On Frequency LTL in Probabilistic Systems

Vojtěch Forejt[1] and Jan Krčál[2]

1 **Department of Computer Science, University of Oxford, UK**
2 **Saarland University – Computer Science, Saarbrücken, Germany**

Abstract

We study frequency linear-time temporal logic (fLTL) which extends the linear-time temporal logic (LTL) with a path operator $\mathbf{G}^p$ expressing that on a path, certain formula holds with at least a given frequency p, thus relaxing the semantics of the usual $\mathbf{G}$ operator of LTL. Such logic is particularly useful in probabilistic systems, where some undesirable events such as random failures may occur and are acceptable if they are rare enough. Frequency-related extensions of LTL have been previously studied by several authors, where mostly the logic is equipped with an extended "until" and "globally" operator, leading to undecidability of most interesting problems.

For the variant we study, we are able to establish fundamental decidability results. We show that for Markov chains, the problem of computing the probability with which a given fLTL formula holds has the same complexity as the analogous problem for LTL. We also show that for Markov decision processes the problem becomes more delicate, but when restricting the frequency bound p to be 1 and negations not to be outside any $\mathbf{G}^p$ operator, we can compute the maximum probability of satisfying the fLTL formula. This can be again performed with the same time complexity as for the ordinary LTL formulas.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Markov chains, Markov decision processes, LTL, controller synthesis

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2015.184

1 Introduction

Probabilistic verification is a vibrant area of research that aims to formally check properties of stochastic systems. Among the most prominent formalisms, with applications in e.g. modelling of network security protocols [19] or randomised algorithms [17], are Markov chains and Markov decision processes (MDPs). Markov chains are apt for modelling systems that contain purely stochastic behaviour, for example random failures, while MDPs can also express nondeterminism, most commonly present as decisions of a controller or dually as adversarial events in the system.

More technically, MDP is a process that moves in discrete steps within a finite state space (labelled by sets of atomic propositions). Its evolution starts in a given initial state s_0. In each step a *controller* chooses an action a_i from a finite set $A(s_i)$ of actions available in the current state s_i. The next state s_{i+1} is then chosen randomly according to a fixed probability distribution $\Delta(s_i, a_i)$. The controller may base its choice on the previous evolution $s_0 a_0 \dots a_{i-1} s_i$ and may also choose the action randomly. A Markov chain is an MDP where the set $A(s)$ is a singleton for each state s.

For the systems modelled as Markov chains or MDPs, the desired properties such as "whenever a signal arrives to the system, the system eventually switches off" can be often captured by a suitable linear-time logic. The most prominent one in the verification community is Linear Temporal Logic (LTL). Although LTL is suitable in many scenarios, it does not allow to capture some important linear-time properties, for example that a given

26th International Conference on Concurrency Theory (CONCUR 2015).
Editors: Luca Aceto and David de Frutos Escrig; pp. 184–197

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

event takes place *sufficiently often*. The need for such properties becomes even more apparent in stochastic systems, in which probabilities often model random failures. Instead of requiring that no failure ever happens, it is natural to require that failures are infrequent, while still having the power of the LTL to specify these failures using a complex LTL formula.

A natural solution to the above problem is to extend LTL with operators that allow us to talk about *frequencies* of events. Adding such operators can easily lead to undecidability as they often allow one to encode values of potentially infinite counters [6, 7]. In both the above papers this is caused by a variant of a "frequency until" operator that talks about the ratio of the number of given events happening along a finite path. The undecidability results from [6, 7] carry over to the stochastic setting easily, and so, to avoid undecidability, care needs to be taken.

In this paper, we take an approach similar to [21] and in addition to usual operators of LTL such as $\mathbf{X}$, $\mathbf{U}$, $\mathbf{G}$ or $\mathbf{F}$ we only allow *frequency globally* formulae $\mathbf{G}^p\varphi$ that require the formula φ to hold on p-fraction of suffixes of an infinite path, or more formally, $\mathbf{G}^p\varphi$ is true on an infinite path $s_0a_0s_1a_1\ldots$ of an MDP if and only if

$$\liminf_{n\to\infty} \frac{1}{n} \cdot \Big|\{i \mid i < n \text{ and } s_i a_i s_{i+1} a_{i+1} \ldots \text{ satisfies } \varphi\}\Big| \geq p$$

This logic, which we call *frequency LTL (fLTL)*, is still a significant extension to LTL, and because all operators can be nested, it allows to express much larger class of properties (a careful reader will notice that *nesting* of frequency operators is not the main challenge when dealing with fLTL as it can be easily removed for the price of exponential blow-up of the size of the formula).

The problem studied in this paper asks, given a Markov chain and an fLTL formula, to compute the probability with which the formula is satisfied in the Markov chain when starting in the initial state. Analogously, for MDPs we study the *controller synthesis* problem which asks to compute the maximal probability of satisfying the formula, over all controllers.

For an example of possible application, suppose a network service accepts queries by immediately sending back responses, and in addition it needs to be switched off for maintenance during which the queries are not accepted. In most states, a new query comes in the next step with probability 0.5. In the waiting state, the system chooses either to wait further (action w), or to start a maintenance (action m) which takes one step to finish. The service is modelled as an MDP from Figure 1, leaving some parts of the behaviour unspecified. The aim is to synthesise a control strategy that meets with a given probability the requirements on the system. Example requirements can be given by a formula $\mathbf{G}\,\mathbf{F}\,\mathsf{m} \wedge \mathbf{G}\,\mathbf{F}\,(\mathsf{q} \to \mathbf{X}\,\mathsf{r})$ which will require that the service sometimes accepts the request, and sometimes goes for maintenance. However, there is no quantitative restriction on how often the maintenance can take place, and such restriction is inexpressible in LTL. However, in fLTL we can use the formula $\mathbf{G}\,\mathbf{F}\,\mathsf{m} \wedge \mathbf{G}^{0.95}(\mathsf{q} \to \mathbf{X}\,\mathsf{r})$ to restrict that the service is running sufficiently often, or a strong restriction $\mathbf{G}\,\mathbf{F}\,\mathsf{m} \wedge \mathbf{G}^{1}(\mathsf{q} \to \mathbf{X}\,\mathsf{r})$ saying that it is running with frequency 1. The formula may also contain several frequency operators. In order to push the frequency of correctly handled queries towards a bound p, the controller needs to choose to perform the maintenance less and less frequently during operation.

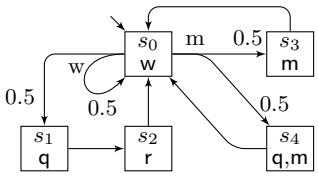

Figure 1 An example MDP.

Related work. Controller synthesis for ordinary LTL is a well-studied problem solvable in time polynomial in the size of the model and doubly exponential in the size of the

formula [2]. Usually, the LTL formula is transformed to an equivalent Rabin automaton, and the probability of reaching certain subgraphs is computed in a product of the MDP (or Markov Chain) with the automaton.

A similar approach is taken by [21]. They study a logic similar to our fLTL, where LTL is extended with a mean-payoff reward constraints in which the reward structures are determined by validity of given subformulas. The authors show that any formula can be converted to a variant of non-deterministic Büchi automata, called multi-threshold mean-payoff Büchi automata, with decidable emptiness problem, thus yielding decidability for model-checking and satisfiability problems of labelled transition systems. Results of [21] cannot be applied to probabilistic systems: here one needs to work with *deterministic automata*, but as pointed out in [21, Section 4, Footnote 4] the approach of [21] heavily relies on non-determinism, since reward values depend on complete future, and so deterministic "multi-threshold mean-payoff Rabin automata" are strictly less expressive than the logic. Another variant of frequency LTL was studied in [6, 7], in which also a modified until operator is introduced. The work [6] maintains boolean semantics of the logic, while in [7] the value of a formula is a number between 0 and 1. Both works obtain undecidability results for their logics, and [6] also yields decidability for restricted nesting. Another logic that speaks about frequencies on a finite interval was introduced in [20] but provides analysis algorithm only for a bounded fragment.

Significant attention has been given to the study of quantitative objectives. The work [5] adds mean-payoff objectives to temporal logics, but only as atomic propositions and not allowing more complex properties to be quantified. The work [3] extends LTL with another form of quantitative operators, allowing accumulated weight constraint expressed using automata, again not allowing quantification over complex formulas. [4] introduces lexicographically ordered mean-payoff objectives in non-stochastic parity games and [9] gives a polynomial time algorithm for almost-sure winning in MDPs with mean-payoff and parity objectives. These objectives do not allow to attach mean-payoff (i.e. frequencies) to properties more complex than atomic propositions. The solution to the problem requires infinite-memory strategy which at high level has a form similar to the form of strategies we construct for MDPs. Similar strategies also occur in [11, 10, 8] although each of these works deals with a fundamentally different problem.

In branching-time logics, CSL is sometimes equipped with a "steady-state" operator whose semantics is similar to our $\mathbf{G}^p$ (see e.g. [1]), and an analogous approach has been taken for the logic PCTL [16, 13]. In such logics every temporal subformula is evaluated over states, and thus the model-checking of a frequency operator can be directly reduced to achieving a single mean-payoff reward. This is contrasted with our setting in which the whole formula is evaluated over a single path, giving rise to much more complex behaviour.

Our contributions. To our best knowledge, this paper gives the first decidability results for probabilistic verification against linear-time temporal logics extended by frequency operators with *complex nested subformulas* of the logic.

We first give an algorithm for computing the probability of satisfying an fLTL formula in a Markov Chain. The algorithm works by breaking the fLTL formula into linearly many ordinary LTL formulas, and then off-the-shelf verification algorithms can be applied. We obtain that the complexity of fLTL model-checking is the same as the complexity of LTL model checking. Although the algorithm itself is very simple, some care needs to be taken when proving its correctness: as we explain later, the "obvious" proof approach would fail since some common assumptions on independence of events are not satisfied.

We then proceed with Markov decision processes, where we show that the controller synthesis problem is significantly more complex. Unlike the ordinary LTL, for fLTL the

controller-synthesis problem may require strategies to use *infinite memory*, even for very simple formulas. On the positive side, we give an algorithm for synthesis of strategies for formulas in which the negations are pushed to atomic propositions, and all the frequency operators have lower bound 1. Although this might appear to be a simple problem, it is not easily reducible to the problem for LTL, and the proof of the correctness of the algorithm is in fact very involved. This is partly because even if a strategy satisfies the formula, it can exhibit a very "insensible" behaviour, as long as this behaviour has zero frequency in the limit. In the proof, we need to identify these cases and eliminate them. Ultimately, our construction again yields the same complexity as the problem for ordinary LTL. We believe the contribution of the fragment is both practical, as it gives a "weaker" alternative of the $\mathbf{G}$ operator usable in controller synthesis, and theoretical, giving new insights into many of the challenges one will face in solving the controller-synthesis problem for the whole fLTL.

2 Preliminaries

We now proceed with introducing basic notions we use throughout this paper.

A *probability distribution* over a finite or countable set X is a function $d : X \to [0, 1]$ such that $\sum_{x \in X} d(x) = 1$, and $\mathcal{D}(X)$ denotes the set of all probability distributions over X.

Markov decision processes and Markov chains. A *Markov decision process* (MDP) is a tuple $\mathcal{M} = (S, A, \Delta)$ where S is a finite set of states, A is a finite set of actions, and $\Delta : S \times A \to \mathcal{D}(S)$ is a partial probabilistic transition function. A *Markov chain* (MC) is an MDP in which for every $s \in S$ there is exactly one a with $\Delta(s, a)$ being defined. We omit actions completely when we speak about Markov chains and no confusion can arise.

An infinite *path*, also called *run*, in $\mathcal{M}$ is a sequence $\omega = s_0 a_0 s_1 a_1 \cdots$ of states and actions such that $\Delta(s_i, a_i)(s_{i+1}) > 0$ for all i, and we denote by $\omega(i)$ the suffix $s_i a_i s_{i+1} a_{i+1} \cdots$. A finite path h, also called *history*, is a prefix of an infinite path ending in a state. Given a finite path $h = s_0 a_0 s_1 a_1 \cdots s_i$ and a finite or infinite path $h' = s_i a_i s_{i+1} a_{i+1} \cdots$ we use $h \cdot h'$ to denote the concatenated path $s_0 a_0 s_1 a_1 \cdots$. The set of paths starting with a prefix h is denoted by $\mathit{Cyl}(h)$, or simply by h if it leads to no confusion. We overload the notation also for sets of histories, we simply use H instead of $\bigcup_{h \in H} \mathit{Cyl}(h)$.

A *strategy* is a function σ that to every finite path h assigns a probability distribution over actions such that if an action a is assigned a non-zero probability, then $\Delta(s, a)$ is defined where s denotes the last state in h. A strategy σ is *deterministic* if it assigns Dirac distribution to any history, and *randomised* otherwise. Further, it is *memoryless* if its choice only depends on the last state of the history, and *finite-memory* if there is a finite automaton such that σ only makes its choice based on the state the automaton ends in after reading the history.

An MDP $\mathcal{M}$, a strategy σ and an initial state s_{in} give rise to a probability space $\mathbb{P}^{s_{in}}_{\sigma}$ defined in a standard way [15]. For a history h and a measurable set of runs U starting from the last state of h, we denote by $\mathbb{P}^{h}_{\sigma}(U)$ the probability $\mathbb{P}^{s_{in}}_{\sigma}(\{h \cdot \omega \mid \omega \in U\} \mid h)$. Similarly, for a random variable X we denote by $\mathbb{E}^{s_{in}}_{\sigma}(X)$ the expectation of X in this probability space and by $\mathbb{E}^{h}_{\sigma}(X)$ the expectation $\mathbb{E}^{s_{in}}_{\sigma}(X_h \mid h)$. Here, X_h is defined by $X_h(h \cdot \omega) = X(\omega)$ for runs of the form $h \cdot \omega$, and by $X_h(\omega') = 0$ for all other runs. We say that a property holds *almost surely* (or for almost all runs, or almost every run) if the probability of the runs satisfying the property is 1.

Frequency LTL. The syntax of *frequency LTL (fLTL)* is defined by the equation:

$$\varphi ::= \alpha \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi \mid \mathbf{G}^{p}\varphi$$

where α ranges over a set AP of atomic propositions. The logic LTL is obtained by omitting the rule for $\mathbf{G}^{p}\varphi$. For Markov chains we study the whole fLTL whereas for MDP, we restrict to a fragment that we call *1-fLTL*. In this fragment, negations only occur immediately preceding atomic propositions, and $\mathbf{G}^{p}$ operators occur only with $p = 1$.

For an infinite sequence $\gamma = x_1 x_2 \ldots$ of numbers, we set $\mathsf{freq}(\gamma) := \liminf_{i\to\infty} \frac{1}{i}\sum_{j=1}^{i} x_i$. Given a valuation $\nu : S \to 2^{AP}$, the semantics of fLTL is defined over a path $\omega = s_0 a_0 s_1 \ldots$ of an MDP as follows.

$\omega \models \alpha$	iff $\alpha \in \nu(s_0)$	$\omega \models \mathbf{X}\,\varphi$	iff $\omega(1) \models \varphi$
$\omega \models \neg\varphi$	iff $\omega \not\models \varphi$	$\omega \models \varphi_1\,\mathbf{U}\,\varphi_2$	iff $\exists k : \omega(k) \models \varphi_2 \wedge \forall \ell{<}k : \omega(\ell) \models \varphi_1$
$\omega \models \varphi_1 \vee \varphi_2$	iff $\omega{\models}\varphi_1$ or $\omega{\models}\varphi_2$	$\omega \models \mathbf{G}^{p}\varphi$	iff $\mathsf{freq}(\mathbf{1}_{\varphi,0}\mathbf{1}_{\varphi,1}\ldots) \geq p$

where $\mathbf{1}_{\varphi,i}$ is 1 for ω iff $\omega(i) \models \varphi$, and 0 otherwise. We define $\mathsf{true}, \mathsf{false}, \wedge$, and $\rightarrow$ by their usual definitions and introduce standard operators $\mathbf{F}$ and $\mathbf{G}$ by putting $\mathbf{F}\,\varphi \equiv \mathsf{true}\,\mathbf{U}\,\varphi$ and $\mathbf{G}\,\varphi \equiv \neg\mathbf{F}\,\neg\varphi$. Finally, we use $\mathbb{P}_\sigma(\varphi)$ as a shorthand for $\mathbb{P}_\sigma(\{\omega \mid \omega \models \varphi\})$.

▶ **Definition 1** (Controller synthesis)**.** The controller synthesis problem asks to decide, given an MDP $\mathcal{M}$, a valuation ν, an initial state s_{in}, an fLTL formula φ and a probability bound x, whether $\mathbb{P}^{s_{in}}_{\sigma}(\varphi) \geq x$ for some strategy σ.

As an alternative to the above problem, we can ask to compute the maximal possible x for which the answer is true. In the case of Markov chains, we speak about Satisfaction problem since there is no strategy to synthesise.

Rabin automata. A *(deterministic) Rabin automaton* is a tuple $R = (Q, q_{in}, \Sigma, \delta, \mathcal{F})$ where Q is a finite set of states, Σ is an input alphabet, $\delta : Q \times \Sigma \to Q$ is a transition function, and $\mathcal{F} \subseteq Q \times Q$ is an accepting condition. A computation of R on an infinite word $\varrho = a_0 a_1 \ldots$ over the alphabet Σ is the infinite sequence $R[\varrho] = q_0 q_1 \ldots$ with $q_0 = q_{in}$ and $\delta(q_i, a_i) = q_{i+1}$. A computation is accepting (or "R accepts ϱ") if there is $(E, F) \in \mathcal{F}$ such that all states of E occur only finitely many times in the computation, and some state of F occurs in it infinitely many times. For a run $\omega = s_0 a_0 s_1 a_1 \ldots$ and a valuation ν, we use $\nu(\omega)$ for the sequence $\nu(s_0)\nu(s_1)\ldots$ of sets of atomic propositions.

As a well known result [2], for every MDP $\mathcal{M}$, valuation ν and an LTL formula φ there is a Rabin automaton R over the alphabet 2^{AP} such that R is constructible in doubly exponential time and $\omega \models \varphi$ iff R accepts $\nu(\omega)$. We say that R is equivalent to φ. It is not clear whether this result and the definition of Rabin automata can be extended to work with fLTL in a way that would be useful for our goals. The reason for this is, as pointed out in [21, Section 4, Footnote 4], that the frequencies in fLTL depend on the future of a run, and so require non-determinism, which is undesirable in stochastic verification.

3 Satisfaction problem for Markov Chains

In this section we show how to solve the satisfaction problem for MCs and fLTL. Let us fix a MC $\mathcal{M} = (S, \Delta)$, an initial state s_{in} and fLTL formula ψ. We will use the notion of *bottom strongly connected component* (bscc) of $\mathcal{M}$, which is a set of states S' such that for all $s \in S'$ the set of states reachable from s is exactly S'. If s is in a bscc, by $\mathsf{bscc}(s)$ we denote the bscc containing s.

We first describe the algorithm computing the probability of satisfying ψ from s_{in}, and then prove its correctness.

The algorithm. The algorithm proceeds in the following steps. First, for each state contained in some bscc B, we compute the steady-state frequency x_s of s within B. It is the number $\mathbb{E}^s(\mathsf{freq}(\mathbf{1}_{s,0}\mathbf{1}_{s,1}\ldots))$ where $\mathbf{1}_{s,i}(\omega)$ equals 1 if the ith state of ω is s, and 0, otherwise. Afterwards, we repeat the following steps and keep modifying ψ for as long as it contains any $\mathbf{G}^p$ operators:

1. Let φ be a LTL formula and p a number such that ψ contains $\mathbf{G}^p\varphi$.
2. Compute $\mathbb{P}^s(\varphi)$ for every state s contained in some bscc.
3. Create a fresh atomic proposition $\alpha_{\varphi,p}$ which is true in a state s iff s is contained in a bscc and $\sum_{t\in\mathsf{bscc}(s)} x_t \cdot \mathbb{P}^t(\varphi) \geq p$.
4. Modify ψ by replacing any occurrence of $\mathbf{G}^p\varphi$ with $\mathbf{F}\,\alpha_{\varphi,p}$.

Once ψ contains no $\mathbf{G}^p$ operators, it is an LTL formula and we can use off-the-shelf techniques to compute $\mathbb{P}^{s_{in}}(\psi)$, which is our desired value.

Correctness. The correctness of the algorithm relies on the fact that $\alpha_{\varphi,p}$ labels states in a bscc B if almost every run reaching B satisfies the corresponding frequency constraint:

▶ **Proposition 2.** *For every LTL formula φ, every number p, every bscc B and almost every run ω that enters B we have $\omega \models \mathbf{G}^p\varphi$ if and only if $\sum_{t\in B} x_t \cdot \mathbb{P}^t(\varphi) \geq p$.*

The proposition might seem "obviously true", but the proof is not trivial. The main obstacle is that satisfactions of φ on $\omega(i)$ and $\omega(j)$ are *not independent* events in general: for example if $\varphi \equiv \mathbf{F}\,\alpha$ and $i < j$, then $\omega(j) \models \varphi$ implies $\omega(i) \models \varphi$. Hence we cannot apply the Strong law of large numbers (SLLN) for independent random variables or Ergodic theorem for Markov chains [18, Theorems 1.10.1-2], which would otherwise be obvious candidates. Nevertheless, we can use the following variant of SLLN for correlated events.

▶ **Lemma 3.** *Let $Y_0, Y_1 \ldots$ be a sequence of random variables which only take values 0 or 1 and have expectation μ. Assume there are $0 < r, c < 1$ such that for all $i, j \in \mathbb{N}$ we have $\mathbb{E}((Y_i - \mu)(Y_j - \mu)) \leq r^{\lfloor c|i-j|\rfloor}$. Then $\lim_{n\to\infty} \sum_{i=0}^{n} Y_i/n = \mu$ almost surely.*

Using the above lemma, we now prove Proposition 2 for fixed φ, B, p. Let R denote the Rabin automaton equivalent to φ and $\mathcal{M} \times R$ be the Markov chain product of $\mathcal{M}$ and R.

First, we say that a finite path $s_0 \ldots s_k$ of $\mathcal{M}$ is *determined* if the state q_k reached by R after reading $\nu(s_0 \ldots s_{k-1})$ satisfies that (s_k, q_k) is in a bscc of $\mathcal{M} \times R$. We point out that for a determined path $s_0 \ldots s_k$, either almost every run of $Cyl(s_0 \ldots s_k)$ satisfies φ, or almost no run of $Cyl(s_0 \ldots s_k)$ satisfies φ. Also, the probability of runs determined within k steps is at least $\sum_{i=0}^{\lfloor k/M\rfloor}(1-r^M)^i r^M = 1 - (1-r^M)^{\lfloor k/M\rfloor}$ where M is the number of states of $\mathcal{M} \times R$ and r is the minimum probability that occurs in $\mathcal{M} \times R$.

Now fix a state $s \in B$. For all $t \in B$ and $i \geq 0$ we define random variables X_i^t over runs initiated in s. We let $X_i^t(\omega)$ take value 1 if t is visited at least i times in ω and the suffix of ω starting from the ith visit to t satisfies φ. Otherwise, we let $X_i^t(\omega) = 0$. Note that all X_i^t have a common expected value $\mu_t = \mathbb{P}^t(\varphi)$.

Next, let i and j be two numbers with $i \leq j$. We denote by Ω the set of all runs and by D the set of runs ω for which the suffixes starting from the ith visit to t are determined before the jth visit to t (note that D can possibly be $\emptyset$). Because on these determined runs $\mathbb{E}^s(X_j^t - \mu_t \mid D) = 0$, we get

$$\mathbb{E}^s((X_i^t - \mu_t)(X_j^t - \mu_t)) \leq 1 - \mathbb{P}^s(D) \leq (1 - r^M)^{\lfloor (i-j)/M\rfloor}$$

as shown in [14]. Thus, Lemma 3 applies to the random variables X_i^t for a fixed t. Considering all $t \in B$ together, we show in [14] that $\mathsf{freq}(\mathbf{1}_{\varphi,0}\mathbf{1}_{\varphi,1}\ldots) = \sum_{t\in\mathsf{bscc}(s)} x_t \mathbb{P}^s(\varphi)$ for almost all runs initiated in the state s we fixed above. Because almost all runs that enter B eventually visit s, and because satisfaction of $\mathbf{G}^p\varphi$ is independent of any prefix, the proof of Proposition 2 is finished, and we can establish the following.

▶ **Theorem 4.** *The satisfaction problem for Markov chains and fLTL is solvable in time polynomial in the size of the model, and doubly exponential in the size of the formula.*

4 Controller synthesis for MDPs

We now proceed with the controller synthesis problem for MDPs and 1-fLTL. The problem for this restricted fragment of 1-fLTL is still highly non-trivial. In particular, it is not equivalent to synthesis for the *LTL* formula where every $\mathbf{G}^1$ is replaced with $\mathbf{G}$. Indeed, for satisfying any LTL formula, finite memory is sufficient, while for 1-fLTL, the following theorem shows that infinite memory may be necessary.

▶ **Theorem 5.** *There is a 1-fLTL formula ψ and a Markov decision process $\mathcal{M}$ with valuation ν such that the answer to the controller synthesis problem is "yes", but there is no finite-memory strategy witnessing this.*

Proof idea. Consider the MDP from Figure 1 together with the formula $\psi = \mathbf{G}\,\mathbf{F}\,\mathsf{m} \wedge \mathbf{G}^1(\mathsf{q} \to \mathbf{X}\,\mathsf{r})$. Independent of the strategy being used, no run initiated in s_4 satisfies the subformula $\mathsf{q} \to \mathbf{X}\,\mathsf{r}$, while every run initiated in any other state satisfies this subformula. This means that we need the frequency of visiting s_4 to be 0. The only finite-memory strategies achieving this are those that from some history on never choose to go right in the controllable state. However, under such strategies the formula $\mathbf{G}\,\mathbf{F}\,\mathsf{m}$ is not almost surely satisfied. On the other hand, the infinite-memory strategy that on i-th visit to s_0 picks m if and only if i is of the form 2^j for some j satisfies ψ.

Note that although the above formula requires infinite memory due to "conflicting" conjuncts, infinite memory is needed already for simpler formulae of the form $\mathbf{G}^1(\mathsf{a}\,\mathbf{U}\,\mathsf{b})$. ◀

The above result suggests that it is not possible to easily re-use verification algorithms for ordinary LTL. Nevertheless, our results allow us to establish the following theorem.

▶ **Theorem 6.** *The controller-synthesis problem for MDPs and 1-fLTL is solvable in time polynomial in the size of the model and doubly exponential in the size of the formula.*

For the rest of this section, in which we prove Theorem 6, we fix an MDP $\mathcal{M}$, valuation ν, an initial state s_{in}, and a 1-fLTL formula ψ. The proof is given in two parts. In the first part, in Section 4.1 we show that the controller-synthesis problem is equivalent to problems of reaching a certain set Υ and then "almost surely winning" from this set. To prove this, the "almost surely winning" property will further be reduced to finding certain set of states and actions on a product MDP (Lemma 13). In the second part of the proof, given in Section 4.2, we will show that all the above sets can be computed.

4.1 Properties of satisfying strategies

Without loss of generality suppose that the formula ψ does not contain $\mathbf{G}^1$ as the outermost operator, and that it contains n subformulas of the form $\mathbf{G}^1\varphi$. Denote these subformulas $\mathbf{G}^1\varphi_1, \ldots \mathbf{G}^1\varphi_n$. For example, $\psi = \mathsf{i} \to \big(\mathbf{G}\,(\mathsf{q} \to \mathsf{a}) \wedge \mathbf{G}^1(\mathsf{p}_1\,\mathbf{U}\,\mathsf{r} \vee \mathbf{G}^1\mathsf{a})\big)$ contains $\varphi_1 = \mathsf{p}_1\,\mathbf{U}\,\mathsf{r} \vee \mathbf{G}^1\mathsf{a}$ and $\varphi_2 = \mathsf{a}$.

The first step of our construction is to convert these formulae $\psi, \varphi_1, \ldots, \varphi_n$ to equivalent Rabin automata. However, as the formulae contain $\mathbf{G}^1$ operators, they cannot be directly expressed using Rabin automata (and as pointed out by [21], there is a fundamental obstacle preventing us from extending Rabin automata to capture $\mathbf{G}^p$).

To overcome this, we replace all occurrences of $\mathbf{G}^1\varphi_i$ in such formulae by either true or false, to capture that the frequency constraint is or is not satisfied on a run. Such a replacement can be fixed only after a point in the execution is reached where it becomes clear which frequency constraints in ψ can be satisfied. For a formula $\xi \in \{\psi, \varphi_1, \ldots \varphi_n\}$, any subset $I \subseteq \{1, \ldots, n\}$ of satisfied constraints defines a LTL formula ξ^I obtained from ξ by replacing all subformulas $\mathbf{G}^1\varphi_i$ (not contained in any other $\mathbf{G}^1$) with true if $i \in I$ and with false if $i \notin I$. The Rabin automaton for ξ^I is then denoted by $R_{\xi,I}$. For the formula ψ above, we have, e.g., $\psi^{\{1\}} = \psi^{\{1,2\}} = \mathsf{i} \to \big(\mathbf{G}\,(\mathsf{q} \to \mathsf{a}) \wedge \mathsf{true}\big)$, and $\varphi_1^{\emptyset} = \mathsf{p_1}\,\mathbf{U}\,\mathsf{r} \vee \mathsf{false}$.

We use Q for a disjoint union of the state spaces of these distinct Rabin automata, and Q_ψ for a disjoint union of the state spaces of the automata $R_{\psi,I}$, called *main* automata, for all I. Finally, for $q \in Q$ belonging to a Rabin automaton R we denote by R^q the automaton obtained from R by changing its initial state to q.

Let us fix a state s of $\mathcal{M}$ and a state q of $R_{\psi,I}$ for some $I \subseteq \{1, \ldots, n\}$. We say that a run $s_0 a_0 s_1 a_1 \ldots$ *reaches* (s, q) if for some k we have $s = s_k$ and q is the state reached by the main automaton $R_{\psi,I}$ after reading $\nu(s_0 a_0 s_1 \ldots s_{k-1})$. Once (s, q) is reached, we say that a strategy σ' is *almost-surely winning* from (s, q) if $\mathbb{P}^s_{\sigma'}$ assigns probability 1 to the set of runs ω such that $\nu(\omega)$ is accepted by $R^q_{\psi,I}$, and $\omega \models G^1\varphi_i$ whenever[1] we have $i \in I$.

▶ **Proposition 7.** *There is a strategy σ such that $\mathbb{P}_\sigma(\psi) = x$ if and only if there is a set $\Upsilon \subseteq S \times Q_\psi$ for which the following two conditions are satisfied:*

1. *There is a strategy σ' such that $\mathbb{P}_{\sigma'}(\{\omega \mid \omega \text{ reaches a pair from } \Upsilon\}) = x$.*
2. *For any $(s, q) \in \Upsilon$ there is $\sigma_{s,q}$ almost-surely winning from (s, q).*

Intuitively, the proposition relies on the fact that if $\mathbf{G}^1\varphi_i$ holds on a run, then it holds on all its suffixes, and says that any strategy σ can be altered so that almost surely there will be a prefix after which we know which of the $\mathbf{G}^1\varphi_i$ will be satisfied.

▶ **Example 8.** Let us first illustrate the set Υ on a formula $\mathbf{X}\,q \wedge \mathbf{G}\,\mathbf{F}\,\mathsf{m} \wedge \mathbf{G}^1(\mathsf{q} \to \mathbf{X}\,\mathsf{r})$ that can be satisfied on the MDP from Figure 1 with probability 0.5. Figure 2 shows Rabin automata for the formulae $\psi^{\{1\}} = \mathbf{X}\,\mathsf{q} \wedge \mathbf{G}\,\mathbf{F}\,\mathsf{m} \wedge \mathsf{true}$ (left) and $\varphi_1^{\{1\}} = \mathsf{q} \to \mathbf{X}\,\mathsf{r}$. In this simple example, the "decision" whether the formula will be satisfied (and which $\mathbf{G}^1$ subformulas will be satisfied) comes after the first step. Thus, we can set $\Upsilon = \{(s_1, q_1)\}$.

We now prove Proposition 7. The direction $\Leftarrow$ is straightforward. It suffices to define σ so that it behaves as σ' initially until it reaches some $(s, q) \in \Upsilon$ for the first time; then it behaves as $\sigma_{s,q}$.

Significantly more difficult is the direction $\Rightarrow$ of Proposition 7 that we address now. We fix a strategy σ with $\mathbb{P}_\sigma(\psi) = x$. The proof is split into three steps. We first show how to identify the set Υ, and then we show that items 1 and 2 of Proposition 7 are satisfied. The last part of the proof requires most of the effort. In the proof, we

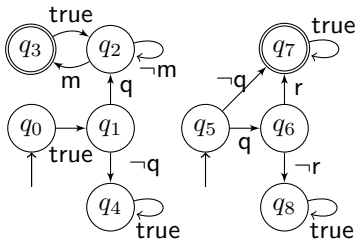

Figure 2 Example Rabin aut.

[1] Note that the product construction that we later introduce does not give us "iff" here. This is also why we require the negations to only occur in front of atomic propositions

will need to eliminate some unlikely events, and for this we will require that their probability is small to start with. For this purpose, we fix a *very small* positive number λ; to avoid cluttering of notation, we do not give a precise value of λ, but instead point out that it needs to be chosen such that any numbers that depend on it in the following text have the required properties (i.e. are sufficiently small or big; note that such choice is indeed possible). We should stress that λ is influencing *neither* the size of representation of our strategy *nor* the complexity of our algorithm.

Identifying the set Υ

In the first step, we identify an appropriate set Υ. Intuitively, we put into Υ positions of the runs satisfying ψ where *the way* ψ *is satisfied* is (nearly) decided, i.e. where it is (nearly) clear which frequency constraints will be satisfied by σ in the future. To this end, we mark every run ω satisfying ψ with a set $I_\omega \subseteq \{1, \dots, n\}$ such that $i \in I_\omega$ iff the formula $\mathbf{G}^1\varphi_i$ holds on the run. We then define a set of finite paths Γ to contain all paths h for which there is $I_h \subseteq \{1, \dots, n\}$ such that exactly the frequency constraints from I_h as well as ψ^{I_h} are satisfied on (nearly) all runs starting with h. Precisely, such that $\mathbb{P}_\sigma(\{\omega' \mid \omega' \models \psi^{I_h} \wedge I_{\omega'} = I_h\} \mid h) \geq 1 - \lambda$. Finally, for every $h \in \Gamma$ we add to Υ the pair (s, q) where $h = h's$ and q is the state in which R_{ψ,I_h} ends after reading $\nu(h')$.

Reaching Υ

It suffices to show that the strategy σ itself satisfies $\mathbb{P}_\sigma(\Gamma) = x$. We will use the following variant of Lévy's Zero-One Law, a surprisingly powerful formalization of the intuitive fact that "things need to get (nearly) decided, eventually".

▶ **Lemma 9** (Lévy's Zero-One Law [12]). *Let σ be a strategy and X a measurable set of runs. Then for* almost *every run ω we have $\lim_{n\to\infty} \mathbb{P}_\sigma(X \mid h_n) = \mathbf{1}_X(\omega)$ where each h_n denotes the prefix of ω with n states and the function $\mathbf{1}_X$ assigns 1 to $\omega \in X$ and 0 to $\omega \notin X$.*

For every $I \subseteq \{1, \dots, n\}$ we define $X_I = \{\omega' \mid \omega' \models \psi^I \wedge I_{\omega'} = I\}$ to be the set of runs that are marked by I and satisfy the formula ψ^I. Then by Lemma 9, for almost every run ω that satisfies ψ and has $I_\omega = I$, there must be a prefix h of the run for which $\mathbb{P}_\sigma(X_I \mid h) \geq 1 - \lambda$ because $\omega \in X_I$. Any such prefix was added to Γ, with $I_h = I$.

Almost-surely winning from Υ

For the third step of the proof of direction $\Rightarrow$ of Proposition 7 we fix $(s^\star, q^\star) \in \Upsilon$ and we construct a strategy $\sigma_{s^\star,q^\star}$ that is almost-surely winning from $(s^\star, q^\star)$. Furthermore, let $I^\star \subseteq \{1, \dots, n\}$ denote the set such that $q^\star$ is a state from $R_{\psi,I^\star}$. As we have shown in Theorem 5, strategies might require infinite memory, and this needs to be taken into consideration when constructing $\sigma_{s^\star,q^\star}$. The strategy cycles through two "phases", called *accumulating* and *reaching* that we illustrate on our example.

▶ **Example 10.** Returning to Example 8, we fix $(s^\star, q^\star) = (s_1, q_1)$ and $I^\star = \{1\}$, with the corresponding history from Γ being $s_0 w s_1$. The strategy σ_{s_1,q_1} we would like to obtain

- first "accumulates" arbitrarily many steps from which all $\varphi_1^{\{1\}}$ can be almost surely satisfied. I.e., it accumulates arbitrarily many newly started instances of the Rabin automaton $R_{\varphi_1,\{1\}}$ (all being in state q_5) by repeating action w in s_0.
- Then it "reaches" with all the Rabin automata $R_{\psi,\{1\}}$ and $R_{\varphi_1,\{1\}}$ accumulated in the previous phase their accepting states q_3 and q_7 respectively. For $R_{\varphi_1,\{1\}}$ this happens

without any intervention of the strategy, but for $R_{\psi,\{1\}}$ the strategy needs to take the action m. Then after returning to s_0 it comes back to a state where the next accumulating phase starts. Thus, we need to make sure we make the accumulating phases progressively longer so that in the long run they take place with frequency 1.

The proof that such a simple behaviour suffices is highly non-trivial. To illustrate this, let us extend the MDP from Figure 1 with an action $\mathsf{decline}$ with $\Delta(s_1, \mathsf{decline}) = s_0$. The strategy σ from the proof of Theorem 5 satisfies $\mathbb{P}_\sigma(\psi) = 1$ for $\psi = \mathbf{G}\,\mathbf{F}\,\mathsf{m} \wedge \mathbf{G}^1(\mathsf{q} \to \mathbf{X}\,\mathsf{r})$. However, we can modify σ and obtain a "weird" strategy σ' that takes the action $\mathsf{decline}$ in the i-th visit to s_1 with probability $1/2^i$. Such a strategy (a) still satisfies $\mathbb{P}_{\sigma'}(\psi) = 1/2$ but (b) it does not guarantee almost sure satisfaction of $\varphi_1^{\{1\}}$ in s_1. Thus, it does not accumulate in the sense explained above. We will show that any such weird strategy can be slightly altered to fit into our scheme. ◀

To show that alternation between such accumulating and reaching suffices (and to make a step towards the algorithm to construct such $\sigma_{s^\star,q^\star}$), we introduce a tailor-made product construction $\mathcal{M}_\otimes$. The product keeps track of a *collection* of arbitrarily many Rabin automata accumulated up to now. We need to make sure that almost all runs of all automata in the collection are accepting, and we will do this by ensuring that: (i) almost every computation of all Rabin automata eventually commits to an accepting condition (E, F), and (ii) from the point the automaton "commits" to the accepting condition, no more elements of E are visited and (iii) some element of F is visited infinitely often. To ensure this, we store additional information along any state $q \in Q$ of each automaton:

- $(q, \star)$ is a new instance that has to commit to an accepting condition soon;
- $(q, (E, F)_\circ)$ is an instance that has to visit a state of F soon;
- $(q, (E, F)_\bullet)$ is an instance that recently fulfilled the accepting condition by visiting F;
- $(q, \bot)$ is an instance that violated the accepting condition it had committed to.

Let $\mathcal{C}$ denote the set of these pairs for all $q \in Q$ and all accepting conditions (E, F) of the Rabin automaton where the state q belongs. Note that $\mathcal{C}$ is finite; because we need to encode unbounded number of instances of Rabin automata along the run, each element of a collection $C \subseteq \mathcal{C}$ might stand for multiple instances that are in exactly the same configuration. We say that $C \subseteq \mathcal{C}$ is *fulfilled* if it contains only elements of the form $(q, (E, F)_\bullet)$. The aim is to fulfil the collection infinitely often, the precise meaning of "recently" and "soon" above is then "since the last fulfilled state" and "before the next fulfilled state".

Using the product $\mathcal{M}_\otimes$, we show that if there is a satisfying strategy in $\mathcal{M}$, there is a strategy in $\mathcal{M}_\otimes$ with a simple structure that visits a fulfilled state infinitely often (in Lemma 13). Due to the simple structure, such a strategy can be found algorithmically. Finally, we show that such a strategy in the product induces a satisfying strategy in $\mathcal{M}$ (in Lemma 12) yielding correctness of the algorithm.

The product. Let $\mathcal{M}_\otimes$ be an MDP with states $S_\otimes = S \times 2^{\mathcal{C}}$, actions $A_\otimes = A \times 2^{\mathcal{C}}$, and transition function $\Delta_\otimes$ defined as follows. We first define possible choices of a strategy in $\mathcal{M}_\otimes$. Given a state (s, C_s), we say that an action (a, C_a) is *legal* in (s, C_s) if a is a valid choice in s in the original MDP, i.e. $\Delta(s, a)$ is defined; and C_a satisfies the following:

- for all tuples $(q, \star) \in C_s$ we have $(q, \star) \in C_a$ or $(q, (E, F)_\circ) \in C_a$ for some accepting condition (E, F), (q can "commit" to (E, F), or keep waiting)
- for all $(q, x) \in C_s$ with $x \neq \star$ we have $(q, x) \in C_a$, (all q are kept along with the commitments)
- all $(q, x) \in C_a$, not added by one of the two above items, are of the form $(q_{in}, \star)$ where q_{in} is the initial state of a Rabin automaton $R_{\varphi_i, I^\star}$ for $i \in I^\star$, (initial states can be added)

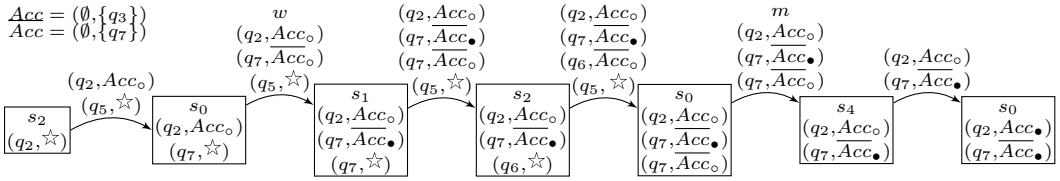

Figure 3 Illustration for Example 11. The names of actions from $\mathcal{M}$ are omitted when only a single action is available.

The randomness in $\mathcal{M}_\otimes$ comes only from $\mathcal{M}$. We set $\Delta_\otimes((s, C_s), (a, C_a))(t, C_t) = \Delta(s, a)(t)$ for any state (s, C_s), any action (a, C_a) legal in (s, C_s), and any state (t, C_t) such that C_a "deterministically evolves" by reading s into C_t. Precisely, we require that C_t is the minimal set such that for any $(q, x) \in C_a$ there is $(q', x') \in C_t$ with $q \overset{\nu(s)}{\to} q'$ and $x \overset{q'}{\rightsquigarrow} x'$ where the latter relation is defined by $\star \overset{q'}{\rightsquigarrow} \star$ and $\bot \overset{q'}{\rightsquigarrow} \bot$ and for any $\cdot \in \{\bullet, \circ\}$ by

- $(E, F). \overset{q'}{\rightsquigarrow} (E, F).$ if $q' \notin E \cup F$ and C is not fulfilled, (no special state visited)
- $(E, F). \overset{q'}{\rightsquigarrow} (E, F)_\circ$ if $q' \notin E \cup F$ and C is fulfilled, (resetting back to $\circ$)
- $(E, F). \overset{q'}{\rightsquigarrow} (E, F)_\bullet$ if $q' \in F$, (the accepting condition becomes fulfilled)
- $(E, F). \overset{q'}{\rightsquigarrow} \bot$ if $q' \in E$; (the accepting condition is violated)

Finally, a state is called *fulfilled* if its second component is fulfilled.

▶ **Example 11.** Figure 3 shows one path in the product $\mathcal{M}_\otimes$ for the MDP and the Rabin automata from Example 8. The path shown illustrates how the initial states can be added non-deterministically (in the first three steps), and then reaches a fulfilled state.

A very useful property of the product is that any strategy that ensures visiting fulfilled states infinitely often yields a strategy in the original MDP such that the automata the strategy added almost surely accept. This is formalised in the following lemma.

▶ **Lemma 12.** *For a deterministic strategy π in $\mathcal{M}_\otimes$ there is a strategy π' in $\mathcal{M}$ that for any $h = (s_0, C_0) \cdots (a_n, D_n)(s_{n+1}, C_{n+1})$ with $\mathbb{P}^h_\pi(\{\text{fulfilled state visited infinitely often}\}) = 1$:*

- $\mathbb{P}^{s_0}_{\pi'}(s_0 \dots a_n s_{n+1}) = \mathbb{P}^{(s_0, C_0)}_\pi(h)$, *and*
- *for any $(q, \star) \in D_n$ where R is the automaton of q, $\mathbb{P}^{s_0 a_0 \cdots s_n}_{\pi'}(\{\omega \mid R^q \text{ accepts } \omega\}) = 1$.*

To be able to use above lemma, we need to establish that it is *sufficient* to look for a strategy that visits fulfilled states infinitely often. In other words that existence of the satisfying strategy σ implies existence of a strategy that visits fulfilled states infinitely often. Here we use the following lemma saying that σ and $(s^\star, q^\star)$ give rise to two strategies in the product $\mathcal{M}_\otimes$ that can be used to add initial states into the collections, and to reach fulfilled states. We will show below how these strategies can be used to finish the proof of Proposition 7.

▶ **Lemma 13.** *Assume $s^\star, q^\star, I^\star$ are chosen as described on page 192. Then there are sets $M \subseteq S_\otimes$, $N \subseteq A_\otimes$ where N contains only "accumulating" actions, i.e. actions (a, C) with $\{(q_{in}, \star) \mid q_{in} \text{ is the initial state of } R_{\varphi_i, I^\star} \text{ for } i \in I^\star\} \subseteq C$; and there are finite-memory deterministic strategies π and ζ such that:*

1. *When starting in $(s, C) \in M$, π only uses actions from N and never leaves M*
2. *When starting in $(s, C) \in M \cup \{(s^\star, \{(q^\star, \star)\})\}$, ζ almost surely reaches a fulfilled state (possibly leaving M) and then reaches M.*

Proof idea. The proof is involved and gives a crucial insight into the main obstacles of the proof of Theorem 6. Due to the space constraints we only sketch it here.

We first prove that for any fixed ℓ, almost every ω that satisfies all $\mathbf{G}^1\varphi_i^{I^\star}$ has infinitely many *good* prefixes. Intuitively, a finite path h is *good* if, when starting from h, all the automata $R_{\varphi_i,I^\star}$ for $i \in I^\star$ started within ℓ first steps accept with probability at least $1-\lambda$.

In the second step, we show how to avoid actions that cause that any $R_{\varphi_i,I^\star}$ does not accept. To do so, we inductively start labelling the prefixes of runs of the MDP with elements of $\mathcal{C}$. Having fixed a label for a prefix, the label for its extension is obtained by "deterministic evolving" as in the definition of the product MDP, and by (non-deterministically) adding $(q_{in}, \star)$. The latter part is performed by switching between a "pseudo-accumulating" and "pseudo-reaching" phase. Initially, we start in a pseudo-reaching phase, only with singletons corresponding to the current state of $R_{\psi,I^\star}$, and do not add any $(q_{in}, \star)$. When a good prefix is reached (which happens almost surely), we switch to a pseudo-accumulating phase for the next ℓ steps and we keep adding "initial states" $(q_{in}, \star)$ of $R_{\varphi_i,I^\star}$ for each $i \in I^\star$. After ℓ steps, we switch back to a pseudo-reaching phase and do not add any new elements to the label until we pass through a state whose label is fulfilled and get to a good prefix again, in which point another pseudo-accumulating phase starts.

Along the way, we might obtain tuples of the form $(q, \bot)$ in the label, or we might not ever visit a fulfilled state. Indeed, if we repeated our steps to infinity, such an "error" might take place almost surely. However, before an error happens with too high probability, the labels start repeating because $\mathcal{C}$ is finite. We show that supposing ℓ was large enough and our tolerance λ was small enough, there must be a strategy that *almost-surely* traverses such a cycle without any error. We can extract from the pseudo-accumulating and pseudo-reaching phases of such a strategy the sets M (and N), given by the tuples of the MDP states (actions) and their labels. ◀

We are now ready to finish the proof of Proposition 7. We show that Lemma 13 allows us to construct a strategy $\sigma_\otimes$ for $\mathcal{M}_\otimes$ that almost surely (i) visits fulfilled states, and (ii) with frequency 1 it takes actions from N. By Lemma 12 this strategy yields an almost-surely winning strategy $\sigma_{s^\star,q^\star}$ in $\mathcal{M}$.

The strategy $\sigma_\otimes$ is constructed as follows. Inductively, for path h in $\mathcal{M}_\otimes$, we say that its first accumulating phase starts in the first step, ith accumulating phase takes i steps, and the $(i+1)$th accumulating phase starts when the set M is reached through a fulfilled state after the ith accumulating phase ended. Within every accumulating phase started in a history h, $\sigma_\otimes$ is defined to play as π initiated after h. Similarly, outside every accumulating phase ended in a history h, $\sigma_\otimes$ is defined to play as ζ.

4.2 The algorithm

To conclude the proof of Theorem 6, we need to give a procedure for computing the optimal probability of satisfying ψ. It works in the following steps (for details, see [14]):

1. Initialize $\Upsilon := \emptyset$, and construct $R_{\xi,I}$ for all $\xi \in \{\psi, \varphi_1, \ldots, \varphi_n\}$ and $I \subseteq \{1, \ldots, n\}$.
2. For every I find the largest sets (M_I, N_I) satisfying the conditions 1–2 of Lemma 13, and add to Υ all pairs (s, q) such that M_I can be almost-surely reached from $(s, \{(q, \star)\})$.
3. Compute an optimal strategy σ' for "reaching" Υ and return the probability. Intuitively,
   - we build the "naive" product of $\mathcal{M}$ with all the main automata $R_{\psi,I}$ for $I \subseteq \{1, \ldots, n\}$;
   - reaching Υ is reduced to ordinary reachability of all states of the form $(s, q_1, \ldots, q_m)$ such that $(s, q_i) \in \Upsilon$ for some i.
   - By standard algorithms for reachability in MDP, we find an optimal strategy σ'' in the naive product that easily induces the strategy σ' in $\mathcal{M}$.

By connecting Proposition 7, Lemmas 12 and 13, and the construction of $\sigma_{\otimes}$ above, there is a strategy σ in $\mathcal{M}$ yielding probability $\geq p$ iff the set Υ computed by the algorithm can also be reached with probability $\geq p$.

We briefly discuss the complexity of the algorithm. Each of the Rabin automata in step 1 above can be computed in time $2^{2^{\mathsf{poly}(|\varphi|)}}$, and since there is exponentially many such automata (in $|\varphi|$), step 1. takes time $2^{2^{poly(|\varphi|)}}$. Step 2 can be performed in time $\mathsf{poly}(S) \cdot 2^{2^{\mathsf{poly}(|\varphi|)}}$. In step 3 we are computing reachability probability in the naive product MDP which is of size $\mathsf{poly}(S) \cdot 2^{2^{\mathsf{poly}(|\varphi|)}}$, and so also this step can be done in time $\mathsf{poly}(S) \cdot 2^{2^{\mathsf{poly}(|\varphi|)}}$.

5 Conclusions

We have given algorithms for controller synthesis of the logic LTL extended with an operator expressing that frequencies of some events exceed a given bound. For Markov chains we gave an algorithm working with the complete logic, and for MDPs we require the formula to be from a certain fragment. The obvious next step is extending the MDP results to the whole fLTL. This will require new insights. Our product construction relies on the (non-trivial) observation that given $\mathbf{G}^1\varphi$, the formula φ is almost surely satisfied from any history of an accumulating phase. This is no longer true when the frequency bound is lower than 1. In such cases different histories may require different probability of satisfying φ. However, both authors strongly believe that even for these cases the problem is decidable. Another promising direction for future work is implementing the algorithms into a probabilistic model checker and evaluating their time requirements experimentally.

Acknowledgements. The authors would like to thank anonymous reviewers for their insightful comments on an earlier version of this paper. This work is partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center AVACS (SFB/TR 14), by the Czech Science Foundation under grant agreement P202/12/G061, by the EU 7th Framework Programme under grant agreement no. 295261 (MEALS) and 318490 (SENSATION), by the CDZ project 1023 (CAP), by the CAS/SAFEA International Partnership Program for Creative Research Teams, and EPSRC grant EP/M023656/1. Vojtěch Forejt is also affiliated with Faculty of Informatics, Masaryk University, Brno, Czech Republic.

References

1 Christel Baier, Boudewijn Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model checking continuous-time Markov chains by transient analysis. In *CAV*, volume 1855 of *LNCS*. Springer, 2000.

2 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

3 Christel Baier, Joachim Klein, Sascha Klüppelholz, and Sascha Wunderlich. Weight monitoring with linear temporal logic: Complexity and decidability. In *CSL-LICS*, page 11. ACM, 2014.

4 Roderick Bloem, Krishnendu Chatterjee, Thomas A Henzinger, and Barbara Jobstmann. Better quality in synthesis through quantitative objectives. In *CAV*, pages 140–156. Springer, 2009.

5 Udi Boker, Krishnendu Chatterjee, Thomas A Henzinger, and Orna Kupferman. Temporal specifications with accumulative values. In *LICS 2011*, pages 43–52. IEEE, 2011.

6 Benedikt Bollig, Normann Decker, and Martin Leucker. Frequency linear-time temporal logic. In *TASE'12*, pages 85–92, Beijing, China, July 2012. IEEE Computer Society Press.

7 Patricia Bouyer, Nicolas Markey, and Raj Mohan Matteplackel. Averaging in LTL. In Paolo Baldan and Daniele Gorla, editors, *CONCUR 2014*, volume 8704 of *LNCS*, pages 266–280. Springer, 2014.

8 Tomáš Brázdil, Vojtěch Forejt, and Antonín Kučera. Controller synthesis and verification for markov decision processes with qualitative branching time objectives. In *ICALP 2008*, volume 5126 of *LNCS*, pages 148–159. Springer, 2008.

9 Krishnendu Chatterjee and Laurent Doyen. Energy and mean-payoff parity Markov decision processes. In *MFCS 2011*, pages 206–218. Springer, 2011.

10 Krishnendu Chatterjee and Laurent Doyen. Games and markov decision processes with mean-payoff parity and energy parity objectives. In *MEMICS*, pages 37–46. Springer, 2012.

11 Krishnendu Chatterjee, Thomas A Henzinger, and Marcin Jurdzinski. Mean-payoff parity games. In *LICS 2005*, pages 178–187. IEEE, 2005.

12 Kai-lai Chung. *A Course in Probability Theory*. Academic Press, 3 edition, 2001.

13 Luca De Alfaro. How to specify and verify the long-run average behaviour of probabilistic systems. In *LICS 1998*, pages 454–465. IEEE, 1998.

14 Vojtěch Forejt and Jan Krčál. On frequency LTL in probabilistic systems. *CoRR*, abs/1501.05561, 2015.

15 J. Kemeny, J. Snell, and A. Knapp. *Denumerable Markov Chains*. Springer, 2nd edition, 1976.

16 Antonín Kučera and Oldřich Stražovský. On the controller synthesis for finite-state Markov decision processes. In *FSTTCS 2005*, pages 541–552. Springer, 2005.

17 M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic verification of herman's self-stabilisation algorithm. *Formal Aspects of Computing*, 24(4):661–670, 2012.

18 J. R. Norris. *Markov chains*. Cambridge University Press, 1998.

19 V. Shmatikov. Probabilistic model checking of an anonymity system. *Journal of Computer Security*, 12(3/4):355–377, 2004.

20 Takashi Tomita, Shigeki Hagihara, and Naoki Yonezaki. A probabilistic temporal logic with frequency operators and its model checking. In *INFINITY*, volume 73 of *EPTCS*, pages 79–93, 2011.

21 Takashi Tomita, Shin Hiura, Shigeki Hagihara, and Naoki Yonezaki. A temporal logic with mean-payoff constraints. In *Formal Methods and Soft. Eng.*, volume 7635 of *LNCS*. Springer, 2012.