Controller synthesis for MDPs and Frequency $\text{LTL}_{\setminus \mathbf{GU}}$

Vojtěch Forejt[1], Jan Krčál[2], and Jan Křetínský[3]

[1] Department of Computer Science, University of Oxford, UK
[2] Saarland University – Computer Science, Saarbrücken, Germany
[3] IST Austria

Abstract. Quantitative extensions of temporal logics have recently attracted significant attention. In this work, we study frequency LTL (fLTL), an extension of LTL which allows to speak about frequencies of events along an execution. Such an extension is particularly useful for probabilistic systems that often cannot fulfil strict qualitative guarantees on the behaviour. It has been recently shown that controller synthesis for Markov decision processes and fLTL is decidable when all the bounds on frequencies are 1. As a step towards a complete quantitative solution, we show that the problem is decidable for the fragment $\text{fLTL}_{\setminus \mathbf{GU}}$, where $\mathbf{U}$ does not occur in the scope of $\mathbf{G}$ (but still $\mathbf{F}$ can). Our solution is based on a novel translation of such quantitative formulae into equivalent deterministic automata.

1 Introduction

Markov decision processes (MDP) are a common choice when modelling systems that exhibit (un)controllable and probabilistic behaviour. In controller synthesis of MDPs, the goal is then to steer the system so that it meets certain property. Many properties specifying the desired behaviour, such as "the system is always responsive" can be easily captured by Linear Temporal Logic (LTL). This logic is in its nature qualitative and cannot express *quantitative* linear-time properties such as "a given failure happens only *rarely*". To overcome this limitation, especially apparent for stochastic systems, extensions of LTL with *frequency* operators have been recently studied [7, 8].

Such extensions come at a cost, and for example the "frequency until" operator can make the controller-synthesis problem undecidable already for non-stochastic systems [7, 8]. It turns out [19, 30, 31] that a way of providing significant added expressive power while preserving tractability is to extend LTL only by the "frequency globally" formulae $\mathbf{G}^{\geq p}\varphi$. Such a formula is satisfied if the long-run frequency of satisfying φ on an infinite path is at least p. More formally, $\mathbf{G}^{\geq p}\varphi$ is true on an infinite path $s_0 s_1 \cdots$ of an MDP if and only if $\frac{1}{n} \cdot |\{i \mid i < n \text{ and } s_i s_{i+1} \cdots \text{ satisfies } \varphi\}|$ is at least p as n tends to infinity. Because the relevant limit might not be defined, we need to consider two distinct operators, $\mathbf{G}^{\geq p}_{\inf}$ and $\mathbf{G}^{\geq p}_{\sup}$, whose definitions use limit inferior and limit superior, respectively. We call the resulting logic *frequency LTL (fLTL)*.

So far, MDP controller synthesis for fLTL has been shown decidable for the fragment containing only the operator $\mathbf{G}^{\geq 1}_{\text{inf}}$ [19]. Our paper makes a significant further step towards the ultimate goal of a model checking procedure for the whole fLTL. We address the general *quantitative* setting with arbitrary frequency bounds p and consider the fragment $\text{fLTL}_{\setminus \mathbf{GU}}$, which is obtained from frequency LTL by preventing the $\mathbf{U}$ operator from occurring inside $\mathbf{G}$ or $\mathbf{G}^{\geq p}$ formulas (but still allowing the $\mathbf{F}$ operator to occur anywhere in the formula). The approach we take is completely different from [19] where ad hoc product MDP construction is used, heavily relying on existence of certain types of strategies in the $\mathbf{G}^{\geq 1}_{\text{inf}}$ case. In this paper we provide, to the best of our knowledge, the first translation of a quantitative logic to equivalent *deterministic* automata. This allows us to take the standard automata-theoretic approach to verification [33]: after obtaining the finite automaton, we do not deal with the structure of the formula originally given, and we solve a (reasonably simple) synthesis problem on a product of the single automaton with the MDP.

Relations of various kinds of logics and automata are widely studied (see e.g. [32, 29, 16]), and our results provide new insights into this area for quantitative logics. Previous work [31] offered only translation of a similar logic to *non-deterministic* "mean-payoff Büchi automata" noting that it is difficult to give an analogous reduction to *deterministic* "mean-payoff Rabin automata". The reason is that the non-determinism is inherently present in the form of guessing whether the subformulas of $\mathbf{G}^{\geq p}$ are satisfied on a suffix. Our construction overcomes this difficulty and offers equivalent deterministic automata. It is a first and highly non-trivial step towards providing a reduction for the complete logic.

Although our algorithm does not allow us to handle the extension of the whole LTL, the considered fragment $\text{fLTL}_{\setminus \mathbf{GU}}$ contains a large class of formulas and offers significant expressive power. It subsumes the GR(1) fragment of LTL [5], which has found use in synthesis for hardware designs. The $\mathbf{U}$ operator, although not allowed within a scope of a $\mathbf{G}$ operator, can still be used for example to distinguish paths based on their prefixes. As an example synthesis problem expressible in this fragment, consider a cluster of servers where each server plays either a role of a load-balancer or a worker. On startup, each server listens for a message specifying its role. A load-balancer forwards each request and only waits for a confirmation whereas a worker processes the requests itself. A specification for a single server in the cluster can require, for example, that the following formula (with propositions explained above) holds with probability at least 0.95:

$$\Big(\big(l \,\mathbf{U}\, b\big) \to \mathbf{G}^{\geq 0.99}\big(r \to \mathbf{X}(f \wedge \mathbf{F}c)\big)\Big) \wedge \Big(\big(l \,\mathbf{U}\, w\big) \to \mathbf{G}^{\geq 0.85}\big(r \to (\mathbf{X}p \vee \mathbf{X}\mathbf{X}p)\big)\Big)$$

Related work. Frequency LTL was studied in another variant in [7, 8] where a *frequency until* operator is introduced in two different LTL-like logics, and undecidability is proved for problems relevant to our setting. The work [7] also yields decidability with restricted nesting of the frequency until operator; as the decidable fragment in [7] does not contain frequency-globally operator, it is not possible to express many useful properties expressible in our logic. A logic that

speaks about frequencies on a finite interval was introduced in [30], but the paper provides algorithms only for Markov chains and a bounded fragment of the logic.

Model checking MDPs against LTL objectives relies on the automata-theoretic approach, namely on translating LTL to automata that are to some extent deterministic [15]. This typically involves translating LTL to non-deterministic automata, which are then determinized using e.g. Safra's construction. During the determinization, the original structure of the formula is lost, which prevents us from extending this technique to the frequency setting. However, an alternative technique of translating LTL directly to deterministic automata has been developed [26, 25, 17], where the logical structure is preserved. In our work, we extend the algorithm for $\mathrm{LTL}_{\setminus \mathbf{GU}}$ partially sketched in [25]. In Section 6, we explain why adapting the algorithm for full LTL [17] is difficult. Translation of $\mathrm{LTL}_{\setminus \mathbf{GU}}$ to other kinds of automata has been considered also in [22].

Our technique relies on a solution of a multi-objective mean-payoff problem on MDP [9, 13]. Previous results only consider limit inferior rewards, and so we cannot use them as off-the-shelf results, but need to adapt them first to our setting with both inferior and superior limits together with Rabin condition. There are several works that combine mean-payoff objectives with e.g. logics or parity objectives, but in most cases only simple atomic propositions can be used to define the payoff [4, 6, 11]. 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. Further, [1] introduces a variant of LTL with a discounted-future operator. Finally, techniques closely related to the ones in this paper are used in [18, 14, 28].

Our contributions. To our best knowledge, this paper gives the first decidability result for probabilistic verification against linear-time temporal logics extended by *quantitative* frequency operators with *complex nested subformulas* of the logic. It works in two steps, keeping the same time complexity as for ordinary LTL. In the first step, a $\mathrm{fLTL}_{\setminus \mathbf{GU}}$ formula gets translated to an equivalent *deterministic* generalized Rabin automaton extended with mean-payoff objectives. This step is inspired by previous work [25], but the extension with auxiliary automata for $\mathbf{G}^{\geq p}$ requires a different construction. The second step is the analysis of MDPs against conjunction of limit inferior mean-payoff, limit superior mean-payoff, and generalized Rabin objectives. This result is obtained by adapting and combining several existing involved proof techniques [13, 10].

The paper is organised as follows: the main algorithm is given in Sec. 3, relegating details of the two steps above to Sec. 4 and 5. Full proofs are in [20].

2 Preliminaries

We use $\mathbb{N}$ and $\mathbb{Q}$ to denote the sets of non-negative integers and rational numbers. The set of all distributions over a countable set X is denoted by $\mathit{Dist}(X)$. For a predicate P, the *indicator function* $\mathbb{1}_P$ equals 1 if P is true, and 0 if P is false.

Markov decision processes (MDPs). An MDP is a tuple $\mathsf{M} = (S, A, \mathit{Act}, \delta, \hat{s})$ where S is a finite set of states, A is a finite set of actions, $\mathit{Act} : S \to 2^A \setminus \{\emptyset\}$

assigns to each state s the set $Act(s)$ of actions enabled in s, $\delta : A \to Dist(S)$ is a probabilistic transition function that given an action a gives a probability distribution over the successor states, and $\hat{s}$ is the initial state. To simplify notation, w.l.o.g. we require that every action is enabled in exactly one state.

Strategies. A strategy in an MDP M is a "recipe" to choose actions. Formally, it is a function $\sigma : (SA)^*S \to Dist(A)$ that given a finite path w, representing the history of a play, gives a probability distribution over the actions enabled in the last state. A strategy σ in M induces a *Markov chain* M^σ which is a tuple $(L, P, \hat{s})$ where the set of *locations* $L = (S \times A)^* \times S$ encodes the history of the play, $\hat{s}$ is an *initial location*, and P is a *probabilistic transition function* that assigns to each location a probability distribution over successor locations defined by $P(h)(h\,a\,s) \;=\; \sigma(h)(a) \cdot \delta(a)(s)$. for all $h \in (SA)^*S$, $a \in A$ and $s \in S$.

The probability space of the runs of the Markov chain is denoted by $\mathbb{P}^\sigma_{\mathsf{M}}$ and defined in the standard way [21, 20].

End components. A tuple (T, B) with $\emptyset \neq T \subseteq S$ and $B \subseteq \bigcup_{t \in T} Act(t)$ is an *end component* of M if (1) for all $a \in B$, whenever $\delta(a)(s') > 0$ then $s' \in T$; and (2) for all $s, t \in T$ there is a path $w = s_1 a_1 \cdots a_{k-1} s_k$ such that $s_1 = s$, $s_k = t$, and all states and actions that appear in w belong to T and B, respectively. An end component (T, B) is a *maximal end component (MEC)* if it is maximal with respect to the componentwise subset ordering. Given an MDP, the set of MECs is denoted by MEC. Finally, an MDP is *strongly connected* if (S, A) is a MEC.

Frequency linear temporal logic (fLTL). The formulae of the logic fLTL are given by the following syntax:

$$\varphi \quad ::= \quad \mathbf{tt} \mid \mathbf{ff} \mid a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \mathbf{F}\varphi \mid \mathbf{G}\varphi \mid \varphi \mathbf{U} \varphi \mid \mathbf{G}^{\bowtie p}_{\mathrm{ext}} \varphi$$

over a finite set Ap of atomic propositions, $\bowtie \in \{\geq, >\}$, $p \in [0, 1] \cap \mathbb{Q}$, and $\mathrm{ext} \in \{\inf, \sup\}$. A formula that is neither a conjunction, nor a disjunction is called *non-Boolean*. The set of non-Boolean subformulas of φ is denoted by $\mathsf{sf}(\varphi)$.

Words and fLTL Semantics. Let $w \in (2^{Ap})^\omega$ be an infinite word. The ith letter of w is denoted $w[i]$, i.e. $w = w[0]w[1] \cdots$. We write w^{ij} for the finite word $w[i]w[i+1] \cdots w[j]$, and $w^{i\infty}$ or just w^i for the suffix $w[i]w[i+1] \cdots$. The semantics of a formula on a word w is defined inductively: for $\mathbf{tt}$, $\mathbf{ff}$, $\wedge$, $\vee$, and for atomic propositions and their negations, the definition is straightforward, for the remaining operators we define:

$$\begin{array}{lll} w \models \mathbf{X}\varphi \iff w^1 \models \varphi & & w \models \varphi \mathbf{U} \psi \iff \exists\, k \in \mathbb{N} : w^k \models \psi \text{ and} \\ w \models \mathbf{F}\varphi \iff \exists\, k \in \mathbb{N} : w^k \models \varphi & & \qquad\qquad\quad \forall\, 0 \leq j < k : w^j \models \varphi \\ w \models \mathbf{G}\varphi \iff \forall\, k \in \mathbb{N} : w^k \models \varphi & & w \models \mathbf{G}^{\bowtie p}_{\mathrm{ext}} \varphi \iff \mathrm{lr}_{\mathrm{ext}}(\mathbb{1}_{w^0 \models \varphi} \mathbb{1}_{w^1 \models \varphi} \cdots) \bowtie p \end{array}$$

where we set $\mathrm{lr}_{\mathrm{ext}}(q_1 q_2 \cdots) := \lim \mathrm{ext}_{i \to \infty} \frac{1}{i} \sum_{j=1}^{i} q_i$. By $\mathsf{L}(\varphi)$ we denote the set $\{w \in (2^{Ap})^\omega \mid w \models \varphi\}$ of words satisfying φ.

The $\mathrm{fLTL}_{\setminus \mathbf{GU}}$ fragment of fLTL is defined by disallowing occurrences of $\mathbf{U}$ in $\mathbf{G}$-formulae, i.e. it is given by the following syntax for φ:

$$\varphi ::= a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U} \varphi \mid \mathbf{F}\varphi \mid \mathbf{G}\xi \mid \mathbf{G}^{\bowtie p}_{\mathrm{ext}} \xi$$

$$\xi ::= a \mid \neg a \mid \xi \wedge \xi \mid \xi \vee \xi \mid \mathbf{X}\xi \mid \mathbf{F}\xi \mid \mathbf{G}\xi \mid \mathbf{G}^{\bowtie p}_{\text{ext}}\xi$$

Note that restricting negations to atomic propositions is without loss of generality as all operators are closed under negation, for example $\neg\mathbf{G}^{\geq p}_{\text{inf}}\varphi \equiv \mathbf{G}^{>1-p}_{\text{sup}}\neg\varphi$ or $\neg\mathbf{G}^{>p}_{\text{sup}}\varphi \equiv \mathbf{G}^{\geq 1-p}_{\text{inf}}\neg\varphi$. Furthermore, we could easily allow $\bowtie$ to range also over $\leq$ and $<$ as $\mathbf{G}^{\leq p}_{\text{inf}}\varphi \equiv \mathbf{G}^{\geq 1-p}_{\text{sup}}\neg\varphi$ and $\mathbf{G}^{<p}_{\text{inf}}\varphi \equiv \mathbf{G}^{>1-p}_{\text{sup}}\neg\varphi$.

Automata. Let us fix a finite alphabet Σ. A deterministic *labelled transition system (LTS)* over Σ is a tuple (Q, q_0, δ) where Q is a finite set of states, q_0 is the initial state, and $\delta : Q \times \Sigma \to Q$ is a partial transition function. We denote $\delta(q, a) = q'$ also by $q \xrightarrow{a} q'$. A *run* of the LTS $\mathcal{S}$ over an infinite word w is a sequence of states $\mathcal{S}(w) = q_0 q_1 \cdots$ such that $q_{i+1} = \delta(q_i, w[i])$. For a finite word w of length n, we denote by $\mathcal{S}(w)$ the state q_n in which $\mathcal{S}$ is after reading w.

An *acceptance condition* is a positive boolean formula over formal variables

$$\{Inf(S), Fin(S), MP^{\bowtie p}_{\text{ext}}(r) \mid S \subseteq Q, \text{ext} \in \{\text{inf}, \text{sup}\}, \bowtie \in \{\geq, >\}, p \in \mathbb{Q}, r : Q \to \mathbb{Q}\}.$$

Given a run ρ and an acceptance condition α, we assign truth values as follows:

- $Inf(S)$ is true iff ρ visits (some state of) S infinitely often,
- $Fin(S)$ is true iff ρ visits (all states of) S finitely often,
- $MP^{\bowtie p}_{\text{ext}}(r)$ is true iff $\text{lr}_{\text{ext}}(r(\rho[0])r(\rho[1])\cdots) \bowtie p$.

The run ρ satisfies α if this truth-assignment makes α true. An *automaton* $\mathcal{A}$ is an LTS with an acceptance condition α. The language of $\mathcal{A}$, denoted by $\mathsf{L}(\mathcal{A})$, is the set of all words inducing a run satisfying α. An acceptance condition α is a *Büchi*, *generalized Büchi*, or *co-Büchi* acceptance condition if it is of the form $Inf(S)$, $\bigwedge_i Inf(S_i)$, or $Fin(S)$, respectively. Further, α is a *generalized Rabin mean-payoff*, or a *generalized Büchi mean-payoff* acceptance condition if it is in disjunctive normal form, or if it is a conjunction not containing any $Fin(S)$, respectively. For each acceptance condition we define a corresponding automaton, e.g. *deterministic generalized Rabin mean-payoff automaton (DGRMA)*.

3 Model-checking algorithm

In this section, we state the problem of model checking MDPs against $\text{fLTL}_{\setminus\mathbf{GU}}$ specifications and provide a solution. As a black-box we use two novel routines described in detail in the following two sections. All proofs are in the appendix.

Given an MDP M and a valuation $\nu : S \to 2^{Ap}$ of its states, we say that its run $\omega = s_0 a_0 s_1 a_1 \cdots$ *satisfies* φ, written $\omega \models \varphi$, if $\nu(s_0)\nu(s_1)\cdots \models \varphi$. We use $\mathbb{P}^\sigma[\varphi]$ as a shorthand for the probability of all runs satisfying φ, i.e. $\mathbb{P}^\sigma_{\mathsf{M}}[\{\omega \mid \omega \models \varphi\}]$. This paper is concerned with the following task:

> **Controller synthesis problem:** Given an MDP with a valuation, an $\text{fLTL}_{\setminus\mathbf{GU}}$ formula φ and $x \in [0, 1]$, decide whether $\mathbb{P}^\sigma[\varphi] \geq x$ for some strategy σ, and if so, construct such a *witness* strategy.

The following is the main result of the paper.

Theorem 1. *The controller synthesis problem for MDPs and fLTL$_{\setminus \mathbf{GU}}$ is decidable and the witness strategy can be constructed in doubly exponential time.*

In this section, we present an algorithm for Theorem 1. The skeleton of our algorithm is the same as for the standard model-checking algorithm for MDPs against LTL. It proceeds in three steps. Given an MDP M and a formula φ,

1. compute a deterministic automaton $\mathcal{A}$ such that $\mathsf{L}(\mathcal{A}) = \mathsf{L}(\varphi)$,
2. compute the product MDP $\mathsf{M} \times \mathcal{A}$,
3. analyse the product MDP $\mathsf{M} \times \mathcal{A}$.

In the following, we concretize these three steps to fit our setting.

1. Deterministic automaton For ordinary LTL, usually a Rabin automaton or a generalized Rabin automaton is constructed [27, 23, 17, 24]. Since in our setting, along with ω-regular language the specification also includes quantitative constraints over runs, we generate a DGRMA. The next theorem is the first black box, detailed in Section 4.

Theorem 2. *For any fLTL$_{\setminus \mathbf{GU}}$ formula, there is a DGRMA $\mathcal{A}$, constructible in doubly exponential time, such that $\mathsf{L}(\mathcal{A}) = \mathsf{L}(\varphi)$, and the acceptance condition is of exponential size.*

2. Product Computing the synchronous parallel product of the MDP $\mathsf{M} = (S, A, Act, \Delta, \hat{s})$ with valuation $\nu : S \to 2^{Ap}$ and the LTS (Q, i, δ) over 2^{Ap} underlying $\mathcal{A}$ is rather straightforward. The product $\mathsf{M} \times \mathcal{A}$ is again an MDP $(S \times Q, A \times Q, Act', \Delta', (\hat{s}, \hat{q}))$ where[4] $Act'((s,q)) = Act(s) \times \{q\}$, $\hat{q} = \delta(i, \nu(\hat{s}))$, and $\Delta'\big((a,q)\big)\big((s,\bar{q})\big)$ is equal to $\Delta(a)(s)$ if $\delta(q, \nu(s)) = \bar{q}$, and to 0 otherwise. We lift acceptance conditions Acc of $\mathcal{A}$ to $\mathsf{M} \times \mathcal{A}$: a run of $\mathsf{M} \times \mathcal{A}$ satisfies Acc if its projection to the component of the automata states satisfies Acc.[5]

3. Product analysis The MDP $\mathsf{M} \times \mathcal{A}$ is solved with respect to Acc, i.e., a strategy in $\mathsf{M} \times \mathcal{A}$ is found that maximizes the probability of satisfying Acc. Such a strategy then induces a (history-dependent) strategy on M in a straightforward manner. Observe that for DGRMA, it is sufficient to consider the setting with

$$Acc = \bigvee_{i=1}^{k} (Fin(F_i) \wedge Acc'_i) \tag{1}$$

where Acc'_i is a conjunction of several Inf and MP (in contrast with a Rabin condition used for ordinary LTL where Acc'_i is simply of the form $Inf(I_i)$). Indeed, one can replace each $\bigwedge_j Fin(F_j)$ by $Fin(\bigcup_j F_j)$ to obtain the desired form, since avoiding several sets is equivalent to avoiding their union.

For a condition of the form (1), the solution is obtained as follows:

[4] In order to guarantee that each action is enabled in at most one state, we have a copy of each original action for each state of the automaton.

[5] Technically, the projection should be preceded by i to get a run of the automaton, but the acceptance does not depend on any finite prefix of the sequence of states.

1. For $i = 1, 2, \ldots, k$:
   (a) Remove the set of states F_i from the MDP.
   (b) Compute the MEC decomposition.
   (c) Mark each MEC C as winning iff AcceptingMEC(C, Acc'_i) returns Yes.
   (d) Let W_i be the componentwise union of winning MECs above.
2. Let W be the componentwise union of all W_i for $1 \leq i \leq k$.
3. Return the maximal probability to reach the set W in the MDP.

The procedure AcceptingMEC(C, Acc'_i) is the second black box used in our algorithm, detailed in Section 5. It decides, whether the maximum probability of satisfying Acc'_i in C is 1 (return Yes), or 0 (return No).

Theorem 3. *For a strongly connected MDP* M *and a generalized Büchi mean-payoff acceptance condition Acc, the maximal probability to satisfy Acc is either* 1 *or* 0*, and is the same for all initial states. Moreover, there is a polynomial-time algorithm that computes this probability, and also outputs a witnessing strategy if the probability is* 1*.*

The procedure is rather complex in our case, as opposed to standard cases such as Rabin condition, where a MEC is accepting for $Acc'_i = Inf(I_i)$ if its states intersect I_i; or a generalized Rabin condition [12], where a MEC is accepting for $Acc'_i = \bigwedge_{j=1}^{\ell_i} Inf(I_{ij})$ if its states intersect with each I_i^j, for $j = 1, 2, \ldots, \ell_i$.

Finishing the proof of Theorem 1 Note that for MDPs that are not strongly connected, the maximum probability might not be in $\{0, 1\}$. Therefore, the problem is decomposed into a qualitative satisfaction problem in step 1.(c) and a quantitative reachability problem in step 3. Consequently, the proof of correctness is the same as the proofs for LTL via Rabin automata [2] and generalized Rabin automata [12]. The complexity follows from Theorem 2 and 3. Finally, the overall witness strategy first reaches the winning MECs and if they are reached it switches to the witness strategies from Theorem 3.

Remark 1. We remark that by a simple modification of the product construction above and of the proof of Theorem 3, we obtain an algorithm synthesising a strategy achieving a given bound w.r.t. multiple mean-payoff objectives (with a combination of superior and inferior limits) and (generalized) Rabin acceptance condition for *general* (not necessarily strongly connected) MDP.

4 Automata characterization of fLTL$_{\setminus \mathbf{GU}}$

In this section, we prove Theorem 2. We give an algorithm for translating a given fLTL$_{\setminus \mathbf{GU}}$ formula φ into a deterministic generalized Rabin mean-payoff automaton $\mathcal{A}$ that recognizes words satisfying φ. For the rest of the section, let φ be an fLTL$_{\setminus \mathbf{GU}}$ formula. Further, $\mathbb{F}$, $\mathbb{G}$, $\mathbb{G}^{\bowtie}$, and sf denote the set of $\mathbf{F}$-, $\mathbf{G}$-, $\mathbf{G}^{\bowtie p}_{\text{ext}}$-, and non-Boolean subformulas of φ, respectively.

In order to obtain an automaton for the formula, we first need to give a more operational view on fLTL. To this end, we use expansions of the formulae in a

very similar way as they are used, for instance, in tableaux techniques for LTL translation to automata, or for deciding LTL satisfiability. We define a symbolic one-step unfolding (expansion) Unf of a formula inductively by the rules below. Further, for a valuation $\nu \subseteq Ap$, we define the "next step under ν"-operator. This operator (1) substitutes unguarded atomic propositions for their truth values, and (2) peels off the outer $\mathbf{X}$-operator whenever it is present. Formally, we define

$$\begin{aligned}
\mathsf{Unf}(\psi_1 \wedge \psi_2) &= \mathsf{Unf}(\psi_1) \wedge \mathsf{Unf}(\psi_2) \\
\mathsf{Unf}(\psi_1 \vee \psi_2) &= \mathsf{Unf}(\psi_1) \vee \mathsf{Unf}(\psi_2) \\
\mathsf{Unf}(\mathbf{F}\psi_1) &= \mathsf{Unf}(\psi_1) \vee \mathbf{XF}\psi_1 \\
\mathsf{Unf}(\mathbf{G}\psi_1) &= \mathsf{Unf}(\psi_1) \wedge \mathbf{XG}\psi_1 \\
\mathsf{Unf}(\psi_1 \mathbf{U} \psi_2) &= \mathsf{Unf}(\psi_2) \vee \big(\mathsf{Unf}(\psi_1) \wedge \mathbf{X}(\psi_1 \mathbf{U} \psi_2)\big) \\
\mathsf{Unf}(\mathbf{G}^{\bowtie p}_{\mathrm{ext}}\psi_1) &= \mathbf{tt} \wedge \mathbf{XG}^{\bowtie p}_{\mathrm{ext}}\psi_1 \\
\mathsf{Unf}(\psi) &= \psi \text{ for any other } \psi
\end{aligned}$$

$$\begin{aligned}
(\psi_1 \wedge \psi_2)[\nu] &= \psi_1[\nu] \wedge \psi_2[\nu] \\
(\psi_1 \vee \psi_2)[\nu] &= \psi_1[\nu] \vee \psi_2[\nu] \\
a[\nu] &= \begin{cases} \mathbf{tt} & \text{if } a \in \nu \\ \mathbf{ff} & \text{if } a \notin \nu \end{cases} \\
\neg a[\nu] &= \begin{cases} \mathbf{ff} & \text{if } a \in \nu \\ \mathbf{tt} & \text{if } a \notin \nu \end{cases} \\
(\mathbf{X}\psi_1)[\nu] &= \psi_1 \\
\psi[\nu] &= \psi \text{ for any other } \psi
\end{aligned}$$

Note that after unfolding, a formula becomes a positive Boolean combination over literals (atomic propositions and their negations) and $\mathbf{X}$-formulae. The resulting formula is LTL-equivalent to the original formula. The formulae of the form $\mathbf{G}^{\bowtie p}_{\mathrm{ext}}\psi$ have "dummy" unfolding; they are dealt with in a special way later. Combined with unfolding, the "next step"-operator then preserves and reflects satisfaction on the given word:

Lemma 1. *For every word w and $fLTL_{\setminus \mathbf{GU}}$ formula φ, we have $w \models \varphi$ if and only if $w^1 \models (\mathsf{Unf}(\varphi))[w[0]]$.*

The construction of $\mathcal{A}$ proceeds in several steps. We first construct a "master" transition system, which monitors the formula and transforms it in each step to always keep exactly the formula that needs to be satisfied at the moment. However, this can only deal with properties whose satisfaction has a finite witness, e.g. $\mathbf{F}a$. Therefore we construct a set of "slave" automata, which check whether "infinitary" properties (with no finite witness), e.g., $\mathbf{FG}a$, hold or not. They pass this information to the master, who decides on acceptance of the word.

4.1 Construction of master transition system $\mathcal{M}$

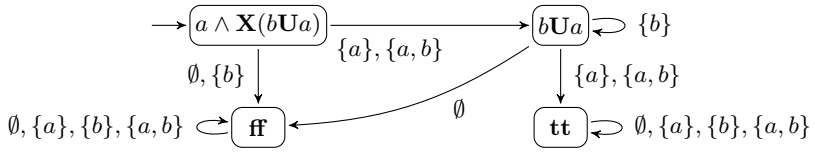
We define a LTS $\mathcal{M} = (Q, \varphi, \delta^{\mathcal{M}})$ over 2^{Ap} by letting Q be the set of positive Boolean functions[6] over sf, by letting φ be the initial state, and by letting the transition function $\delta^{\mathcal{M}}$, for every $\nu \subseteq Ap$ and $\psi \in Q$, contain $\psi \xrightarrow{\nu} (\mathsf{Unf}(\psi))[\nu]$.

The master automaton keeps the property that is still required up to date:

[6] We use Boolean functions, i.e. classes of propositionally equivalent formulae, to obtain a finite state space. To avoid clutter, when referring to such a Boolean function, we use some formula representing the respective equivalence class. The choice of the representing formula is not relevant since, for all operations we use, the propositional equivalence is a congruence, see [20]. Note that, in particular, $\mathbf{tt}, \mathbf{ff} \in Q$.

Lemma 2 (Local (finitary) correctness of master LTS). *Let w be a word and $\mathcal{M}(w) = \varphi_0\varphi_1\cdots$ the corresponding run. Then for all $n \in \mathbb{N}$, we have $w \models \varphi$ if and only if $w^n \models \varphi_n$.*

Example 1. The formula $\varphi = a \wedge \mathbf{X}(b\mathbf{U}a)$ yields a master LTS depicted below.

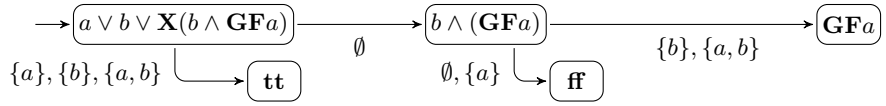


One can observe that for an fLTL formula φ with no $\mathbf{G}$- and $\mathbf{G}^{\bowtie p}_{\text{ext}}$-operators, we have $w \models \varphi$ iff the state $\mathbf{tt}$ is reached while reading w. However, for formulae with $\mathbf{G}$-operators (and thus without finite witnesses in general), this claim no longer holds. To check such behaviour we construct auxiliary "slave" automata.

4.2 Construction of slave transition systems $\mathcal{S}(\xi)$

We define a LTS $\mathcal{S}(\xi) = (Q, \xi, \delta^{\mathcal{S}})$ over 2^{Ap} with the same state space as $\mathcal{M}$ and the initial state $\xi \in Q$. Furthermore, we call a state ψ a *sink*, written $\psi \in \mathsf{Sink}$, iff for all $\nu \subseteq Ap$ we have $\psi[\nu] = \psi$. Finally, the transition relation $\delta^{\mathcal{S}}$, for every $\nu \subseteq Ap$ and $\psi \in Q \setminus \mathsf{Sink}$, contains $\psi \xrightarrow{\nu} \psi[\nu]$.

Example 2. The slave LTS for the formula $\xi = a \vee b \vee \mathbf{X}(b \wedge \mathbf{GF}a)$ has a structure depicted in the following diagram:



Note that we do not unfold any inner $\mathbf{F}$- and $\mathbf{G}$-formulae. Observe that if we start reading w at the ith position and end up in $\mathbf{tt}$, we have $w^i \models \xi$. Similarly, if we end up in $\mathbf{ff}$ we have $w^i \not\models \xi$. This way we can monitor for which position ξ holds and will be able to determine if it holds, for instance, infinitely often. But what about when we end up in $\mathbf{GF}a$? Intuitively, this state is accepting or rejecting depending on whether $\mathbf{GF}a$ holds or not. Since this cannot be checked in finite time, we delegate this task to yet another slave, now responsible for $\mathbf{GF}a$. Thus instead of deciding whether $\mathbf{GF}a$ holds, we may use it as an *assumption* in the automaton for ξ and let the automaton for $\mathbf{GF}a$ check whether the assumption turns out correct.

Let $\mathcal{R}\mathrm{ec} := \mathbb{F} \cup \mathbb{G} \cup \mathbb{G}^{\bowtie}$. This is the set of subformulas that are potentially difficult to check in finite time. Subsets of $\mathcal{R}\mathrm{ec}$ can be used as assumptions to prove other assumptions and in the end also the acceptance. Given a set of formulae Ψ and a formula ψ, we say that Ψ *(propositionally) proves* ψ, written $\Psi \vdash \psi$, if ψ can be deduced from formulae in Ψ using only propositional reasoning (for a formal definition see [20]). So, for instance, $\{\mathbf{GF}a\} \vdash \mathbf{GF}a \vee \mathbf{G}b$, but $\mathbf{GF}a \nvdash \mathbf{F}a$.

The following is the ideal assumption set we would like our automaton to identify. For a fixed word w, we denote by $\mathcal{R}(w)$ the set

$$\{\mathbf{F}\xi \in \mathbb{F} \mid w \models \mathbf{GF}\xi\} \cup \{\mathbf{G}\xi \in \mathbb{G} \mid w \models \mathbf{FG}\xi\} \cup \{\mathbf{G}^{\bowtie p}_{\mathrm{ext}}\xi \in \mathbb{G}^{\bowtie} \mid w \models \mathbf{G}^{\bowtie p}_{\mathrm{ext}}\xi\}$$

of formulae in $\mathcal{R}\mathrm{ec}$ eventually always satisfied on w. The slave LTS is useful for recognizing whether its respective formula ξ holds infinitely often, almost always, or with the given frequency. Intuitively, it reduces this problem for a given formula to the problems for its subformulas in $\mathcal{R}\mathrm{ec}$:

Lemma 3 (Correctness of slave LTS). *Let us fix $\xi \in \mathsf{sf}$ and a word w. For any $\mathcal{R} \in \mathcal{R}\mathrm{ec}$, we denote by $Sat(\mathcal{R})$ the set $\{i \in \mathbb{N} \mid \exists j \geq i : \mathcal{R} \vdash \mathcal{S}(\xi)(w^{ij})\}$. Then for any $\underline{\mathcal{R}}, \overline{\mathcal{R}} \subseteq \mathcal{R}\mathrm{ec}$ such that $\underline{\mathcal{R}} \subseteq \mathcal{R}(w) \subseteq \overline{\mathcal{R}}$, we have*

$$Sat(\underline{\mathcal{R}}) \text{ is infinite} \implies w \models \mathbf{GF}\xi \implies Sat(\overline{\mathcal{R}}) \text{ is infinite} \tag{2}$$

$$\mathbb{N} \setminus Sat(\underline{\mathcal{R}}) \text{ is finite} \implies w \models \mathbf{FG}\xi \implies \mathbb{N} \setminus Sat(\overline{\mathcal{R}}) \text{ is finite} \tag{3}$$

$$\mathrm{lr}_{\mathrm{ext}}\big(\big(\mathbb{1}_{i \in Sat(\underline{\mathcal{R}})}\big)_{i=0}^{\infty}\big) \bowtie p \implies w \models \mathbf{G}^{\bowtie p}_{\mathrm{ext}}\xi \implies \mathrm{lr}_{\mathrm{ext}}\big(\big(\mathbb{1}_{i \in Sat(\overline{\mathcal{R}})}\big)_{i=0}^{\infty}\big) \bowtie p \tag{4}$$

Before we put the slaves together to determine $\mathcal{R}(w)$, we define *slave automata*. In order to express the constraints from Lemma 3 as acceptance conditions, we need to transform the underlying LTS. Intuitively, we replace quantification over various starting positions for runs by a subset construction. This means that in each step we put a *token* to the initial state and move all previously present tokens to their successor states.

Büchi For a formula $\mathbf{F}\xi \in \mathbb{F}$, its slave LTS $\mathcal{S}(\xi) = (Q, \xi, \delta^{\mathcal{S}})$, and $\mathcal{R} \subseteq \mathcal{R}\mathrm{ec}$, we define a Büchi automaton $\mathcal{S}_{\mathbf{GF}}(\xi, \mathcal{R}) = (2^Q, \{\xi\}, \delta)$ over 2^{Ap} by setting

$$\Psi \xrightarrow{\nu} \{\delta^{\mathcal{S}}(\psi, \nu) \mid \psi \in \Psi \setminus \mathsf{Sink}\} \cup \{\xi\} \qquad \text{for every } \nu \subseteq Ap$$

and the Büchi acceptance condition $Inf(\{\Psi \subseteq Q \mid \exists \psi \in \Psi \cap \mathsf{Sink} : \mathcal{R} \vdash \psi\})$.

In other words, the automaton accepts if infinitely often a token ends up in an *accepting sink*, i.e., element of Sink that is provable from $\mathcal{R}$. For Example 2, depending on whether we assume $\mathbf{GF}a \in \mathcal{R}$ or not, the accepting sinks are $\mathbf{tt}$ and $\mathbf{GF}a$, or only $\mathbf{tt}$, respectively.

Co-Büchi For a formula $\mathbf{G}\xi \in \mathbb{G}$, its slave LTS $\mathcal{S}(\xi) = (Q, \xi, \delta^{\mathcal{S}})$ and $\mathcal{R} \subseteq \mathcal{R}\mathrm{ec}$, we define a co-Büchi automaton $\mathcal{S}_{\mathbf{FG}}(\xi, \mathcal{R}) = (2^Q, \{\xi\}, \delta)$ over 2^{Ap} with the same LTS as above. It differs from the Büchi automaton only by having a co-Büchi acceptance condition $Fin(\{\Psi \subseteq Q \mid \exists \psi \in \Psi \cap \mathsf{Sink} : \mathcal{R} \nvdash \psi\})$.

Mean-payoff For a formula $\mathbf{G}^{\bowtie p}_{\mathrm{ext}}\xi \in \mathbb{G}^{\bowtie}$, its slave LTS $\mathcal{S}(\xi) = (Q, \xi, \delta^{\mathcal{S}})$, and $\mathcal{R} \subseteq \mathcal{R}\mathrm{ec}$ we define a *mean-payoff automaton* $\mathcal{S}_{\mathbf{G}^{\bowtie p}_{\mathrm{ext}}}(\xi, \mathcal{R}) = (|Q|^Q, \mathbb{1}_{\xi}, \delta)$ over 2^{Ap} so that for every $\nu \subseteq Ap$, we have $f \xrightarrow{\nu} f'$ where

$$f'(\psi') = \mathbb{1}_{\xi}(\psi') + \sum_{\delta^{\mathcal{S}}(\psi, \nu) = \psi'} f(\psi).$$

Intuitively, we always count the number of tokens in each state. When a step is taken, all tokens moving to a state are summed up and, moreover, one token is

added to the initial state. Since the slave LTS is acyclic the number of tokens in each state is bounded.

Finally, the acceptance condition is $MP^{\bowtie p}_{\text{ext}}(r(\mathcal{R}))$ where the function $r(\mathcal{R})$ assigns to every state f the reward:

$$\sum_{\psi \in \mathsf{Sink}, \mathcal{R} \vdash \psi} f(\psi).$$

Each state thus has a reward that is the number of tokens in accepting sinks. Note that each token either causes a reward 1 once per its life-time when it reaches an accepting sink, or never causes any reward in the case when it never reaches any accepting state.

Lemma 4 (Correctness of slave automata). *Let $\xi \in \mathsf{sf}$, w, and $\underline{\mathcal{R}}, \overline{\mathcal{R}} \subseteq \mathcal{R}\text{ec}$ be such that $\underline{\mathcal{R}} \subseteq \mathcal{R}(w) \subseteq \overline{\mathcal{R}}$. Then*

$$w \in \mathsf{L}(\mathcal{S}_{\mathbf{GF}}(\xi, \underline{\mathcal{R}})) \implies w \models \mathbf{GF}\xi \implies w \in \mathsf{L}(\mathcal{S}_{\mathbf{GF}}(\xi, \overline{\mathcal{R}})) \tag{5}$$

$$w \in \mathsf{L}(\mathcal{S}_{\mathbf{FG}}(\xi, \underline{\mathcal{R}})) \implies w \models \mathbf{FG}\xi \implies w \in \mathsf{L}(\mathcal{S}_{\mathbf{FG}}(\xi, \overline{\mathcal{R}})) \tag{6}$$

$$w \in \mathsf{L}(\mathcal{S}_{\mathbf{G}^{\bowtie p}_{\text{ext}}}(\xi, \underline{\mathcal{R}})) \implies w \models \mathbf{G}^{\bowtie p}_{\text{ext}}\xi \implies w \in \mathsf{L}(\mathcal{S}_{\mathbf{G}^{\bowtie p}_{\text{ext}}}(\xi, \overline{\mathcal{R}})) \tag{7}$$

4.3 Product of slave automata

Observe that the LTS of slave automata never depend on the assumptions $\mathcal{R}$. Let $\mathcal{S}_1, \ldots, \mathcal{S}_n$ be the LTS of automata for elements of $\mathcal{R}\text{ec} = \{\xi_1, \ldots, \xi_n\}$. Further, given $\mathcal{R} \subseteq \mathcal{R}\text{ec}$, let $Acc_i(\mathcal{R})$ be the acceptance condition for the slave automaton for ξ_i with assumptions $\mathcal{R}$.

We define $\mathcal{P}$ to be the LTS product $\mathcal{S}_1 \times \cdots \times \mathcal{S}_n$. The slaves run independently in parallel. For $\mathcal{R} \subseteq \mathcal{R}\text{ec}$, we define the acceptance condition for the product[7]

$$Acc(\mathcal{R}) = \bigwedge_{\xi_i \in \mathcal{R}} Acc_i(\mathcal{R})$$

and $\mathcal{P}(\mathcal{R})$ denotes the LTS $\mathcal{P}$ endowed with the acceptance condition $Acc(\mathcal{R})$. Note that $Acc(\mathcal{R})$ checks that $\mathcal{R}$ is satisfied when each slave assumes $\mathcal{R}$.

Lemma 5 (Correctness of slave product). *For w and $\mathcal{R} \subseteq \mathcal{R}\text{ec}$, we have*

(soundness) *whenever $w \in \mathsf{L}(\mathcal{P}(\mathcal{R}))$ then $\mathcal{R} \subseteq \mathcal{R}(w)$;*
(completeness) *$w \in \mathsf{L}(\mathcal{P}(\mathcal{R}(w)))$.*

Intuitively, soundness means that whatever set of assumptions we prove with $\mathcal{P}$ it is also satisfied on the word. Note that the first line can be written as

$$w \in \mathsf{L}(\mathcal{P}(\mathcal{R})) \implies w \models \bigwedge_{\mathbf{F}\xi \in \mathcal{R}} \mathbf{GF}\xi \wedge \bigwedge_{\mathbf{G}\xi \in \mathcal{R}} \mathbf{FG}\xi \wedge \bigwedge_{\mathbf{G}^{\bowtie p}_{\text{ext}}\xi \in \mathcal{R}} \mathbf{G}^{\bowtie p}_{\text{ext}}\xi$$

Completeness means that for every word the set of all satisfied assumptions can be proven by the automaton.

[7] An acceptance condition of an automaton is defined to hold on a run of the automata product if it holds on the projection of the run to this automaton. We can still write this as a standard acceptance condition. Indeed, for instance, a Büchi condition for the first automaton given by $F \subseteq Q$ is a Büchi condition on the product given by $\{(q_1, q_2, \ldots, q_n) \mid q_1 \in F, q_2, \ldots, q_n \in Q\}$.

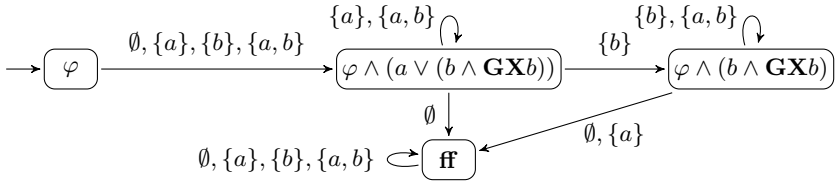
4.4 The final automaton: product of slaves and master

Finally, we define the generalized Rabin mean-payoff automaton $\mathcal{A}$ to have the LTS $\mathcal{M} \times \mathcal{P}$ and the acceptance condition $\bigvee_{\mathcal{R} \subseteq \mathcal{R}\mathrm{ec}} Acc_{\mathcal{M}}(\mathcal{R}) \wedge Acc(\mathcal{R})$ where

$$Acc_{\mathcal{M}}(\mathcal{R}) = Fin\left(\left\{ \left(\psi, (\Psi_\xi)_{\xi \in \mathcal{R}\mathrm{ec}}\right) \;\middle|\; \mathcal{R} \cup \bigcup_{\mathbf{G}\xi \in \mathcal{R}} \Psi_\xi[(\mathcal{R}\mathrm{ec} \setminus \mathcal{R})/\mathbf{ff}] \not\vdash \psi \right\}\right)$$

eventually prohibits states where the current formula of the master ψ is not proved by the assumptions and by all tokens of the slaves for $\mathbf{G}\xi \in \mathcal{R}$. Here $\Psi[X/\mathbf{ff}]$ denotes the set of formulae of Ψ where each element of X in the Boolean combination is replaced by $\mathbf{ff}$. For instance, $\{a \vee \mathbf{F}a\}[\{a\}/\mathbf{ff}] = \mathbf{ff} \vee \mathbf{F}a = \mathbf{F}a$. (For formal definition, see [20].) We illustrate how the information from the slaves in this form helps to decide whether the master formula holds or not.

Example 3. Consider $\varphi = \mathbf{G}(\mathbf{X}a \vee \mathbf{GX}b)$, and its respective master transition system as depicted below:



Assume we enter the second state and stay there forever, e.g., under words $\{a\}^\omega$ or $\{a, b\}^\omega$. How do we show that $\varphi \wedge (a \vee (b \wedge \mathbf{GX}b))$ holds? For the first conjunct, we obviously have $\mathcal{R} \vdash \varphi$ for all $\mathcal{R}$ containing φ. However, the second conjunct is more difficult to prove.

One option is that we have $\mathbf{GX}b \in \mathcal{R}$ and want to prove the second disjunct. To this end, we also need to prove b. We can see that if $\mathbf{GX}b$ holds then in its slave for $\mathbf{X}b$, there is always a token in the state b, which is eventually always guaranteed to hold. This illustrates why we need the tokens of the $\mathbf{G}$-slaves for proving the master formula.

The other option is that $\mathbf{GX}b$ is not in $\mathcal{R}$, and so we need to prove the first disjunct. However, from the slave for $\mathbf{G}(\mathbf{X}a \vee \mathbf{GX}b)$ we eventually always get only the tokens $\mathbf{X}a \vee \mathbf{GX}b$, $a \vee \mathbf{GX}b$, and $\mathbf{tt}$. None of them can prove $a \vee (b \wedge \mathbf{GX}b)$. However, since the slave does not rely on the assumption $\mathbf{GX}b$, we may safely assume it not to hold here. Therefore, we can substitute $\mathbf{ff}$ for $\mathbf{GX}b$ and after the substitution the tokens turn into $\mathbf{X}a$, a, and $\mathbf{tt}$. The second one is then trivially sufficient to prove the first disjunct.

Proposition 1 (Soundness). *If $w \in \mathsf{L}(\mathcal{A})$, then $w \models \varphi$.*

The key proof idea is that for the slaves of $\mathbf{G}$-formulae in $\mathcal{R}$, all the tokens eventually always hold true. Since also the assumptions hold true so does the conclusion ψ. By Lemma 2, φ holds true, too.

Proposition 2 (Completeness). *If $w \models \varphi$, then $w \in \mathsf{L}(\mathcal{A})$.*

$$\sum_{a \in A} x_{i,a} = 1 \qquad \text{for all } 1 \leq i \leq n \tag{8}$$

$$\sum_{a \in A} x_{i,a} \cdot \delta(a)(s) = \sum_{a \in Act(s)} x_{i,a} \qquad \text{for all } s \in S \text{ and } 1 \leq i \leq n \tag{9}$$

$$\sum_{s \in S, a \in Act(s)} x_{i,a} \cdot r_j(s) \bowtie v_j \qquad \text{for all } 1 \leq j \leq m \text{ and } 1 \leq i \leq n \tag{10}$$

$$\sum_{s \in S, a \in Act(s)} x_{i,a} \cdot q_i(s) \bowtie u_i \qquad \text{for all } 1 \leq i \leq n \tag{11}$$

Fig. 1: Linear constraints L of Proposition 3

The key idea is that subformulas generated in the master from **G**-formulae closely correspond to their slaves' tokens. Further, observe that for an **F**-formula χ, its unfolding is a disjunction of χ and other formulae. Therefore, it is sufficient to prove χ, which can be done directly from $\mathcal{R}$. Similarly, for $\mathbf{G}^{\bowtie p}_{\text{ext}}$-formula χ, its unfolding is just χ and is thus also provable directly from $\mathcal{R}$.

Complexity Since the number of Boolean functions over a set of size n is 2^{2^n}, the size of each automaton is bounded by $2^{2^{|\text{sf}|}}$, i.e., doubly exponential in the length of the formula. Their product is thus still doubly exponential. Finally, the acceptance condition is polynomial for each fixed $\mathcal{R} \subseteq \mathcal{R}\text{ec}$. Since the whole condition is a disjunction over all possible values of $\mathcal{R}$, it is exponential in the size of the formula, which finishes the proof of Theorem 2.

5 Verifying strongly connected MDPs against generalized Büchi mean-payoff automata

Theorem 3 can be obtained from the following proposition.

Proposition 3. *Let $\mathsf{M} = (S, A, Act, \delta, \hat{s})$ be a strongly connected MDP, and Acc an acceptance condition over S given by:*

$$\bigwedge_{i=1}^{k} Inf(S_i) \quad \wedge \quad \bigwedge_{i=1}^{m} MP^{\bowtie v_i}_{\inf}(r_i) \quad \wedge \quad \bigwedge_{i=1}^{n} MP^{\bowtie u_i}_{\sup}(q_i))$$

The constraints from Figure 1 have a non-negative solution if and only if there is a strategy σ and a set of runs R of non-zero probability such that Acc holds true on all $\omega \in R$. Moreover, σ and R can be chosen so that R has probability 1.

Intuitively, variables $x_{i,a}$ describe the frequencies of using action a. Equation (9) is Kirchhof's law of flow. Equation (10) says the inferior limits must be satisfied by all flows, while Equation (11) says that the ith limit superior has its own dedicated ith flow. Note that L does not dependent on the initial state $\hat{s}$.

Proof (Sketch). Existing results for multi-objective mean payoff MDPs would only allow to establish the proposition in absence of supremum limits, and so we need to extend and combine results of several works to prove the proposition. In the direction $\Rightarrow$, [13, Corollary 12] gives a strategy σ_i for every i such that for

almost every run $s_0 a_0 s_1 a_1 \ldots$ we have $\mathrm{lr}_{\inf}((\mathbb{1}_{a_t=a})_{t=0}^{\infty}) = x_{i,a}$, and in fact the corresponding limit exists. Hence, for the number $p = \sum_{s \in S, a \in Act(s)} r(s) \cdot x_{i,a}$ the predicates $MP_{\inf}^{\geq p}(r)$ and $MP_{\sup}^{\geq p}(r)$ almost surely holds, for any reward function r. Hence, our constraints ensure that σ_i satisfies $MP_{\inf}^{\bowtie v_j}(r_j)$ for all j, and $MP_{\sup}^{\bowtie u_i}(q_i)$. Moreover, σ_i is guaranteed to visit every state of M infinitely often almost surely. The strategy σ is then constructed to take these strategies $\sigma_i, 1 \leq i \leq n$ in turn and mimic each one of them for longer and longer periods.

For the direction $\Leftarrow$, we combine the ideas of [13, 9, 10] and select solutions to $x_{i,a}$ from "frequencies" of actions under the strategy σ.

6 Conclusions

We have given an algorithm for computing the optimal probability of satisfying an $\mathrm{fLTL}_{\setminus \mathbf{GU}}$ formula in an MDP. The proof relies on a decomposition of the formula into master and slave automata, and on solving a mean-payoff problem in a product MDP. The obvious next step is to extend the algorithm so that it can handle arbitrary formulae of fLTL. This appears to be a major task, since our present construction relies on acyclicity of slave LTS, a property which is not satisfied for unrestricted formulae [17]. Indeed, since $\mathbf{G}^{\bowtie p}$-slaves count the number of tokens in each state, this property ensures a bounded number of tokens and thus finiteness of the slave automata.

Acknowledgments. 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, by the EPSRC grant EP/M023656/1, by the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007–2013) REA Grant No 291734, by the Austrian Science Fund (FWF) S11407-N23 (RiSE/SHiNE), and by the ERC Start Grant (279307: Graph Games). Vojtěch Forejt is also affiliated with FI MU, Brno, Czech Republic.

References

1. S. Almagor, U. Boker, and O. Kupferman. Discounting in LTL. In *TACAS*, volume 8413 of *LNCS*. Springer, 2014.
2. C. Baier and J.-P. Katoen. *Principles of model checking*. MIT Press, 2008.
3. C. Baier, J. Klein, S. Klüppelholz, and S. Wunderlich. Weight monitoring with linear temporal logic: Complexity and decidability. In *CSL-LICS*. ACM, 2014.
4. R. Bloem, K. Chatterjee, T. A. Henzinger, and B. Jobstmann. Better quality in synthesis through quantitative objectives. In *CAV*. Springer, 2009.
5. R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.*, 78(3), 2012.
6. U. Boker, K. Chatterjee, T. A. Henzinger, and O. Kupferman. Temporal specifications with accumulative values. In *LICS*. IEEE, 2011.
7. B. Bollig, N. Decker, and M. Leucker. Frequency linear-time temporal logic. In *TASE*, Beijing, China, July 2012. IEEE.

8. P. Bouyer, N. Markey, and R. M. Matteplackel. Averaging in LTL. In *CONCUR*, volume 8704 of *LNCS*. Springer, 2014.
9. T. Brázdil, V. Brožek, K. Chatterjee, V. Forejt, and A. Kučera. Markov decision processes with multiple long-run average objectives. *LMCS*, 10(4), 2014.
10. T. Brázdil, K. Chatterjee, V. Forejt, and A. Kučera. Trading performance for stability in Markov decision processes. In *LICS*. IEEE, 2013.
11. K. Chatterjee and L. Doyen. Energy and mean-payoff parity Markov decision processes. In *MFCS*. Springer, 2011.
12. K. Chatterjee, A. Gaiser, and J. Křetínský. Automata with generalized Rabin pairs for probabilistic model checking and LTL synthesis. In *CAV*, 2013.
13. K. Chatterjee, Z. Komárková, and J. Křetínský. Unifying two views on multiple mean-payoff objectives in Markov decision processes. In *LICS*, 2015.
14. L. Clemente and J.-F. Raskin. Multidimensional beyond worst-case and almost-sure problems for mean-payoff objectives. In *LICS*. To appear, 2015.
15. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.
16. M. Droste and V. Perevoshchikov. Multi-weighted automata and mso logic. In *Comp. Sci. – Theory and Applications*, volume 7913 of *LNCS*. Springer, 2013.
17. J. Esparza and J. Křetínský. From LTL to deterministic automata: A safraless compositional approach. In *CAV*, 2014.
18. K. Etessami, M. Kwiatkowska, M. Vardi, and M. Yannakakis. Multi-objective model checking of Markov decision processes. *LMCS*, 4(4):1–21, 2008.
19. V. Forejt and J. Krčál. On frequency LTL in probabilistic systems. In *CONCUR*, volume 42 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.
20. V. Forejt, J. Krčál, and J. Křetínský. Controller synthesis for mdps and frequency $\text{LTL}_{\setminus \mathbf{GU}}$. *CoRR*, abs/1509.04116, 2015.
21. J. Kemeny, J. Snell, and A. Knapp. *Denumerable Markov Chains*. Springer, 2nd edition, 1976.
22. D. Kini and M. Viswanathan. Limit deterministic and probabilistic automata for LTL\GU. In *TACAS*, 2015.
23. J. Klein and C. Baier. Experiments with deterministic ω-automata for formulas of linear temporal logic. *Theor. Comput. Sci.*, 363(2), Oct. 2006.
24. Z. Komárková and J. Křetínský. Rabinizer 3: Safraless translation of LTL to small deterministic automata. In *ATVA*, volume 8837 of *LNCS*. Springer, 2014.
25. J. Křetínský and R. Ledesma-Garza. Rabinizer 2: Small deterministic automata for LTL\GU. In *ATVA*, 2013.
26. J. Křetínský and J. Esparza. Deterministic automata for the (F,G)-fragment of LTL. In *CAV*, 2012.
27. M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV 2011*, volume 6806 of *LNCS*. Springer, 2011.
28. M. Randour, J.-F. Raskin, and O. Sankur. Percentile queries in multi-dimensional Markov decision processes. In *CAV*. To appear, 2015.
29. W. Thomas. *Languages, automata, and logic*. Springer, 1997.
30. T. Tomita, S. Hagihara, and N. Yonezaki. A probabilistic temporal logic with frequency operators and its model checking. In *INFINITY*, 2011.
31. T. Tomita, S. Hiura, S. Hagihara, and N. Yonezaki. A temporal logic with mean-payoff constraints. In *ICFEM*, volume 7635 of *LNCS*. Springer, 2012.
32. M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structure versus Automata*, volume 1043 of *LNCS*. Springer, 1996.
33. M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *LICS*. IEEE, 1986.