds

In this section we establish the lower-bounds in Theorem 2. We first quote a result from the literature (we also offer a new proof below, which does not reduce from 2EXPTIME-Turing machines, as in the cited paper):

Proposition 4. *(Aminof, Giacomo, and Rubin 2020) The probabilistic synthesis problem for LTLf under existential semantics is 2EXPTIME-hard.*

Next, we will establish the 2EXPTIME-hardness of LTLf under universal semantics. In order to do so, we will reduce

[6] Alternatively, consider $\Diamond F'$ and $\Box F'$ as PCTL formulas and use the results in section 10.6.2 of (Baier and Katoen 2008).

from the *sure-winning synthesis problem for LTLf under universal semantics*. The latter is 2EXPTIME-hard via a simple reduction from the *LTLf reactive synthesis problem*, a known 2EXPTIME-hard problem (De Giacomo and Vardi 2015). We first recall that problem. Fix two finite disjoint sets X, Y of atomic propositions, and let $AP = X \cup Y$. An element of 2^X (resp. 2^Y) is called an *environment move* (resp. *agent move*). An *agent strategy* is a function $\sigma : (2^X)^+ \to 2^Y$ and an *environment strategy* is a function $\delta : (2^Y)^* \to 2^X$. For an infinite trace $\tau = (x_0 \cup y_0), (x_1 \cup y_1), \cdots$, we say that σ is consistent with τ if $y_i = \sigma(x_0 \cdots x_i)$ for all $i \geq 0$, and that δ is consistent with τ if $x_0 = \delta(\varepsilon)$ and $x_i = \delta(y_0 \cdots y_{i-1})$ for all $i > 0$. A pair of strategies, one of each kind, *induce* the unique infinite trace τ consistent with both. Intuitively, at each step, the environment moves and then the agent responds. The agent is trying to ensure that every finite prefix of the induced infinite trace satisfies φ, i.e., that τ satisfies $[\varphi]_\forall$, and the environment is trying to ensure the opposite. An agent strategy σ is said to *enforce* $[\varphi]_\forall$ if for all environment strategies δ, the induced trace τ satisfies $[\varphi]_\forall$.

Theorem 3. *(De Giacomo and Vardi 2015) The following problem is 2EXPTIME-hard: given an LTLf formula φ, decide if there is an agent strategy that enforces $[\varphi]_\forall$.*

We now define the sure-winning synthesis problem for LTLf under universal semantics. (We will not need the corresponding problem under existential semantics.) A *game arena* $D = (St, s_0, Act, \Delta, AP, lab)$ is like an MDP except that the transition function is of the form $\Delta : St \times Act \to 2^{St} \setminus \{\emptyset\}$, i.e., with no probabilities. An *LTLf game* G is a pair $(D, [\varphi]_\forall)$ where D is a game arena and φ is an LTLf formula over AP. The *sure-winning synthesis problem for LTLf games under universal semantics* is the following problem: given an LTLf game $G = (D, [\varphi]_\forall)$, to decide whether there exists a strategy $\sigma : Hist \to Act$ such that for every σ-play t we have that $lab(t) \in [\varphi]_\forall$. Such a strategy is said to *sure-win the game* G. We now prove the hardness of sure-winning (we recall that, as is typical, we consider the number of atoms in AP to be a constant when measuring the complexity of reactive synthesis):

Proposition 5. *The sure-winning synthesis problem for LTLf under universal semantics is 2EXPTIME-hard.*

Proof. We reduce from the problem in Theorem 3. Given an LTLf formula ψ over AP, construct an LTLf game $G = (D, [\varphi]_\forall)$ with $\varphi \doteq \tilde{\mathsf{X}}\tilde{\mathsf{X}}\,\psi$ (recall that $\tilde{\mathsf{X}}$ is the weak-next operator), and $D = (St, s_0, Act, \Delta, AP, lab)$ as follows:

- $s_0 = \iota$, here ι is a new symbol;
- $St \doteq \{\iota\} \cup (\{\iota\} \times 2^Y \times 2^X) \cup (2^X \times 2^Y \times 2^X)$;
- $Act \doteq 2^Y$;
- $\Delta(\iota, y') \doteq \{(\iota, y', x') \mid x' \in 2^X\}$ and for $x \in 2^X$ or $x = \iota$, define $\Delta((x, y, x'), y') \doteq \{(x', y', x'') \mid x'' \in 2^X\}$;
- $lab(\iota) \doteq \emptyset$, $lab(\iota, y, x) = \emptyset$, $lab(x, y, z) \doteq x \cup y$;

Intuitively, every state (x, y, x') of the domain that does not mention ι stores in the first two coordinates x, y, the atomic propositions that currently hold (for the benefit of the labeling function), and stores in the third coordinate x' the X portion of AP that will be combined with the next action (i.e., the next Y portion of AP) to become the atomic propositions that hold in the next state. This seeming complexity is purely of a technical nature, and is due to the fact that in game there is an initial state and the player in the game (that wants to satisfy $[\varphi]_\forall$) moves first (by picking actions), while in LTLf reactive synthesis there is no initial state and the agent (that wants to satisfy $[\psi]_\forall$) moves second.

As we now show, there is a correspondence between agent strategies $\sigma : (2^X)^+ \to 2^Y$ and strategies $\sigma' : Hist \to Act$, such that σ enforces $[\psi]_\forall$ iff σ' sure-wins the game $G = (D, [\varphi]_\forall)$. In one direction, an agent strategy σ induces the strategy σ' that first uses a dummy action y, after which it copies σ. Thus, a trace $\pi = (x_0 \cup y_0)(x_1 \cup y_1)(x_2 \cup y_2) \cdots$ consistent with σ induces a σ'-play $\pi' = \iota, (\iota, y, x_0), (x_0, y_0, x_1)(x_1, y_1, x_2) \cdots$. Clearly if π satisfies $[\psi]_\forall$ then π' satisfies $[\varphi]_\forall$ (the weak-next ensures that the prefixes of length 1 and 2 satisfy φ). Moreover, every σ'-play is induced by some trace consistent with σ. Thus if σ enforces $[\psi]_\forall$ then σ' wins the game G. The other direction is similar, i.e., a strategy σ' induces the agent strategy σ that ignores the first action, and afterwards copies σ'. □

We now reduce the problem in Proposition 5 to the synthesis problem for LTLf under each semantics.

Proposition 6. *The probabilistic synthesis problem for LTLf under universal semantics is 2EXPTIME-hard.*

Proof. We reduce from the sure-winning synthesis problem for LTLf under universal semantics (Proposition 5). Let $G = (D, [\varphi]_\forall)$ be an LTLf game. Say D has transition function Δ. Build an MDP D' which is the same as D except that the transition function Δ' has support Δ, e.g., if $\Delta(s, a) = \{s_1, \cdots, s_n\}$ in D then let $\Delta'(s, a)(s_i) = \frac{1}{n}$ in D'. We will show that a strategy $\sigma : Hist \to Act$ sure-wins the game G iff $Pr_{D',\sigma}([\varphi]_\forall) = 1$. The left-to-right direction is immediate. For the other direction, suppose that σ does not sure-win the game. Then, there is some σ-play π that does not satisfy $[\varphi]_\forall$, i.e., there is some finite prefix ρ of π such that $lab(\rho) \not\models \neg\varphi$. It follows that every play in D that extends ρ does not satisfy $[\varphi]_\forall$. Thus every play in $cone(\rho)$ satisfies $\overline{[\varphi]_\forall}$, and thus $Pr_{D',\sigma}(\overline{[\varphi]_\forall}) > 0$, and thus $Pr_{D',\sigma}([\varphi]_\forall) < 1$. □

Alternate Proof of Proposition 4: We reduce from the sure-winning synthesis problem for LTLf under universal semantics (Proposition 5). Let $G = (D, [\varphi]_\forall)$ be an LTLf game, say $D = (St, s_0, Act, \Delta, AP, lab)$. Build an MDP $D' = (St', s_0', Act', \Delta', AP', lab')$, and an LTLf formula φ' over atomic propositions $AP' = AP \cup \{sink, mirror\}$, as follows. Let $St' \doteq \{sink\} \cup (St \times \{0, 1\})$, $Act' \doteq Act$, and $s_0' \doteq (s_0, 0)$. Let $\Delta'(s, a)$ be the uniform distribution over the states $S(s, a)$, defined as follows: $S(s, a) \doteq \{sink\}$ if $s \in \{sink\} \cup (St \times \{1\})$, and $S(s, a) \doteq \Delta(s, a) \times \{0, 1\}$ if $s \in St \times \{0\}$. Let $lab'(sink) \doteq \{sink\}$, $lab'(s, 0) \doteq lab(s)$, and $lab'(s, 1) = lab(s) \cup \{mirror\}$ for every $s \in St$. Let $\varphi' \doteq \varphi \wedge (\mathsf{F}\, mirror) \wedge (\mathsf{G} \neg sink)$.

We do not provide a formal proof here, but give an intuition for the correctness of the construction. The support of

Δ' induces a graph with state set $St \times \{0\}$, and transitions between them make a copy of the domain D. In addition, every transition from $(s, 0)$ to $(s', 0)$ has a mirror transition to $(s', 1)$, and once in $(s', 1)$ the only option is to go to the $sink$ state. The new goal φ' expresses that φ holds, and a mirror state is eventually seen, and the sink is never seen (recall that φ' will be interpreted on finite prefixes of plays). A strategy σ in D may be lifted to a strategy σ' in D' that ignores the second component (if a sink is reached, from that point on it does an arbitrary action). Let π be a σ'-play that reaches a mirror state — this happens with probability 1. Let ρ be the prefix of π that ends in a mirror state. Note that $lab(\rho)$ satisfies φ iff it satisfies φ'. Thus, σ sure-wins the game $(D, [\varphi]_\forall)$ iff $Pr_{D',\sigma'}([\varphi']_\exists) = 1$. The other direction is similar. □

8 Discussion

Perhaps not unexpectedly, the synthesis problem for LTLf specifications (under both semantics) is 2EXPTIME-complete— the same complexity as ordinary LTL synthesis (Vardi 1995). Intuitively, this is because synthesis typically requires determinizing the specification, and determinizing an LTLf formula incurs, in the worst case, a doubly exponential sized blowup (as it is does for ordinary LTL (Kupferman and Vardi 2001)).

The verification problem for LTLf specifications is PSPACE-complete with the universal-prefix semantics, and EXPSPACE-complete for the existential-prefix semantics. While this asymmetry may be surprising at first sight, looking at the non-probabilistic case may provide some intuition. The verification problem for the existential-semantics involves an alternation of quantifiers: "is it the case that for *every* trace there *exists* a finite prefix satisfying the formula"; whereas for the universal-semantics no such an alternation is involved: "is it the case that for *every* trace *every* finite prefix satisfies the formula". We remark that although such an alternation is present also in the synthesis problem, the complexities are dominated by the cost of determinization. actually builds the NFA, which (in the worst case) is exponentially larger than the given LTLf formula, and then simulates the equivalent DFA.

Future Work An extension of this work is to study quantitative probabilistic questions, i.e., to compute the probability of satisfaction in the case of Markov chains, and compute the largest probability in the case of MDPs, as has been done for LTL with infinite-trace semantics, cf. (Baier et al. 2018).

The planning and BPM literature often assume compact representations of Markov chains, e.g., factored probabilistic domains (Sanner 2010), Petri nets (Leemans, Maggi, and Montali 2022). An important avenue would be to find optimal algorithms for the verification and synthesis problems for such compact representations of systems.

Recall from the introduction that there are (at least) two other natural approaches for interpreting LTLf formulas on probabilistic systems (and Markov chains in particular). It would be interesting to establish optimal algorithms for these. Doing this for approach (ii) would be a counterpart to the terminating semantics of (Bansal et al. 2023).

Acknowledgments

We thank the authors of (Bansal et al. 2023) for sending us that paper before it was published.

Work supported in part by NSF grants IIS-1527668, CCF-1704883, IIS-1830549, CNS-2016656, DoD MURI grant N00014-20-1-2787, an award from the Maryland Procurement Office, the ERC Advanced Grant WhiteMech (No.834228).

References

Alman, A.; Maggi, F. M.; Montali, M.; and Peñaloza, R. 2022. Probabilistic declarative process mining. *Information Systems* 109:102033.

Aminof, B.; Kwiatkowska, M.; Maubert, B.; Murano, A.; and Rubin, S. 2019. Probabilistic strategy logic. In *IJCAI*.

Aminof, B.; Giacomo, G. D.; Rubin, S.; and Zuleger, F. 2022. Beyond strong-cyclic: Doing your best in stochastic environments. In *IJCAI*.

Aminof, B.; Giacomo, G. D.; Rubin, S.; and Zuleger, F. 2023. Stochastic best-effort strategies for Borel goals. In *LICS*.

Aminof, B.; Giacomo, G. D.; and Rubin, S. 2020. Stochastic fairness and language-theoretic fairness in planning in nondeterministic domains. In *ICAPS*.

Bacchus, F., and Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artif. Intell.* 116(1-2).

Baier, C., and Katoen, J. 2008. *Principles of model checking*. MIT Press.

Baier, J. A., and McIlraith, S. A. 2006. Planning with first-order temporally extended goals using heuristic search. In *NCAI/IAAI*.

Baier, C.; de Alfaro, L.; Forejt, V.; and Kwiatkowska, M. 2018. Model checking probabilistic systems. In *Handbook of Model Checking*. Springer.

Bansal, S.; Li, Y.; Tabajara, L. M.; Vardi, M. Y.; and Wells, A. 2023. Model checking strategies from synthesis over finite traces. In *ATVA*.

Bianco, A., and de Alfaro, L. 1995. Model checking of probabilistic and nondeterministic systems. In *FSTTCS*.

Bustan, D.; Rubin, S.; and Vardi, M. Y. 2004. Verifying ω-regular properties of markov chains. In *CAV*.

Chandra, A. K.; Kozen, D. C.; and Stockmeyer, L. J. 1981. Alternation. *J. ACM* 28(1):114–133.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artif. Intell.* 147(1-2):35–84.

Clarke, E. M.; Henzinger, T. A.; Veith, H.; and Bloem, R., eds. 2018. *Handbook of Model Checking*. Springer.

Courcoubetis, C., and Yannakakis, M. 1995. The complexity of probabilistic verification. *J. ACM* 42(4):857–907.

De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*.

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

De Giacomo, G.; Masellis, R. D.; Grasso, M.; Maggi, F. M.; and Montali, M. 2014. Monitoring business metaconstraints based on LTL and LDL for finite traces. In *BPM*.

D'Ippolito, N.; Rodríguez, N.; and Sardiña, S. 2018. Fully observable non-deterministic planning as assumption-based reactive synthesis. *J. Artif. Intell. Res.* 61:593–621.

Geatti, L.; Montali, M.; and Rivkin, A. 2024. Foundations of reactive synthesis for declarative process specifications. In *AAAI/EAAI*.

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).

Kupferman, O., and Vardi, M. Y. 2000. An automata-theoretic approach to modular model checking. *TOPLAS* 22(1):87–128.

Kupferman, O., and Vardi, M. Y. 2001. Model checking of safety properties. *FMSD* 19:291–314.

Lahijanian, M.; Wasniewski, J.; Andersson, S. B.; and Belta, C. 2010. Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees. In *ICRA*.

Leemans, S. J. J.; Maggi, F. M.; and Montali, M. 2022. Reasoning on labelled petri nets and their dynamics in a stochastic setting. In *BPM*.

Maggi, F. M.; Montali, M.; and Penaloza, R. 2020. Temporal logics over finite traces with uncertainty. In *AAAI*.

Manna, Z., and Pnueli, A. 1990. A hierarchy of temporal properties (invited paper, 1989). In *PODC*.

Montali, M.; Pesic, M.; van der Aalst, W. M. P.; Chesani, F.; Mello, P.; and Storari, S. 2010. Declarative specification and verification of service choreographies. *ACM Trans. on the Web* 4(1).

Pesic, M., and van der Aalst, W. M. P. 2006. A declarative approach for flexible business processes management. In *Proc. of the BPM 2006 Workshops*.

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Sanner, S. 2010. Relational dynamic influence diagram language (RDDL): Language description. https://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf.

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

Vardi, M. Y., and Stockmeyer, L. J. 1985. Improved upper and lower bounds for modal logics of programs: Preliminary report. In *STOC*.

Vardi, M. Y. 1985. Automatic verification of probabilistic concurrent finite state programs. In *FOCS*.

Vardi, M. Y. 1995. An automata-theoretic approach to linear temporal logic. In *Banff workshop*.

Wells, A. M.; Lahijanian, M.; Kavraki, L. E.; and Vardi, M. Y. 2020. LTLf synthesis on probabilistic systems. In *GandALF*.

Wells, A. M.; Kingston, Z. K.; Lahijanian, M.; Kavraki, L. E.; and Vardi, M. Y. 2021. Finite-horizon synthesis for probabilistic manipulation domains. In *ICRA*.

Yu, P.; Zhu, S.; Giacomo, G. D.; Kwiatkowska, M.; and Vardi, M. Y. 2024. The trembling-hand problem for ltlf planning. In *IJCAI*.