Quantified Linear Temporal Logic over Probabilistic Systems with an Application to Vacuity Checking

Jakob Piribauer
Technische Universität Dresden, Germany

Christel Baier
Technische Universität Dresden, Germany

Nathalie Bertrand
Univ Rennes, Inria, CNRS, IRISA, France

Ocan Sankur
Univ Rennes, Inria, CNRS, IRISA, France

Abstract

Quantified linear temporal logic (QLTL) is an ω-regular extension of LTL allowing quantification over propositional variables. We study the model checking problem of QLTL-formulas over Markov chains and Markov decision processes (MDPs) with respect to the number of quantifier alternations of formulas in prenex normal form. For formulas with $k-1$ quantifier alternations, we prove that all qualitative and quantitative model checking problems are k-EXPSPACE-complete over Markov chains and $k+1$-EXPTIME-complete over MDPs.
As an application of these results, we generalize vacuity checking for LTL specifications from the non-probabilistic to the probabilistic setting. We show how to check whether an LTL-formula is affected by a subformula, and also study inherent vacuity for probabilistic systems.

2012 ACM Subject Classification Theory of computation → Verification by model checking

Keywords and phrases Quantified linear temporal logic, Markov chain, Markov decision process, vacuity

Funding This work was funded by DFG grant 389792660 as part of TRR 248 – CPEC (see https://perspicuous-computing.science), the Cluster of Excellence EXC 2050/1 (CeTI, project ID 390696704, as part of Germany's Excellence Strategy), DFG-projects BA-1679/11-1 and BA- 1679/12-1, and the Research Training Group QuantLA (GRK 1763).

1 Introduction

In the formal verification of probabilistic systems, a central problem is the *model-checking problem*: Given a system model $\mathcal{M}$ and a specification φ, decide whether the probability $\Pr_{\mathcal{M}}(\varphi)$ that φ holds on an execution of $\mathcal{M}$ is 1 or whether it is positive, respectively, (qualitative model checking) or compute the probability $\Pr_{\mathcal{M}}(\varphi)$ (quantitative model checking). In case the system exhibits non-deterministic behavior, the model-checking problems address the worst- or best-case resolution of the non-determinism, i.e., the minimal or maximal satisfaction probability among all possible resolutions of the non-deterministic choices. Common probabilistic system models are finite-sate Markov chains that are purely probabilistic and Markov decision processes (MDPs) that also model non-deterministic behavior. Specifications can be formulated in temporal logics, such as linear temporal logic (LTL) as an important example, or be given by automata, such as non-deterministic Büchi automata (NBA). The choice of the specification formalism is a balancing act between expressive power, succinctness, and the complexity of the respective model-checking problems. Additionally, the formalism should allow one to describe desired system behaviors in a way comprehensible to a human user as writing down the specification is itself an error-prone process in practice.
Quantified linear temporal logic (QLTL), introduced by Sistla [20], is an extension of LTL with quantification over propositional variables lifting the expressive power from star-free to all ω-regular languages. A formula of the form $\exists x.\varphi$ holds on a word w if one can choose a set of positions

at which x holds such that the word w extended with this choice satisfies φ. The quantification hence ranges over all sets of positions, i. e., sets of natural numbers. In QLTL, LTL-formulas can be extended with the quantification over propositions that, for example, capture hidden variables or encode annotations of a trace. This can be useful if we want to define properties not expressible in LTL in a context in which one often works with LTL. Examples include definitions of refinement relations in which internal variables are quantified to express equivalence of two specifications with respect to the observable variables [14], a necessary and sufficient condition expressed as a QLTL-formula on the serializability of histories in concurrent database scheduling produced by a scheduler whose behavior is expressed by an LTL-formula [13], or the QLTL-expressible existence of finite counterexamples witnessing the unrealizability of an LTL-specification for distributed fault-tolerant systems [8]. Furthermore, the vacuous satisfaction of a specification in a transitions system indicating that parts of the specification are irrelevant for the satisfaction has been defined using QLTL [1]. We transfer this definition of vacuous satisfaction to the probabilistic setting in this paper and explain the notion of vacuity in more detail below. In all of these successful applications of QLTL to questions in formal verification, the necessary QLTL-formulas require only few quantifier alternations; often even a single block of quantifiers without alternation is sufficient.

The full logic, however, is not suitable for practical applications: the non-probabilistic model-checking problem of QLTL on transition systems has non-elementary complexity [21]. The lower bounds can be pinpointed to the different levels of quantifier alternation of formulas in prenex normal form. Model-checking of QLTL-formulas with $k-1$ quantifier alternations in transition systems is k-EXPSPACE-complete. Distinguishing whether the first block of quantifiers is existential (Σ_k^{QLTL}) or universal (Π_k^{QLTL}) refines the result as for Π_k^{QLTL}-formulas the complexity of model-checking drops to $k-1$-EXPSPACE-completeness [21]. The increase of the complexity by one exponential per quantifier alternation is theoretically intriguing on the one hand, and on the other hand leads to reasonable complexity results for properties that can be expressed succinctly with the use of few quantifier alternations. A similar complexity hierarchy is observed in other settings. The complexity of model checking quantified computation tree logic (CTL) with k quantifier alternations is k-EXPTIME-complete, and it is $k+1$-EXPTIME-complete for quantified CTL* in the tree semantics; while in the structure semantics, these problems span the polynomial hierarchy [17]. The hardness of the fragments of QLTL [21] was used to show that model checking strategy logic is k-EXPSPACE-hard when restricted to k quantifier alternations.

In this paper, we study the model-checking problem of QLTL in probabilistic systems. Our main result is that the complexity of the model-checking problems on Markov chains and MDPs match the upper bounds obtained via straight-forward automata constructions: For Markov chains and Σ_k^{QLTL}- and Π_k^{QLTL}-formulas, all model-checking problems are k-EXPSPACE-complete, while for MDPs the problems are $k+1$-EXPTIME-complete. These complexity results are summarized in Table 1. As the upper bounds are easily obtained, the main contribution lies in proving the lower bounds. The hardness proofs for Markov chains, encode a tiling problem of a k-exponentially wide rectangle with arbitrary height. For the hardness proofs for MDPs, we encode the computation of an alternating k-exponentially space-bounded Turing machine. The alternation can be mimicked in an MDP by letting one player in the acceptance game of the alternating Turing machine be played randomly, while the scheduler takes the role of the other player. We obtain the result that the complexities of the model-checking problems for Σ_k^{QLTL} and Π_k^{QLTL} coincide in the probabilistic setting in contrast to the asymmetry known for the non-probabilistic setting. It is remarkable that the complexities of Σ_1^{QLTL}- and Π_1^{QLTL}-model checking in MDPs are the same as the complexity of LTL-model checking. For each further quantifier alternation, the complexity increases by one exponential. In contrast, we see an exponential increase in complexity already for the first block of quantifiers in Σ_1^{QLTL} and Π_1^{QLTL} compared to LTL-model checking in Markov chains.

	transition system	Markov chain	MDP
LTL	PSPACE [22, 24]	PSPACE [7]	2-EXPTIME [7]
Π_1^{QLTL}	PSPACE [21]	**EXPSPACE**	**2-EXPTIME**
Σ_1^{QLTL}	EXPSPACE [21]	**EXPSPACE**	**2-EXPTIME**
Π_k^{QLTL}	k-1-EXPSPACE [21]	k-**EXPSPACE**	k+1-**EXPTIME**
Σ_k^{QLTL}	k-EXPSPACE [21]	k-**EXPSPACE**	k+1-**EXPTIME**

Table 1 Complexity results for the model-checking problems of fragments of QLTL. All entries state completeness results.

On the one hand, knowledge of the precise complexities of the model-checking problems for Σ_k^{QLTL}- and Π_k^{QLTL}-formulas over probabilistic systems might be useful to determine the complexity of other problems in the formal verification of probabilistic systems – in particular, by using the new lower bounds provided in this paper for new hardness results. On the other hand, the upper bounds are obtained via the construction of automata. It follows easily that all investigated model-checking problems can be solved in time polynomial in the size of the model, i.e., the Markov chain or the MDP. This means that efficient model checking for low levels of the quantifier alternation hierarchy of QLTL might be possible in many application areas despite the high complexities of the model-checking problems because formulas are typically small compared to the size of the models.
As an application of our main results, we extend the definition of vacuous satisfaction of a specification from [1] to the probabilistic setting. For an illustration of vacuous satisfaction, consider the specification: “Whenever a request is sent, it is eventually granted.” If in a system model no requests are ever sent, the specification is satisfied and a model checker would report this result. However, something is obviously wrong with either the specification or – in this case more likely – the system model. We say the specification is vacuously true. The formal definition of vacuity that we generalize to the probabilistic setting captures the fact that the truth values of the grants in the specification do not influence the satisfaction of the specification at all. We could replace “it is eventually granted” with any arbitrary requirement or even choose an arbitrary set of positions at which that part of the specification should be true and the specification would still hold in the system model. We say that this subformula does not *affect* the satisfaction of the specification. Perturbing the truth values in arbitrary ways is expressed by a universal quantification over a proposition in the formal definition. A vacuity check during the model checking process can be an invaluable help as it can detect such severe errors in the design of the model or the specification in an early stage of the development that would otherwise stay undetected if the model checker returns the desired result.
We provide a generalization of the definition of affection that is suitable for the probabilistic setting. We prove that Π_1^{QLTL}-model checking is inter-reducible with the question whether a subformula affects a formula in a probabilistic system. Hence, an additional vacuity check according to this definition does not increase the complexity of model checking in MDPs. For Markov chains, however, the additional vacuity check would lead to an exponential blow-up of the procedure as shown by our new lower bound for Π_1^{QLTL}-model checking over Markov chains. Consequently, we turn our attention to the notion of inherent vacuity introduced in [9]. This notion captures that a specification is vacuous, i.e. not affected by some subformula, in all models. So, while disregarding the interplay between model and specification, inherent vacuity indicates a severe error in the specification. For

all natural variants of this definition for Markov chains and MDPs that can be obtained using our notion of affection, we obtain the result that inherent vacuity of a specification can be checked by a (non-probabilistic) validity check of a Π_1^{QLTL}-formula. Therefore, inherent vacuity for Markov chains and MDPs can be checked in polynomial space rendering the addition of a check for inherent vacuity to the model checking procedure potentially useful and reasonable in practice.

Related Work

Closest to our main complexity hierarchy result is the complexity hierarchy result for QLTL in the non-probabilistic setting [21]. Over probabilistic systems, the model-checking problems for Wolper's ETL [25], another ω-regular extension of LTL, which uses automata operators, is investigated in [7] and shown to lie in EXPTIME. We are not aware of any explicit investigations of QLTL or further ω-regular extensions of LTL, such as Gabbay's USF [10], an extension with fixed-point operators, over probabilistic systems.
Concerning vacuity checking, various notions have been studied for non-probabilistic systems. In [3] and [16], a notion of *formula vacuity* for fragments of CTL* is investigated in which the underlying notion of non-affection means that a subformula can be replaced by any other formula without affecting the truth of the formula in a model. *Trace vacuity* for LTL, which we generalize to the probabilistic setting, was introduced in [1]. The authors argue that trace vacuity has advantages over formula vacuity as it is more robust with respect to changes of the model or the specification language. Based on this definition, the notion of inherent vacuity, which we adapt to the probabilistic setting, was introduced in [9]. Trace vacuity has been extended to various other logics such as CTL* [11] relying on a propositionally quantified version of the logic, or to the logic RELTL, an extension of LTL with regular layers, by universally quantifying interval variables [4]. In [12], a variety of degrees to which a formula can be vacuous is defined and analyzed in the context of CTL-model checking. For a survey covering different approaches of vacuity checking, we refer the reader to [15].

2 Preliminaries

We suppose familiarity with basic concepts of discrete Markovian models, LTL, and ω-automata, and only provide a brief summary of the notions and our notation. Details can be found in textbooks, e.g., [2, 6, 19]. Furthermore, we provide definitions regarding QLTL and state basic results.

2.1 Basic definitions

Markov decision processes (MDPs)

An MDP is a tuple $\mathcal{M} = (S, \mathit{Act}, P, s_{init}, \mathsf{AP}, L)$ where S is a finite state space, *Act* a finite set of actions, $P : S \times \mathit{Act} \times S \to [0,1] \cap \mathbb{Q}$ the transition probability function satisfying $\sum_{t \in S} P(s, \alpha, t) \in \{0,1\}$ for all $(s, \alpha) \in S \times \mathit{Act}$, $s_{init} \in S$ the initial state, AP a finite set of atomic propositions, and $L : S \to 2^{\mathsf{AP}}$ a labeling function. The triples (s, α, t) with $P(s, \alpha, t) > 0$ are called transitions of $\mathcal{M}$. The actions enabled in s form $\mathit{Act}(s) = \{\alpha \in \mathit{Act} : \sum_{t \in S} P(s, \alpha, t) = 1\}$. The *size* of an MDP is the number of states and actions plus the sum of the logarithmic lengths of the transition probabilities. Intuitively, when $\mathcal{M}$ is at a state s, then an action α of $\mathit{Act}(s)$ is selected nondeterministically; afterwards the next state is obtained by probabilistically choosing one of the potential successor states according to the probability distribution $P(s, \alpha, \cdot)$. Paths in MDP are alternating sequences of states and actions: $\pi = s_0\, \alpha_0\, s_1\, \alpha_1 \ldots$ where $\alpha_i \in \mathit{Act}(s_i)$ and $P(s_i, \alpha_i, s_{i+1}) > 0$ for all $i \geqslant 0$. We write $\pi_{[i\ldots]}$ for the suffix starting from s_i. The *trace* of π is the word $L(\pi) = L(s_0)\, L(s_1)\, L(s_2) \ldots$ over 2^{AP} obtained by projecting states to their labels. We do not distinguish between a path and its trace when the intended

meaning is clear from context. A *scheduler* for $\mathcal{M}$ is a function $\mathfrak{S}$ that maps a finite path ζ to a probability distribution over $Act(last(\zeta))$ where $last(\zeta)$ is the last state of ζ. The function $\Pr^{\mathfrak{S}}_{\mathcal{M},s}$ denotes the probability measure induced by $\mathfrak{S}$, when s is the initial state. It is well-known that all ω-regular path properties φ are measurable and there exist schedulers maximizing or minimizing the probability for φ (see, e.g., [2]). This justifies the notations $\Pr^{\max}_{\mathcal{M},s}(\varphi) = \max_{\mathfrak{S}} \Pr^{\mathfrak{S}}_{\mathcal{M},s}(\varphi)$ and analogously $\Pr^{\min}_{\mathcal{M},s}(\varphi)$ for ω-regular properties.
A *Markov chain* is a tuple $\mathcal{M} = (S, P, s_{init}, \mathsf{AP}, L)$ which can be seen as an MDP with only one action. The transition probability function $P : S \times S \to [0,1] \cap \mathbb{Q}$ does not include the action anymore and satisfies $\sum_{t \in S} P(s,t) \in \{0,1\}$ for all $s \in S$. There are no non-deterministic choices and $\Pr_{\mathcal{M},s}$ denotes the induced probability measure on maximal paths starting in s.

ω-automata

A *non-deterministic Büchi automaton (NBA)* is a tuple $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ where Q is a finite set of states, Σ an alphabet, $\delta \subseteq S \times \Sigma \times S$ the transition relation, $Q_0 \subseteq Q$ the set of initial states and $F \subseteq Q$ the set of final states. A word $w = w_0 w_1 \dots$ in Σ^ω is accepted by $\mathcal{A}$ if there is a run $q_0 w_0 q_1 w_1 q_2 \dots$ such that $q_0 \in Q_0$, $(q_i, w_i, q_{i+1}) \in \delta$ for all i, and for infinitely many i, $q_i \in F$. The language $\mathcal{L}(\mathcal{A})$ is the set of words accepted by $\mathcal{A}$.

2.2 Quantified linear temporal logic (QLTL)

Let AP be a finite set of atomic propositions. The syntax of linear temporal logic (LTL) is given by

$$\varphi ::= a \mid \varphi \wedge \varphi \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi \,\mathrm{U}\, \varphi$$

where $a \in \mathsf{AP}$. The semantics is given on words in $(2^{\mathsf{AP}})^\omega$: For a word $w = w_0, w_1, \dots$, we have $w \vDash a$ if $a \in w_0$; $w \vDash \bigcirc\varphi$ if $w_1, w_2, \dots \vDash \varphi$; and $w \vDash \varphi \,\mathrm{U}\, \psi$ if there is a $j \in \mathbb{N}$ such that $w_j, w_{j+1}, \dots \vDash \psi$ and $w_i, w_{i+1}, \dots \vDash \varphi$ for all $i < j$. The semantics of the Boolean connectives is defined as usual. For more details, consult, e.g., [2]. The logic QLTL is an extension of LTL with quantification over atomic propositions. We extend the syntax of LTL by allowing existential quantification $\exists x.\varphi$ over additional, fresh atomic propositions $x \notin \mathsf{AP}$ where φ is an LTL-formula over $\mathsf{AP} \cup \{x\}$. We further allow the common abbreviations $\top$ for true, $\bot$ for false, $\vee$, $\to$, $\leftrightarrow$, $\Diamond$, $\Box$, and $\forall x$. For a word $w \in (2^{\mathsf{AP}})^\omega$, we define that $w \vDash \exists x.\varphi$ if and only if there is a set $X \subseteq \mathbb{N}$ such that the word w' with $w'[i] = w[i]$ if $i \notin X$ and $w'[i] = w[i] \cup \{x\}$ if $i \in X$ satisfies $w' \vDash \varphi$. Consider the following example to illustrate the semantics of QLTL:

$$\begin{array}{llllll} \{a\} & \{b\} & \{a\} & \{c\} & \dots & \vDash \exists x.\Box(x \leftrightarrow \neg a) \text{ because} \\ \{a\} & \{b,x\} & \{a\} & \{c,x\} & \dots & \vDash \Box(x \leftrightarrow \neg a). \end{array}$$

For a QLTL-formula ϑ over AP, we allow arbitrarily many additional atomic propositions but require that all atomic propositions not in AP are quantified. We distinguish QLTL-formulas in prenex normal form according to the number of quantifier alternations. For $k \geqslant 1$, let Σ_k^{QLTL} be the set of QLTL-formulas of the form

$$\underbrace{\exists^*\forall^*\exists^*\dots}_{k \text{ blocks of quantifiers}} \varphi \equiv \underbrace{\exists^*\neg\exists^*\neg\exists^*\dots}_{k \text{ blocks of quantifiers}} (\neg)\varphi$$

where φ is quantifier-free, i.e. an LTL-formula. Likewise, let Π_k^{QLTL} be the set of QLTL-formulas starting with k blocks of quantifiers followed by a quantifier-free formula such that the first block is $\forall^*$. The negation of a Σ_k^{QLTL}-formula is equivalent to a Π_k^{QLTL}-formula.
QLTL, and in particular Σ_1^{QLTL}, can express exactly all ω-regular properties. In fact, the existence of an accepting run on a word in an NBA $\mathcal{A}$ with states Q can be expressed by a Σ_1^{QLTL}-formula with $|Q|$-many existential quantifications followed by an LTL-formula [20].

Conversely, for a Σ_k^{QLTL}-formula $\vartheta = \exists^*\neg\exists^*\ldots(\neg)\varphi$, we can build an NBA of k-exponential size accepting exactly the words satisfying ϑ: For the LTL-part $(\neg)\varphi$, we first construct an NBA of exponential size (see [24]). Existential quantification on the NBA-level is easy as it corresponds to standard projection onto the non-quantified variables; a quantified variable x is simply removed from the labels of the transition relation. This introduces new non-deterministic choices between the options to take a transition requiring a letter, i.e. a set of atomic propositions, P or a transition requiring $P \cup \{x\}$ when reading P. The quantifier prefix contains $k-1$ negations in addition to the existential quantifiers. Each of these negations requires a complementation of the automaton constructed so far before we can use projection again to account for the next block of quantifiers. Each complementation increases the size by one further exponential. Hence, the procedure produces an NBA for ϑ of k-exponential size in k-exponential time (see [21] for more details).

3 QLTL model checking in probabilistic systems

This section is devoted to proving the complexity hierarchy results in terms of the quantifier alternation for the model-checking problem of QLTL in probabilistic systems. More precisely, our goal is to pinpoint the complexities of the following problems, for Π_k^{QLTL}- or Σ_k^{QLTL}-formulas φ:

- Qualitative model-checking problems:
  - For a Markov chain $\mathcal{M}$, decide whether $\Pr_{\mathcal{M},s_{init}}(\varphi) = 1$, or whether $\Pr_{\mathcal{M},s_{init}}(\varphi) > 0$, respectively.
  - For an MDP $\mathcal{M}$, decide whether $\Pr^{\max}_{\mathcal{M},s_{init}}(\varphi) = 1$, whether $\Pr^{\max}_{\mathcal{M},s_{init}}(\varphi) > 0$, whether $\Pr^{\min}_{\mathcal{M},s_{init}}(\varphi) = 1$, or whether $\Pr^{\min}_{\mathcal{M},s_{init}}(\varphi) > 0$, respectively.
- Quantitative model-checking problems:
  - For a Markov chain $\mathcal{M}$, compute $\Pr_{\mathcal{M},s_{init}}(\varphi)$. For hardness results, we consider the decision versions whether $\Pr_{\mathcal{M},s_{init}}(\varphi) \bowtie \vartheta$ for a given $\vartheta \in \mathbb{Q}$ and $\bowtie \in \{\leqslant, <, >, \geqslant\}$.
  - For an MDP $\mathcal{M}$, compute $\Pr^{\mathrm{opt}}_{\mathcal{M},s_{init}}(\varphi)$ for $\mathrm{opt} \in \{\max, \min\}$. For hardness results, we consider the decision versions whether $\Pr^{\mathrm{opt}}_{\mathcal{M},s_{init}}(\varphi) \bowtie \vartheta$ for a given $\vartheta \in \mathbb{Q}$, $\bowtie \in \{\leqslant, <, >, \geqslant\}$, and $\mathrm{opt} \in \{\max, \min\}$.

We restrict our attention to QLTL-formulas in prenex normal form. While we have seen that all QLTL-formulas are equivalent to a Σ_1^{QLTL}-formula, the transformation from arbitrary QLTL-formulas to Σ_1^{QLTL}-formulas has non-elementary complexity. The lower bound for this transformation is a direct consequence of the complexity hierarchy result for the non-probabilistic model-checking problem mentioned above. However, there is a polynomial-time transformation to prenex normal form for QLTL-formulas: After renaming all quantified variables such that each quantifier quantifies a unique variable not occurring outside the scope of this quantifier, we can pull out quantifiers using the following equivalences for arbitrary QLTL-formula φ and ψ where ψ does not contain the atomic proposition x and both formulas do not contain t:

1. $(\exists x\varphi)\,\mathrm{U}\,\psi \equiv \forall t\exists x((t\,\mathrm{U}(\neg t \wedge \varphi)) \vee (t\,\mathrm{U}\,\psi))$.
2. $(\forall x\varphi)\,\mathrm{U}\,\psi \equiv \forall x(\varphi\,\mathrm{U}\,\psi)$.
3. $\psi\,\mathrm{U}(\exists x\varphi) \equiv \exists x(\psi\,\mathrm{U}\,\varphi)$.
4. $\psi\,\mathrm{U}(\forall x\varphi) \equiv \exists t\forall x((\psi \wedge t)\,\mathrm{U}(\varphi \wedge \neg t))$.

Note that in the first and last equivalence where t is quantified, only the first position where $\neg t$ holds is important for the subsequent formulas. In this way, the quantification over t corresponds to the quantification over positions in the semantics of the U-operator. For $Q \in \{\exists, \forall\}$, we further have $\bigcirc Qx\varphi \equiv Qx\bigcirc\varphi$ and moving quantifiers to the front over Boolean connectives can be done

as usual. So, we can transform a QLTL-formula to prenex normal form in polynomial time while introducing new quantifiers to account for the implicit quantification over positions of the U-operator. In applications of QLTL in formal verification, however, quantified variables are mostly used to describe possible annotations of a trace or traces of hidden variables. Hence, the quantified traces are supposed to be constant once chosen and not to be reassigned when evaluating subformulas on different suffixes. Thus, these formulas often are already in prenex normal form.
Our main result concerning QLTL-model checking over probabilistic systems is the following complexity hierarchy result:

▶ **Theorem 1** (Main Result). *All qualitative and quantitative model-checking problems for Σ_k^{QLTL} and Π_k^{QLTL} in Markov chains are k-EXPSPACE-complete and can be solved in time polynomial in the size of the Markov chain.*
All qualitative and quantitative model-checking problems for Σ_k^{QLTL} and Π_k^{QLTL} with $k \geqslant 1$ in MDPs are $k+1$-EXPTIME-complete and can be solved in time polynomial in the size of the MDP.

The upper bounds are obtained by the straight-forward construction of NBAs as described above (Section 2.2). The main contribution hence is the proof of the lower bounds. For Markov chains, we provide a reduction from a tiling problem that simultaneously shows hardness for all qualitative model-checking problems (Theorem 2). We afterwards conclude the same complexity result for all quantitative model-checking problems (Corollary 3). For MDPs, the result requires two different hardness proofs (Theorem 4): The hardness results for model-checking problems regarding the maximal satisfaction probability of Π_k^{QLTL}-formulas (or analogously the minimal satisfaction probability of Σ_k^{QLTL}-formulas) are somewhat simpler. We encode computations of an alternating Turing machine that is k-exponentially space bounded and can directly use sequences of k-exponentially many extended tape symbols for the encoding. For the hardness proof concerning the minimal satisfaction probability of Π_k^{QLTL}-formulas, we have to include a binary counter of $k-1$-exponential length separating two successive tape symbols in the encoding. In the hardness proof for Markov chains, we use a similar counter. So, the final hardness proof combines the ideas behind the first hardness proof for MDPs and the hardness proof for Markov chains. The same complexity results for all quantitative model-checking problems in MDPs can be concluded afterwards (Corollary 5).

3.1 Markov chains

We first address the qualitative model-checking problems in Markov chains. We provide a proof sketch for the hardness proof. The full proof can be found in Appendix A.

▶ **Theorem 2.** *For any k, all qualitative model-checking problems for Σ_k^{QLTL} and Π_k^{QLTL} in Markov chains are k-EXPSPACE-complete and can be solved in time polynomial in the size of the Markov chain.*

Proof sketch. The upper bounds are obtained by building NBAs of k-exponential size for Σ_k^{QLTL}-formulas as described in Section 2. The negation of a Π_k^{QLTL}-formula is equivalent to a Σ_k^{QLTL}-formula of the same length. As all qualitative model-checking problems for NBAs in Markov chains are PSPACE-complete and can be solved in time polynomial in the size of the Markov chain [7], we obtain the upper bounds.
For the hardness results, we use a reduction from k-exponential tiling problems. We define the following function $h\colon \mathbb{N}^2 \to \mathbb{N}$: Let $h(0,n) = n$ for all n and $h(k+1,n) = 2^{h(k,n)} \cdot h(k,n)$ for all k. So, $h(k,n)$ is k-exponential in n. The following k-exponential tiling problem is known to be k-EXPSPACE-complete [23]:
Given: a finite set of tiles T, two relations $H \subseteq T^2$ and $V \subseteq T^2$, an initial tile $t_0 \in T$ and a final tile $t_f \in T$ as well as a natural number n in unary.

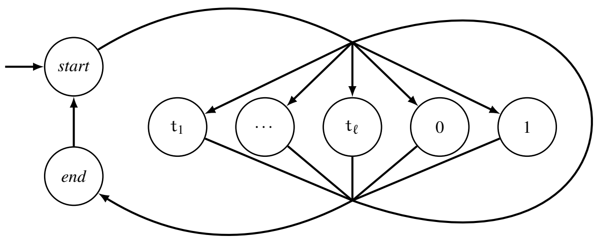


Figure 1 The Markov chain $\mathcal{M}$.

Question: Is there an $m \in \mathbb{N}$ such that the $2^{h(k-1,n)} \times (m+1)$-grid $\{0,\dots,2^{h(k-1,n)}-1\} \times \{0,\dots,m\}$ can be tiled, i. e., is there a function $f\colon \{0,\dots,2^{h(k-1,n)}-1\} \times \{0,\dots,m\} \to T$, such that:

1. the tile at position $(0,0)$ is the initial tile t_0 and the tile at position $(0,m)$ is the final tile t_f; in other words, $f(0,0) = t_0$ and $f(0,m) = t_f$,
2. two tiles placed next to each other horizontally satisfy the relation H; more precisely, for any $0 \leqslant i < 2^{h(k-1,n)}-1$ and $0 \leqslant j \leqslant m$, the pair $(f(i,j), f(i+1,j)) \in H$, and
3. two tiles placed next to each other vertically satisfy the relation V; more precisely, for any $0 \leqslant i \leqslant 2^{h(k-1,n)}-1$ and $0 \leqslant j < m$, the pair $(f(i,j), f(i,j+1)) \in V$?

Given an instance of the k-exponential tiling problem, we construct a Markov chain $\mathcal{M}$ and a ψ in Π_k^{QLTL} such that $\Pr_{\mathcal{M}}(\psi) = 1$ iff $\Pr_{\mathcal{M}}(\psi) > 0$ iff there is a valid tiling. This establishes k-EXPSPACE-hardness for both qualitative model checking problems for Π_k^{QLTL}. As the negation of ψ is in Σ_k^{QLTL} and k-EXPSPACE is closed under complementation, the same result holds for Σ_k^{QLTL}. Let $T = \{t_0,\dots,t_\ell\}$ be the set of tiles that we also use as atomic propositions and let $\{\textit{start}, \textit{end}, 0, 1\}$ be further atomic propositions. We construct a simple Markov chain $\mathcal{M}$, depicted in Figure 1, that almost surely produces a concatenation of infinitely many words from $\textit{start}(T \cup \{0,1\})^+\textit{end}$ that contains each of the finite words in $\textit{start}(T \cup \{0,1\})^+\textit{end}$. Some of these finite words will encode potential tilings. Namely, we encode a function $f\colon \{0,\dots,2^{h(k-1,n)}-1\} \times \{0,\dots,m\} \to T$ in the word

$$\begin{aligned}\textit{start},\ & f(0,0),\ \overbrace{0,0,0,\dots,0}^{h(k-1,n)\text{ steps}},f(1,0),\ \overbrace{1,0,0,\dots,0}^{h(k-1,n)\text{ steps}},\dots,f(2^{h(k-1,n)}-1,0),\ \overbrace{1,1,1,\dots,1}^{h(k-1,n)\text{ steps}},\\ & f(0,1),\ \dots,\\ & f(0,m),\ \overbrace{0,0,0,\dots,0}^{h(k-1,n)\text{ steps}},f(1,m),\ \overbrace{1,0,0,\dots,0}^{h(k-1,n)\text{ steps}},\dots,f(2^{h(k-1,n)}-1,m),\ \overbrace{1,1,1,\dots,1}^{h(k-1,n)\text{ steps}},\textit{end}.\end{aligned}$$

For a valid encoding, the blocks of $h(k-1,n)$ bits have to encode a correct binary counter modulo $2^{h(k-1,n)}$, where the first bit is the least significant one, starting with $0\dots0$ after *start* and ending in $1\dots1$ before *end*. The encoding of the counter makes sure that indeed a function from a rectangle $\{0,\dots,2^{h(k-1,n)}-1\} \times \{0,\dots,m\}$ for some m is encoded.
Further, we construct a Π_k^{QLTL}-formula *valid_tiling* that expresses that at some point a valid tiling is encoded in a run. Several of the conditions including the initial, final and horizontal condition can easily be expressed. As tiles that are vertically adjacent in a tiling are separated by $h(k,n) = h(k-1,n) \cdot 2^{h(k-1,n)}$ steps, however, we have to employ additional ideas to express that all conditions on a valid encoding of a valid tiling are satisfied at some point. An important ingredient for our reduction is the collection of Σ_{k-1}^{QLTL}-formulas $\varphi_{k-1,n}(p,q)$ from [21]. For each n and k from $\mathbb{N}$, the formula $\varphi_{k-1,n}(p,q)$ holds on a word if p and q occur exactly once and, if the position at which p occurs is

i, the position at which q occurs is $i+h(k-1,n)$. In addition to the use of these formulas, we use universally quantified propositions that mark potential violations of the conditions. To illustrate this idea, we sketch a formula that expresses that a run of $\mathcal{M}$ eventually contains a finite word starting with *start* and ending in *end* in which tiles are followed by exactly $h(k-1,n)$-many bits. The atomic proposition *tile* holds if the current letter encodes a tile.

$$\forall d. \Big(\big[\forall p \forall q \big(\varphi_{k-1,n}(p,q) \to \Box[(d \wedge \mathit{tile} \wedge p) \to \\ \text{next occurrence of } \mathit{tile} \text{ or } \mathit{end} \text{ not one step after } q]\big)\big] \to \Diamond(\mathit{start} \wedge (\neg(d \,\mathrm{U}\, \mathit{end}))\Big).$$

The quantified proposition d can be used to mark any tiles for which the next tile or *end* does not follow exactly $h(k-1,n)+1$ steps later. The quantified variables p and q and the formula $\varphi_{k-1,n}(p,q)$ are used to check that the markers are placed correctly, i.e., that indeed the next occurrence of *tile* or *end* after the marked position is not exactly $h(k-1,n)+1$ steps later. If the markers d are not placed correctly, the formula holds. Otherwise, it holds if a finite word between *start* and *end* is contained in the run in which no tile is marked by d. As d is universally quantified, the formula hence holds on a run of $\mathcal{M}$ iff it contains a finite word starting with *start* and ending in *end* in which tiles are followed by exactly $h(k-1,n)$-many bits. Note that $\varphi_{k-1,n}(p,q)$ occurs in the scope of two negations due to the implications while $\forall p \forall q$ occurs in the scope of one negation. So, the formula is in Π_k^{QLTL}. The correctness of the counter can be expressed using the same idea of marking bits that violate the correctness of the counter with a universally quantified variable and the fact that a bit in a binary counter changes during an increment of the counter if and only if all less significant bits are 1. The vertical condition of the tiling is checked by using universally quantified markers v_1 and v_2 that have to be placed on vertically adjacent tiles. The correct placement of the markers is checked by stating that there exists a proposition b that encodes a correct binary counter with $h(k-1,n)$-many bits that starts with $0\dots0$ after v_1 and counts up to $1\dots1$ right before v_2. The correctness of the counter is checked as for the counter using the bits 0 and 1. The additional existential quantification over b does not yield an additional quantifier alternation. The resulting formula *valid_tiling* is in Π_k^{QLTL} and holds on a run of $\mathcal{M}$ if an encoding of a valid tiling is produced. As a run of $\mathcal{M}$ almost surely contains all words in $\mathit{start}(T \cup \{0,1\})^+\mathit{end}$, the formula *valid_tiling* holds with probability 1 iff it holds with positive probability iff there is a valid tiling for the given instance of the k-exponential tiling problem. ◀

As the upper bounds are obtained via the construction of NBAs for the QLTL-formulas, we can conclude the same results for the quantitative model-checking problems over Markov chains.

► **Corollary 3** (Quantitative model checking). *Given a Σ_k^{QLTL}- or Π_k^{QLTL}-formula φ and a Markov chain $\mathcal{M}$, the probability $\Pr_{\mathcal{M}}(\varphi)$ can be computed in k-exponential space and in time polynomial in the size of $\mathcal{M}$. Given a rational $\vartheta \in [0,1]$ and $\bowtie \in \{\leqslant, <, >, \geqslant\}$, deciding whether $\Pr_{\mathcal{M}}(\varphi) \bowtie \vartheta$ is k-EXPSPACE-complete.*

Proof. The lower bounds follow directly from the previous theorem. The upper bound follows from the fact that, given a Markov chain $\mathcal{M}$ and an NBA $\mathcal{A}$, the probability $\Pr_{\mathcal{M}}(\mathcal{A})$ that a word produced by $\mathcal{M}$ is accepted by $\mathcal{A}$ can be computed in time polynomial in $\mathcal{M}$ [7] and in space polynomial in the total size of the input. We sketch a proof of the latter claim: In the algorithm provided by Courcoubetis and Yannakakis in [7] to compute this probability, an exponentially large Markov chain $\mathcal{N}$ is constructed from $\mathcal{M}$ and $\mathcal{A}$. The states of $\mathcal{N}$ have a polynomial representation in the size of $\mathcal{M}$ and $\mathcal{A}$ and one can compute the transition probabilities between any two states in polynomial time. The probability $\Pr_{\mathcal{M}}(\mathcal{A})$ now equals the probability to reach a *recurrent* state in $\mathcal{N}$ – as it is called in [7], but which we do not define here. It is only important to us that one can decide whether a state is recurrent in polynomial space polynomial in the size of $\mathcal{A}$ (and polylogarithmic in the size of $\mathcal{M}$)

as shown in [7]. The probability to reach a recurrent state in $\mathcal{N}$ can be computed by solving a linear equation system. As transition probabilities and whether states are recurrent in $\mathcal{N}$ can be computed in space polynomial in $\mathcal{A}$, each entry of the matrix and vector representing this linear equation system, which is of size exponential in $\mathcal{A}$ and polynomial in $\mathcal{M}$, can be computed in space polynomial in $\mathcal{A}$. Using the fact that solving linear equation systems lies in the complexity class NC and can hence be done in polylogarithmic space (see, e.g., [18, Section 15]) and standard results on the composition of space-bounded transductions (see, e.g., [18, Section 8]), we can conclude that the probability $\Pr_{\mathcal{M}}(\mathcal{A})$ can be computed in space polynomial in the size of $\mathcal{A}$. Applied to the k-exponentially sized NBAs for Σ_k^{QLTL}-formulas, this result leads to the claim of the corollary. ◀

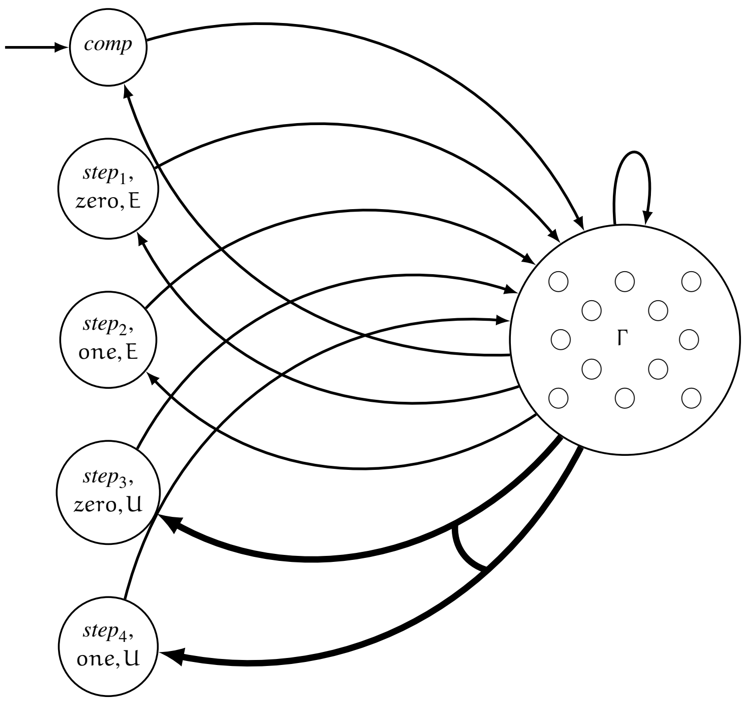
3.2 Markov decision processes

We now provide the complexity results for QLTL-model checking over MDPs.

▶ **Theorem 4.** *Given an MDP $\mathcal{M}$, a Π_k^{QLTL}-formula φ, and $\mathrm{opt} \in \{\max, \min\}$, deciding whether $\Pr_{\mathcal{M}}^{\mathrm{opt}}(\varphi) = 1$ and deciding whether $\Pr_{\mathcal{M}}^{\mathrm{opt}}(\varphi) > 0$ are $k+1$-EXPTIME-complete for any $k \geqslant 1$. The problems are solvable in time polynomial in the size of $\mathcal{M}$.*

As Π_k^{QLTL} is not closed under negation, the model-checking problems in MDPs concerning the maximal and minimal satisfaction probability, respectively, require different hardness proofs. We sketch the two proof ideas in the sequel. The full proofs can be found in Appendix B.

Proof sketch. The upper bounds are obtained via the straight-forward construction of *deterministic* automata (e.g., deterministic Rabin automata; see, e.g., [2]). This requires the determinization of the k-exponentially large NBAs for Σ_k^{QLTL}-formulas, which are computable in k-exponential time, and leads to a $k+1$-exponential-time procedure.
For the lower bounds, first consider the problems with $\mathrm{opt} = \max$. We prove $k+1$-EXPTIME-hardness by encoding the computation of k-exponentially space-bounded alternating Turing machines (ATM). It is well-known that the class of problems decidable by such ATMs coincides with $k+1$-EXPTIME [5]. So, given a k-exponentially space-bounded ATM $\mathcal{T}$ and an input word w, we construct an MDP $\mathcal{M}$ and a Π_k^{QLTL}-formula φ such that $\Pr_{\mathcal{M}}^{\max}(\varphi) > 0$ iff $\Pr_{\mathcal{M}}^{\max}(\varphi) = 1$ iff w is accepted by $\mathcal{T}$. Recall that acceptance in an ATM can be specified in terms of a game between a universal player choosing the next move in universal states and an existential player choosing the next move in existential state. A word is accepted if the existential player has a strategy that ensures that an accepting state is reached from the initial configuration with the input word on the tape.
The idea for the reduction is to construct an MDP $\mathcal{M}$ in which the scheduler can produce a sequence of (k-exponentially long) configurations of $\mathcal{T}$. The sequence of configurations in turn represents a sequence of infinitely many finite computations. The first configuration of each computation has to be the initial configuration with w on the tape. After each configuration, the scheduler has to specify whether the universal or existential player has to choose the next move, or whether the computation ended and a new computation is about to start. If it is the existential player's turn, the scheduler chooses a move and has to construct the successor configuration accordingly. If it is the universal player's turn, the successor move is specified by a random choice and again the scheduler has to construct the correct successor configuration. The constructed MDP is sketched in Figure 2.
The Π_k^{QLTL}-formula φ we construct, on the one hand, expresses that the sequence produced by the scheduler obeys all these requirements. Checking that the successor configurations are constructed correctly is possible with the use of the Σ_k^{QLTL}-formulas $\varphi_{k,n}(p,q)$ from [21] that express that the positions at which p and q are a fixed k-exponentially large number of steps apart. On the other hand, the formula φ expresses that all (infinitely many) encoded computations end in an accepting state. If w is accepted by $\mathcal{T}$, the scheduler can construct correct accepting computations no matter what moves



comp
step$_1$, $\mathsf{zero}, \mathsf{E}$
step$_2$, one, E
step$_3$, $\mathsf{zero}, \mathsf{U}$
step$_4$, one, U
Γ

Figure 2 The MDP $\mathcal{M}$. The state depicted as Γ represents the behavior of each state $\gamma \in \Gamma$. I.e. from each state, there is one action to each state in Γ with probability 1. Further, from all states in Γ, there are actions leading to states *comp*, *step*$_1$ and *step*$_2$ with probability 1, as well as a randomized action (bold lines) leading to states *step*$_3$ and *step*$_4$ with probability $1/2$ each. The labels zero and one indicate which successor move was chosen according to which the new configuration has to be constructed. The labels E and U indicate which player has chosen the move and are used to check whether the successor move was indeed chosen randomly iff it is the universal player's move.

are chosen by the universal player and so $\Pr^{\max}_{\mathcal{M}}(\varphi) = 1$. If w is not accepted, however, the universal player will play according to a winning strategy in any of the encoded computations with positive probability. So, almost surely at some point any scheduler has to violate one of the requirements or construct a rejecting computation. In this case, $\Pr^{\max}_{\mathcal{M}}(\varphi) = 0$.
In contrast to the case just discussed, the statement $\Pr^{\min}_{\mathcal{M}}(\varphi) = 1$ is a statement about all schedulers. So, we cannot let a scheduler construct sequences of computations anymore. Instead, we construct an MDP $\mathcal{M}'$ that randomly generates sequences that potentially encode correct computations. In the acceptance game of the given ATM, we also switch roles and let the choices of the existential player be made randomly while the scheduler can specify which successor move should be chosen in universal states. With positive probability, the correct successor configuration will be generated afterwards. Hence, if the existential player has a winning strategy, a correct accepting computation will eventually be produced randomly with probability 1 no matter what successor moves a scheduler chooses. Otherwise, there is a scheduler that prohibits this.
In order to express that eventually a correct accepting computation is generated in Π_k^{QLTL}, however, it turns out that we cannot use the Σ_k^{QLTL}-formulas $\varphi_{k,n}(p,q)$ from [21] as before. This is in part due to the implicit existential quantification in the eventually-modality. For this reason, we do not encode the computations simply as concatenations of configurations. Instead, we employ the ideas that were also used in the hardness proof for Markov chains (Theorem 2): We separate the symbols of the configurations by $k-1$-exponentially long binary counters to check that configurations have the

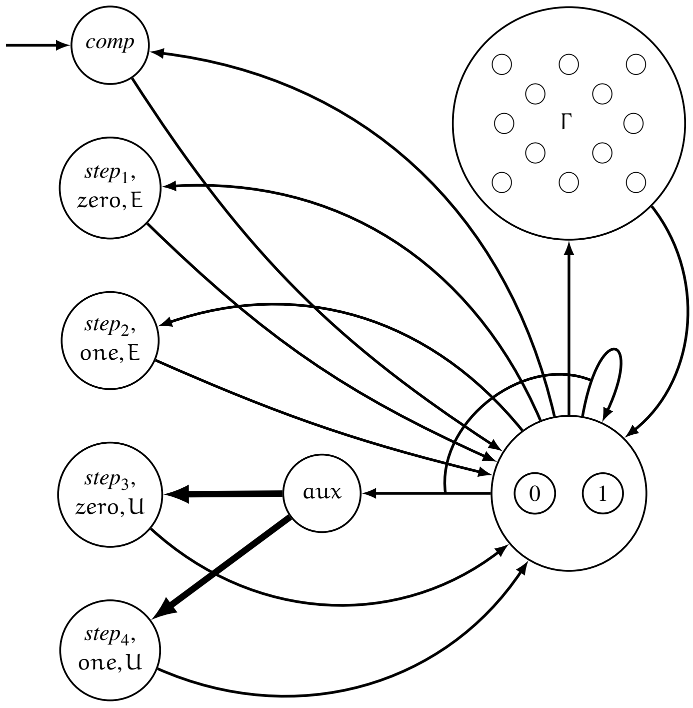


Figure 3 The MDP $\mathcal{M}'$. The behavior is probabilistic except for the choice in the state aux. When entering the cluster of states Γ or the cluster with the two bits 0 and 1, one of the states in the cluster is chosen randomly. Further all states in Γ have only the outgoing transition randomly moving to 0 or 1. The state aux is only an auxiliary state for the graphical representation. That means that in states 0 and 1 two actions are enabled. The first moving randomly to any state except for $step_3$; the second moving randomly to any state except for $step_4$.

correct length and use universally quantified variables to mark violations to any of the requirements of a valid encoding of an accepting computation. The blocks of the potential binary counter values are also randomly generated as sketched in Figure 3. An existentially quantified proposition encoding a further binary counter with $k-1$-exponentially many bits is then used to compare tape cells at the same position in two successive configurations, which are k-exponentially many steps apart in the encoding. Under any scheduler, the resulting Π_k^{QLTL}-formula φ holds on an execution of $\mathcal{M}$ almost surely if w is accepted by $\mathcal{T}$. Similar to before, each of the randomly generated potential computations is correct with positive probability and in each of these computations the randomly chosen moves of the existential player are in accordance with a winning strategy with positive probability against any scheduler, which chooses the moves of the universal player. If w is not accepted by $\mathcal{T}$, however, there is a strategy for the universal player and hence a scheduler that makes sure that no correct accepting computation is generated. In this case, $\Pr^{\min}_{\mathcal{M}}(\varphi) = 0$. ◀

These results allow us to conclude that all qualitative model-checking problems for Σ_k^{QLTL}-formulas in MDPs are $k+1$-EXPTIME-complete for any $k \geqslant 1$, too, as the negation of a Σ_k^{QLTL}-formula is a Π_k^{QLTL}-formula. Furthermore, as the upper bounds are obtained via the naive construction of deterministic automata, also the quantitative model checking problems have the same complexity as the minimal and maximal probabilities that an execution of an MDP is accepted by a suitable deterministic automaton (such as a deterministic Rabin automaton) can be computed in polynomial time (for details see, e.g., [2]).

▶ **Corollary 5** (Quantitative model checking)**.** *Given a Σ_k^{QLTL}- or Π_k^{QLTL}-formula φ and an MDP $\mathcal{M}$, the probabilities $\Pr_{\mathcal{M}}^{\min}(\varphi)$ and $\Pr_{\mathcal{M}}^{\max}(\varphi)$ can be computed in time $k+1$-exponential in the size of φ and polynomial in the size of $\mathcal{M}$. Given a rational $\vartheta \in [0,1]$, $\bowtie \in \{\leqslant, <, >, \geqslant\}$ and $\mathrm{opt} \in \{\min, \max\}$, deciding whether $\Pr_{\mathcal{M}}^{\mathrm{opt}}(\varphi) \bowtie \vartheta$ is $k+1$-EXPSPACE-complete.*

4 Trace Vacuity in Probabilistic Systems

Vacuity notions have been studied for non-probabilistic systems in order to express, roughly, that the truth of a formula is not affected by the truth of one of its subformulae [1, 3, 16]. Among the existing definitions of vacuity in the literature, *trace vacuity* is the strongest.

▶ **Definition 6.** *Let φ be an LTL-formula and ψ a subformula. Let $\mathcal{T}$ be a transition system. We say that ψ* does not affect *φ in $\mathcal{T}$ if for every execution π in $\mathcal{T}$:*

$$\pi \models \forall x.\varphi[\psi \leftarrow x] \quad \iff \quad \pi \models \exists x.\varphi[\psi \leftarrow x].$$

We say that φ holds vacuously in $\mathcal{T}$ if there is a subformula that does not affect φ in $\mathcal{T}$.

The above definition of non-affection generalizes the one from [1] by relaxing the hypothesis that φ holds on $\mathcal{T}$. For any execution π, $\pi \models \forall x.\varphi[\psi \leftarrow x] \Rightarrow \pi \models \varphi \Rightarrow \pi \models \exists x.\varphi[\psi \leftarrow x]$. We thus merely require that the three sets of executions that satisfy, $\forall x.\varphi[\psi \leftarrow x]$, φ, and $\exists x.\varphi[\psi \leftarrow x]$ respectively, coincide. Also, this generalisation allows us to naturally extend the notions of non-affection and vacuity to probabilistic systems. In the remainder of this section, we introduce trace vacuity for probabilistic systems, and establish tight complexity bounds for checking probabilistic vacuity. As in the non-probabilistic case, vacuity checking reduces to checking a Π_1^{QLTL}-formula. Conversely, one can reduce the qualitative model checking of Π_1^{QLTL} to probabilistic vacuity.

4.1 Probabilistic trace vacuity

▶ **Definition 7.** *Let φ be an LTL-formula and ψ a subformula. Let $\mathcal{M}$ be an MDP or a Markov chain. We say that ψ* does not affect *φ in $\mathcal{M}$ iff*

$$\Pr_{\mathcal{M}}^{\min}(\forall x.(\varphi[\psi \leftarrow x] \leftrightarrow \varphi)) = 1.$$

We say that φ is vacuous in $\mathcal{M}$ if there is a subformula that does not affect φ in $\mathcal{M}$.

Note that it does make sense for Markov chains and MDPs to consider that a formula is vacuous if its satisfaction probability (under any scheduler) is not affected when replacing a subformula, even if the global formula does not hold almost-surely. In MDPs, the definition of non-affection generalizes the non-probabilistic definition. This is made more precise in the following proposition. Paths in a transition system correspond to schedulers not making use of randomization when we view a transition system as an MDP.

▶ **Proposition 8.** *A subformula ψ does not affect a formula φ in an MDP (or a Markov chain) $\mathcal{M}$ if and only if for all schedulers $\mathfrak{S}$, $\Pr_{\mathcal{M}}^{\mathfrak{S}}(\forall x.\varphi[\psi \leftarrow x]) = \Pr_{\mathcal{M}}^{\mathfrak{S}}(\varphi) = \Pr_{\mathcal{M}}^{\mathfrak{S}}(\exists x.\varphi[\psi \leftarrow x])$.*

Proof. Let us rewrite $\forall x.(\varphi[\psi \leftarrow x] \leftrightarrow \varphi)$ as $(\forall x.(\varphi \rightarrow \varphi[\psi \leftarrow x])) \wedge (\forall x.(\varphi[\psi \leftarrow x] \rightarrow \varphi))$. The latter is equivalent to $(\varphi \rightarrow \forall x.\varphi[\psi \leftarrow x]) \wedge (\forall x.\neg\varphi[\psi \leftarrow x] \vee \varphi)$. Rewritten as implications, we obtain $(\varphi \rightarrow \forall x.\varphi[\psi \leftarrow x]) \wedge (\exists x.\varphi[\psi \leftarrow x] \rightarrow \varphi)$. As the two implications $(\varphi \leftarrow \forall x.\varphi[\psi \leftarrow x])$ and $(\exists x.\varphi[\psi \leftarrow x] \leftarrow \varphi)$ are tautologies, the claim follows easily considering that the minimal probability in Definition 7 can be read as a universal quantification over schedulers. ◀

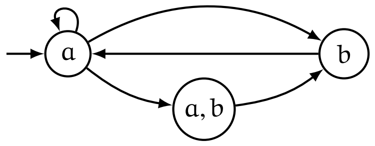


Figure 4 A Markov chain to illustrate the notion of affection.

▶ **Example 9.** We provide a short example of non-affection in Markov chains, also to shed light on the difference with the non-probabilistic setting. Consider the Markov chain on Fig. 4, where we assume arbitrary non-zero probabilities on edges, and the following formulas: $\varphi = \Box\Diamond(a \wedge b) \vee \Box(a \vee b)$ and $\psi = \Box(a \vee b)$. Clearly enough, $\Pr_{\mathcal{M}}(\varphi) = \Pr_{\mathcal{M}}(\exists x.\varphi[\psi \leftarrow x]) = \Pr_{\mathcal{M}}(\forall x.\varphi[\psi \leftarrow x]) = 1$ so that ψ does not affect φ, and φ is vacuous in this Markov chain. However, if one views the graph as a transition system $\mathcal{T}$, then $\mathcal{T} \models \varphi$ and $\mathcal{T} \not\models \forall x.\varphi[\psi \leftarrow x]$. So, ψ affects φ.

Armoni *et al.* [1] observed that if ψ appears only positively in φ, for every execution π in the transition system $\mathcal{T}$ then: $\mathcal{T}, \pi \models \forall x.\varphi[\psi \leftarrow x] \iff \mathcal{T}, \pi \models \varphi[\psi \leftarrow \bot]$. As a consequence, a pure polarity subformulas ψ does not affect φ if and only if $\Pr^{\min}_{\mathcal{M}}(\varphi[\psi \leftarrow \top] \leftrightarrow \varphi[\psi \leftarrow \bot]) = 1$. Therefore, checking whether a pure polarity subformula affects a formula reduces to quantitative model checking of LTL formulas and can be done in PSPACE for Markov chains and in 2-EXPTIME for MDPs.
As also argued in [1], restricting attention to subformulas with pure polarity or to consider single occurrences of subformulas separately is insufficient for a satisfactory vacuity check. For example, a formula like $\Box(p \to p) \equiv \Box(p \vee \neg p)$, in which p occurs positively and negatively, should be rendered vacuous in any system. Restricting attention to only one of the two occurences of p, however, would in general lead to the insight that each of the two occurrences on its own does affect the formula.
Beyond pure polarity formulas, checking affectation is harder for Markov chains. Indeed, hardness of Π_1^{QLTL} model checking transfers to hardness of vacuity checking. As stated in the next theorem, for MDPs affection checking has the same complexity as quantitative LTL model checking, whereas Markov chains exhibit an exponential complexity blowup.

▶ **Theorem 10.** *Checking whether a subformula ψ affects an LTL-formula φ in a Markov chain $\mathcal{M}$ is EXPSPACE-complete. In MDPs, the problem is 2-EXPTIME-complete.*

Proof. The upper bounds follow directly from the upper bounds of qualitative model-checking of Π_1^{QLTL} in Markov chains and MDPs. For the lower bound, we first concentrate on MDPs. We provide a reduction from the problem whether a Π_1^{QLTL}-formula $\vartheta = \forall x.\varphi$ satisfies $\Pr^{\min}_{\mathcal{M}}(\vartheta) = 1$ in an MDP $\mathcal{M}$. We sketch a proof that the restriction to one quantified variable does not influence the complexity in Appendix C. So, let $\mathcal{M}$ be labeled with atomic propositions from AP. Let $\vartheta = \forall x.\varphi$ where φ is an LTL-formula over $\mathsf{AP} \cup \{x\}$ with $x \notin \mathsf{AP}$ be given. We construct the MDP $\mathcal{M}'$ by adding a new initial state s'_{init} from which the original initial state s_{init} is reached in one step with probability 1. Further, we let β be an LTL-formula that is valid and does not occur in φ. Finally, we define φ' to be the LTL-formula $\varphi' = \beta \vee \bigcirc\varphi[x \leftarrow \beta]$. Of course, $\Pr^{\min}_{\mathcal{M}',s'_{init}}(\varphi') = 1$ as β is valid. We claim that β does not affect φ' in $\mathcal{M}'$ if and only if $\Pr^{\min}_{\mathcal{M}}(\forall x.\varphi) = 1$. The subformula β does not affect φ' in $\mathcal{M}'$ iff $\Pr^{\min}_{\mathcal{M}',s'_{init}}(\forall x.(x \vee \bigcirc\varphi)) = 1$ by definition and the fact that β does not occur anywhere else in φ. But $\Pr^{\min}_{\mathcal{M}',s'_{init}}(\forall x.(x \vee \bigcirc\varphi)) = 1$ holds if and only if $\Pr^{\min}_{\mathcal{M}',s'_{init}}(\forall x.\bigcirc\varphi) = 1$ because the universal quantifier can choose x not to hold in the first position of any trace produced by $\mathcal{M}'$. After the first step $\mathcal{M}'$ behaves exactly like $\mathcal{M}$ and hence $\Pr^{\min}_{\mathcal{M}',s'_{init}}(\forall x.\bigcirc\varphi) = 1$ if and only if $\Pr^{\min}_{\mathcal{M},s_{init}}(\forall x.\varphi) = 1$. So, checking affection in MDPs is as hard as the respective qualitative model-checking problem for Π_1^{QLTL} and hence 2-EXPTIME-complete.
For Markov chains, the argument goes analogously. Note that the constructed MDP $\mathcal{M}'$ is a Markov chain if $\mathcal{M}$ is a Markov chain. So, checking affection in Markov chains is also as hard as the respective

qualitative model-checking problem for Π_1^{QLTL} and hence EXPSPACE-complete. ◀

In Markov chains, the exponential blow-up in complexity of non-affection checking compared to LTL-model checking constitutes a major obstacle for vacuity checking. To provide a possibility to check that a specification is not obviously faulty without such an exponential blow-up, we turn our attention to the notion of inherent vacuity.

4.2 Inherent vacuity in probabilistic systems

Inherent vacuity for transition systems expresses whether a formula holds vacuously in every model in which it holds [9]. Using our generalized definition, we do not restrict ourselves to the models in which the formula holds anymore and provide an analogous definition for probabilistic systems. As in [9], we consider two natural variants of the definition and investigate how to check whether a formula is inherently vacuous.

▶ **Definition 11.** *Let φ be an LTL-formula. Let $\mathcal{C}$ be the class of all transition systems, all Markov chains, or all MDPs, respectively. We say that φ is inherently vacuous over $\mathcal{C}$, if φ is vacuous in all models $\mathcal{M} \in \mathcal{C}$. For a subformula ψ of φ we say that ψ inherently does not affect φ over $\mathcal{C}$, if for every $\mathcal{M} \in \mathcal{C}$, ψ does not affect φ in $\mathcal{M}$. If there is a subformula that inherently does not affect φ over $\mathcal{C}$, we say that φ is uniformly inherently vacuous.*

In [9], it is shown that inherent vacuity and uniform inherent vacuity coincide for transition systems. Dropping the restriction to models in which a formula φ holds, the results of [9] show that the notions are equivalent to the existence of a subformula ψ such that $\forall x.(\varphi[\psi \leftarrow x] \leftrightarrow \varphi)$ is valid. We prove that inherent and uniform inherent vacuity for Markov chains and MDPs are also equivalent to this condition and hence to inherent vacuity in transition systems. First, we show that uniform inherent vacuity coincides with inherent vacuity.

▶ **Proposition 12.** *Let φ be an LTL-formula and let $\mathcal{C}$ be the class of all Markov chains or all MDPs. The formula φ is uniformly inherently vacuous over $\mathcal{C}$ if and only if it is inherently vacuous over $\mathcal{C}$.*

Proof. One direction is clear. For the other direction, suppose that φ is inherently vacuous over $\mathcal{C}$, but not uniformly inherently vacuous. Hence, for each subformula ψ of φ, there is a model $\mathcal{M}_\psi \in \mathcal{C}$ such that ψ affects φ over $\mathcal{M}_\psi$. Let $\mathcal{N}$ be the disjoint union of the models $\mathcal{M}_\psi$ for all subformulas ψ with an initial uniform probability distribution over the initial states of these models. We claim that φ is not vacuous in $\mathcal{M}$. For each subformula ψ, there is a positive probability that $\mathcal{M}_\psi$ is chosen. As there is a scheduler $\mathfrak{S}$ (for Markov chains, the unique scheduler) with $\Pr^{\mathfrak{S}}_{\mathcal{M}_\psi}(\forall x.(\varphi[\psi \leftarrow x] \leftrightarrow \varphi) < 1$, the same holds in $\mathcal{N}$. This is a contradiction to the inherent vacuity of φ. ◀

The following proposition establishes that all variants of inherent vacuity considered coincide:

▶ **Proposition 13.** *Let φ be an LTL-formula and ψ a subformula. Then, ψ inherently does not affect φ over Markov chains or MDPs, respectively, if and only if the formula $\forall x.(\varphi \leftrightarrow \varphi[\psi \leftarrow x])$ is valid.*

Proof. Only the left-to-right implication deserves a proof, and we prove the contrapositive. Assume the formula $\chi = \forall x.(\varphi \leftrightarrow \varphi[\psi \leftarrow x])$ is not valid. Since χ expresses a regular property, there exists an ultimately periodic word w that violates χ. It suffices to consider the Markov chain or MDP $\mathcal{M}$ that has only one path, and produces w with probability 1, and observe that ψ does affect φ in $\mathcal{M}$. ◀

As a consequence, checking inherent vacuity for probabilistic systems is as simple as in the non-probabilistic case, and can be done in polynomial space. In particular for Markov chains, an inherent vacuity check might be an interesting option for practical applications as it avoids the exponential blow-up in complexity over LTL-model checking.

5 Conclusion

We determined the precise complexities of the model-checking problems for the different levels of the quantifier alternation hierarchy of QLTL over probabilistic systems. The knowledge of the precise complexities, in particular the established lower bounds, has the potential to serve as the basis for hardness proofs for other questions in the formal verification of probabilistic systems. Despite the high complexities that we obtained, efficient model checking for formulas with few quantifier alternations might still be possible because all problems are solvable in time polynomial in the size of the system and typically formulas are small compared to the size of the models.
These results have been applied to the notion of trace vacuity known from the non-probabilistic setting that we adapted to the probabilistic setting. It turned out that checking whether a formula is affected by a subformula in a system is inter-reducible with Π_1^{QLTL}-model checking. For Markov chains, our new lower bounds allowed us to conclude that affection checking is EXPSPACE-complete and hence exponentially harder than LTL-model checking, while the complexity of affection checking and LTL-model checking are the same in MDPs. Furthermore, we showed that the notion of inherent vacuity – expressing that a formula is vacuous in a class of system models – is invariant under the switch from non-probabilistic to probabilistic models, and hence, known polynomial-space algorithms are applicable for Markov chains and MDPs. In addition to the vacuity notions we studied here, an interesting direction for future research is the investigation of "more probabilistic" notions of vacuity that express that a perturbation of a subformula does not influence the satisfaction probability of a formula in a system.

References

1 Roy Armoni, Limor Fix, Alon Flaisher, Orna Grumberg, Nir Piterman, Andreas Tiemeyer, and Moshe Y. Vardi. Enhanced vacuity detection in linear temporal logic. In *Proceedings of the 15th International Conference on Computer Aided Verification (CAV'03)*, volume 2725 of *Lecture Notes in Computer Science*, pages 368–380. Springer, 2003.

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

3 Ilan Beer, Shoham Ben-David, Cindy Eisner, and Yoav Rodeh. Efficient detection of vacuity in temporal model checking. *Formal Methods in System Design*, 18(2):141–163, 2001.

4 Doron Bustan, Alon Flaisher, Orna Grumberg, Orna Kupferman, and Moshe Y. Vardi. Regular vacuity. In *Proceedings of the 13th IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME'05)*, volume 3725 of *Lecture Notes in Computer Science*, pages 191–206. Springer, 2005.

5 Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, January 1981.

6 Edmund Clarke, Orna Grumberg, and Doron Peled. *Model Checking*. MIT Press, 2000.

7 Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.

8 Bernd Finkbeiner and Leander Tentrup. Detecting unrealizability of distributed fault-tolerant systems. *Logical Methods in Computer Science*, 11(3):1–31, 2015.

9 Dana Fisman, Orna Kupferman, Sarai Sheinvald-Faragy, and Moshe Y. Vardi. A framework for inherent vacuity. In *Proceedings of the 4th International Haifa Verification Conference (HVC'08)*, volume 5394 of *Lecture Notes in Computer Science*, pages 7–22. Springer, 2008.

10 Dov Gabbay. The declarative past and imperative future. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Temporal Logic in Specification*, pages 409–448, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg.

11 Arie Gurfinkel and Marsha Chechik. Extending extended vacuity. In *Proceedings of the 5th International Conference on Formal Methods in Computer-Aided Design (FMCAD'04)*, volume 3312 of *Lecture Notes in Computer Science*, pages 306 – 321. Springer, 2004.

12 Arie Gurfinkel and Marsha Chechik. How vacuous is vacuous? In *Proceedings of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*, pages 451–466, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

13 Walter Hussak. Serializable histories in quantified propositional temporal logic. *International Journal of Computer Mathematics*, 81(10):1203–1211, 2004.

14 Yonit Kesten and Amir Pnueli. Complete proof system for QPTL. *Journal of Logic and Computation*, 12(5):701–745, 2002.

15 Orna Kupferman. Sanity checks in formal verification. In *Proceedings of the 17th International Conference on Concurrency Theory (CONCUR'06)*, volume 4137 of *Lecture Notes in Computer Science*, pages 37 – 51. Springer, 2006.

16 Orna Kupferman and Moshe Y. Vardi. Vacuity detection in temporal model checking. *International Journal on Software Tools for Technology Transfer*, 4(2):224–233, 2003.

17 François Laroussinie and Nicolas Markey. Quantified CTL: Expressiveness and Complexity. *Logical Methods in Computer Science*, 10(4):1–45, 2014.

18 Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

19 Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

20 A. Prasad Sistla. *Theoretical Issues in the Design and Verification of Distributed Systems*. PhD thesis, Carnegie-Mellon University, 1983.

21 A. Prasad Sistla, Moshe Y. Vardi, and Pierre Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49(2–3):217–237, 1987.

22 Aravinda P. Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

23 Peter van Emde Boas. The convenience of tilings. *Lecture Notes in Pure and Applied Mathematics*, pages 331–363, 1997.

24 Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proceedings of the 1st Symposium on Logic in Computer Science (LICS'86)*, pages 332–344. IEEE Computer Society Press, 1986.

25 Pierre Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1):72 – 99, 1983.

A Full proof of Theorem 2

▶ **Theorem 14** (Theorem 2). *For any* k*, all qualitative model checking problems for* Σ_k^{QLTL} *and* Π_k^{QLTL} *in Markov chains are* k*-EXPSPACE-complete.*

Proof. The upper bounds are obtained by building NBAs of k-exponential size for Σ_k^{QLTL}-formulas as described in Section 2. The negation of a Π_k^{QLTL}-formula is equivalent to a Σ_k^{QLTL}-formula of the same length. As all qualitative model-checking problems for NBAs in Markov chains are PSPACE-complete [7], we obtain the upper bounds.

For the hardness results, we use a reduction from k-exponential tiling problems. We define the following function $h\colon \mathbb{N}^2 \to \mathbb{N}$: Let $h(0,n) = n$ for all n and $h(k+1,n) = 2^{h(k,n)} \cdot h(k,n)$ for all k. So, $h(k,n)$ is k-exponential in n.

k-exponential tiling problem

Given: a finite set of tiles T, two relations $H \subseteq T^2$ and $V \subseteq T^2$, an initial tile $t_0 \in T$ and a final tile $t_f \in T$ as well as a natural number n in unary.
Question: Is there a natural number m such that the $2^{h(k-1,n)} \times (m+1)$-grid $\{0,\dots,2^{h(k-1,n)}-1\} \times \{0,\dots,m\}$ can be tiled, i. e., is there a function

$$f\colon \{0,\dots,2^{h(k-1,n)}-1\} \times \{0,\dots,m\} \to T,$$

such that:

1. the tile at position $(0,0)$ is the initial tile t_0 and the tile at position $(0,m)$ is the final tile t_f; in other words, $f(0,0) = t_0$ and $f(0,m) = t_f$,
2. two tiles placed next to each other horizontally satisfy the relation H; more precisely, for any $0 \leqslant i < 2^{h(k-1,n)} - 1$ and $0 \leqslant j \leqslant m$, the pair $(f(i,j), f(i+1,j)) \in H$, and
3. two tiles placed next to each other vertically satisfy the relation V; more precisely, for any $0 \leqslant i \leqslant 2^{h(k-1,n)} - 1$ and $0 \leqslant j < m$, the pair $(f(i,j), f(i,j+1)) \in V$?

This problem is known to be k-EXPSPACE-complete [23].

An important ingredient for our reduction is the collection of Σ_k^{QLTL}-formulas $\varphi_{k,n}(p,q)$ from [21]. For each n and k from $\mathbb{N}$, the formula $\varphi_{k,n}(p,q)$ holds on a word if p and q occur exactly once and, if the position at which p occurs is i, the position at which q occurs is $i + h(k,n)$.

Given an instance of the k-exponential tiling problem, we will construct a Markov chain $\mathcal{M}$ and a Π_k^{QLTL}-formula ψ such that $\Pr_{\mathcal{M}}(\psi) = 1$ iff $\Pr_{\mathcal{M}}(\psi) > 0$ iff there is a valid tiling. This establishes k-EXPSPACE-hardness for both qualitative model checking problems for Π_k^{QLTL}. As the negation of ψ is a Σ_k^{QLTL}-formula and k-EXPSPACE is closed under complementation, the same result holds for Σ_k^{QLTL} in Markov chains.

The Markov chain $\mathcal{M}$ is very simple (see Figure 5). Let $T = \{t_0, \dots, t_\ell\}$. The Markov chain has states $T \cup \{\mathit{start}, 0, 1, \mathit{end}\}$. The initial state is *start*. From there the chain randomly moves to one of the states from $T \cup \{0,1\}$. From each of these states, it continues by moving to one of the states from $T \cup \{0, 1, \mathit{end}\}$ and from *end* it moves back to the state *start*. The precise probabilities are not important. A run of $\mathcal{M}$ almost surely consists of a concatenation of infinitely many words from $\mathit{start}(T \cup \{0,1\})^+\mathit{end}$. Furthermore, each of these finite words is contained in a run with probability 1.

We will use finite words from $\mathit{start}(T \cup \{0,1\})^+\mathit{end}$ to encode potential tilings: Let $f\colon \{0,\dots,2^{h(k-1,n)}-$

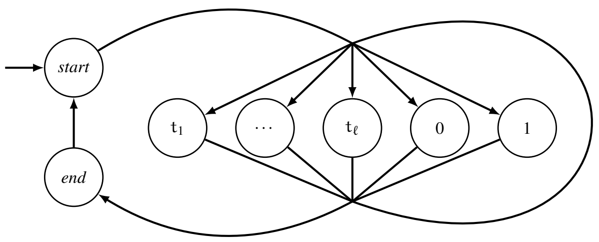


Figure 5 The Markov chain $\mathcal{M}$.

$1\} \times \{0,\ldots,m\} \to T$ be a function. The following word encodes this potential tiling:

$$\begin{aligned}
&\textit{start},\\
&f(0,0),\ \overbrace{0,0,0,\ldots,0}^{h(k-1,n)\text{ steps}},f(1,0),\ \overbrace{1,0,0,\ldots,0}^{h(k-1,n)\text{ steps}},f(2,0),\ \overbrace{0,1,0,\ldots,0}^{h(k-1,n)\text{ steps}},f(3,0),\ \ldots,\\
&\qquad f(2^{h(k-1,n)}-1,0),\ \overbrace{1,1,1,\ldots,1}^{h(k-1,n)\text{ steps}},\\
&f(0,1),\ \overbrace{0,0,0,\ldots,0}^{h(k-1,n)\text{ steps}},f(1,1),\ \overbrace{1,0,0,\ldots,0}^{h(k-1,n)\text{ steps}},f(2,1),\ \overbrace{0,1,0,\ldots,0}^{h(k-1,n)\text{ steps}},f(3,1),\ \ldots,\\
&\qquad f(2^{h(k-1,n)}-1,1),\ \overbrace{1,1,1,\ldots,1}^{h(k-1,n)\text{ steps}},\\
&\quad\vdots\\
&f(0,m),\ \overbrace{0,0,0,\ldots,0}^{h(k-1,n)\text{ steps}},f(1,m),\ \overbrace{1,0,0,\ldots,0}^{h(k-1,n)\text{ steps}},f(2,m),\ \overbrace{0,1,0,\ldots,0}^{h(k-1,n)\text{ steps}},f(3,m),\ldots,\\
&\qquad f(2^{h(k-1,n)}-1,m),\ \overbrace{1,1,1,\ldots,1}^{h(k-1,n)\text{ steps}},\textit{end}
\end{aligned}$$

In order for a word to encode a potential tiling, we hence require that the elements from T are separated by $h(k-1,n)$-many bits from $\{0,1\}$. These bits have to encode a correct binary counter modulo $2^{h(k-1,n)}$ starting with $0\ldots0$ after *start* and ending in $1\ldots1$ before *end*. This ensure that there are $2^{h(k-1,n)}\cdot m$ tiles included in the finite word for some $m \in \mathbb{N}$. The first digit in the counter is considered to be the least significant one. Furthermore, it implies that horizontally adjacent tiles in the tiling are followed by the same binary counter value. Vertically adjacent tiles in the tiling are consecutive tiles in the encoding.
We will now construct a Π_k^{QLTL}-formula $\mathtt{valid\_tiling}$ that expresses that at some point a valid tiling is encoded in a run. Some of the conditions of a valid tiling can be easily expressed in LTL. For the remaining conditions, we use universally quantified variables to mark positions where the condition is violated. Hence, we want to check that eventually a valid tiling containing no markers for violations of conditions is encoded in a run. The formula $\mathtt{valid\_tiling}$ will be of the following form:

$$\forall d \forall c \forall v_1 \forall v_2 \big(\psi \to \Diamond\big(\textit{start} \wedge \mathtt{initial} \wedge \mathtt{final} \wedge \mathtt{horizontal} \wedge \mathtt{correct\_dist}(d) \wedge \\ \mathtt{correct\_counter}(c) \wedge \mathtt{vertical}(v_1,v_2)\big)\big)$$

The subformula ψ expresses requirements on the quantified variables and contains quantifiers itself. As ψ will be a Σ_k^{QLTL}-formula that occurs under one negation as the antecedent of an implication, while the other named subformulas are quantifier-free, the resulting formula $\mathtt{valid\_tiling}$ will be in Π_k^{QLTL}.
The formulas that check the conditions for a valid tiling are always evaluated at the states labeled with *start* due to the structure of $\mathtt{valid\_tiling}$. Let us begin by providing the formulas for the conditions that can be checked by an LTL-formula:
The formula $\mathtt{initial}$ is the formula $\bigcirc t_0$.
The formula $\mathtt{final}$ is the formula

$$\Big(\Big[T \wedge \bigcirc(0\,\mathsf{U}\,T) \wedge \bigcirc\Big((\neg(T \wedge \bigcirc(0\,\mathsf{U}\,T)))\,\mathsf{U}\,\mathit{end}\Big)\Big] \to t_f\Big)\,\mathsf{U}\,\mathit{end}.$$

This formula states that if a tile is followed by the counter value $0\ldots0$ and no later tile before *end* is followed by the counter value $0\ldots0$, then this tile is the final tile t_f.
The formula $\mathtt{horizontal}$ is the following:

$$\big(T \to [\bigvee_{(t,t')\in H} (t \wedge \bigcirc((0\vee 1)\,\mathsf{U}\,t')) \vee \bigcirc(1\,\mathsf{U}(T \vee \mathit{end}))]\big)\,\mathsf{U}\,\mathit{end}.$$

This formula states that each tile before *end* is followed by a counter value and then a tile that is horizontally compatible according to H or by the counter value $1\ldots1$.
For the remaining formulas, we introduce universally quantified variables. Further, we need auxiliary formulas that express that two positions are $h(k-1,n)$-many steps apart. As mentioned above, in [21], it is shown that for each $k \geqslant 1$, there is a $\Sigma_{k-1}^{\mathsf{QLTL}}$-formula $\varphi_{k-1,n}(p,q)$ of length polynomial in n that holds if and only if p and q occur exactly once and the distance between the positions where they hold is exactly $h(k-1,n)$. I.e. if $\varphi_{k-1,n}(p,q)$ holds, and p holds at position i, then q holds at position $i+h(k-1,n)$.
Let us begin by showing how to check whether there are two tiles that are not separated by exactly $h(k-1,n)$-many bits from $\{0,1\}$. We introduce a universally quantified variable d. The variable d is supposed to hold at most once between each pair of *start* and *end*. Further, it is supposed to only hold on tiles. Finally, d has to mark a position that is not followed by exactly $h(k-1,n)$-many bits from $\{0,1\}$. These requirements are formulated in the following formula ψ_d:

$$\begin{aligned}&\Box(d \to T)\wedge\\ &\Box\big((\mathit{start} \wedge \neg \mathit{end}\,\mathsf{U}\,d) \to (\neg d\,\mathsf{U}(d \wedge \bigcirc(\neg d\,\mathsf{U}\,\mathit{end})))\big)\wedge\\ &\forall p \forall q\Big(\varphi_{k-1,n}(p,q) \to \Big[\Box\big((d \wedge \bigcirc p) \to\\ &\qquad\qquad\qquad [\neg\bigcirc(\neg(T \vee \mathit{end})\,\mathsf{U}\,q)) \vee \bigcirc(\neg(T\vee \mathit{end})\,\mathsf{U}(q \wedge (0 \vee 1)))]\big)\Big]\Big)\end{aligned}$$

The first lines simply states that d holds only on tiles and at most once between *start* and *end*. The universally quantified variables p and q can be placed at any two positions that are $h(k-1,n)$ steps apart. If they are placed differently, the antecedent of the following implication fails. The consequent of the implication states that if p holds directly after a position where d holds, then either $T \vee \mathit{end}$ holds at some point before q, or one step after q at the earliest. If this requirement holds for all p and q satisfying $\varphi_{k-1,n}(p,q)$, then d can only hold at tiles that are not followed by exactly $h(k-1,n)$-many bits. As $\varphi_{k-1,n}(p,q)$ is a $\Sigma_{k-1}^{\mathsf{QLTL}}$-formula that occurs as the antecedent of an implication, i.e. in the scope of one negation, the whole formula ψ_d can be written as a $\Pi_{k-1}^{\mathsf{QLTL}}$-formula.
The formula $\mathtt{correct\_dist}$ now simply is $\neg d\,\mathsf{U}\,\mathit{end}$. This simply expresses that no tile before *end* is marked as not being followed by exactly $h(k-1,n)$ bits of a binary counter.

Next, we will use a similar idea to construct a formula that can check whether the encoded binary counter is incorrect. We use a universally quantified variable c that will mark a bit whose successor is incorrect. Note that we let the first digit in a counter value be the least significant one. This means that the ith digit of a counter value does not change between two successive counter values if and only if there is a $j < i$ such that the jth digit is 0. A digit only changes if all previous digits are 1. The check whether the counter is incorrect will only be necessary on finite encodings between *start* and *end* in which the tiles are placed at the correct distances. The requirements for the variable c are now that it holds at most once between each occurrence of *start* and *end*, that it only holds on states labeled with 0 or 1, that another tile and binary counter value follow before *end*, and that the bit that is placed $h(k-1,n)+1$ steps later is incorrect. If the distance between tiles is correct, this is the successor bit of the bit marked by c. The formula ψ_c expresses these requirements:

$$\begin{aligned}
&\Box(c \to (0 \vee 1)) \wedge \\
&\Box\big((\mathit{start} \wedge \neg \mathit{end}\, \mathrm{U}\, c) \to (\neg c\, \mathrm{U}(c \wedge \bigcirc(\neg c\, \mathrm{U}\, \mathit{end})))\big) \wedge \\
&\forall p \forall q \Bigg(\varphi_{k-1,n}(p,q) \to \\
&\qquad \Big[\Box\Big((T \wedge \bigcirc(\neg T\, \mathrm{U}(c \wedge \bigcirc p))) \to \\
&\qquad\qquad \big[\neg \mathit{end}\, \mathrm{U}(q \wedge (0 \vee 1)) \wedge \big((\neg c\, \mathrm{U}(c \wedge 0)) \leftrightarrow \\
&\qquad\qquad\qquad ((\neg q\, \mathrm{U}(q \wedge 0) \wedge \neg \bigcirc (1\, \mathrm{U}\, c)) \vee \neg q\, \mathrm{U}(q \wedge 1) \wedge \bigcirc(1\, \mathrm{U}\, c))\big)\big]\Big)\Big]\Bigg).
\end{aligned}$$

The first two lines work analogously to the first lines of the formula ψ_d and state that c holds at most once between *start* and *end* and only on bits. The universally quantified varibales p and q are again used to simultaneously check the successors of all c labeled bits. The antecedent after $\Box$ in the fourth line now holds on a tile that is followed by a counter value in which a bit is marked by c if p is placed directly after c. This implies that q is placed $h(k-1,n)+1$ steps after c and hence marks the successor bit of c. If in the counter value all bits before the one marked with c are 1, then the bits marked with c and q have to be different. Otherwise, they have to be the same. This is expressed in the bi-implication in the last two lines.
The formula $\mathtt{correct\_counter}$ is now

$$\bigcirc\bigcirc(0\, \mathrm{U}\, T) \wedge \neg(T \wedge \bigcirc(\neg T\, \mathrm{U}\, \mathit{end}))\, \mathrm{U} \bigcirc(1\, \mathrm{U}\, \mathit{end}) \wedge \neg c\, \mathrm{U}\, \mathit{end}.$$

The last conjunct states that no counter bit is marked as incorrect, analogously to the formula $\mathtt{correct\_dist}$ above. The other conjuncts state that the first counter value is $0\ldots0$ and the last counter value before *end* is $1\ldots1$.
Finally, we have to provide a formula that can detect violations of the vertical condition. The difficulty here is that vertically adjacent tiles are $2^{h(k-1,n)} \cdot (h(k-1,n)+1)$ steps apart. We want to mark such tiles with universally quantified variables v_1 and v_2. If we now would check whether the distance between these variables is correct by using universally quantified variables p and q as above, we would have to rely on the formula $\varphi_{k,n}(p,q)$ in the antecedent of an implication. The resulting formula would then not be in Σ_k^{QLTL} anymore. Hence, we will use an auxiliary existentially quantified variable b. This variable then has to encode a correct binary counter between the positions at which v_1 and v_2 hold. This counter has to start with value 0 and count up to $2^{h(k-1,n)}-1$. The check of the vertical condition is only necessary if the tiles are placed at the correct distances. So, the formula only has to work as intended if the distances between tiles are correct. We can then check the correctness of the counter encoded by b and $\neg b$ similar to the correctness of the counter encoded by 0 and 1. The universally quantified variables v_1 and v_2 are supposed to hold at most once between *start* and *end* in

this order. Further, they should only hold on tiles. The formula ψ_v is hence the conjunction of the following formulas:

$$\Box((v_1 \vee v_2) \to \mathsf{T})$$

states that v_1 and v_2 hold only on tiles.

$$\bigwedge_{i\in\{1,2\}} \Box\big((\mathit{start} \wedge \neg \mathit{end}\, \mathrm{U}\, v_i) \to (\neg v_i\, \mathrm{U}(v_i \wedge \bigcirc(\neg v_i\, \mathrm{U}\, \mathit{end})))\big)$$

states that they hold at most once between *start* and *end*, and

$$\Box(v_1 \to \bigcirc\neg(\neg v_2\, \mathrm{U}\, \mathit{end}))$$

that v_1 has to occur before v_2. The correctness of the counter is then expressed in the following formula:

$$\begin{aligned}
&\exists b \Big(\Box(v_1 \to \bigcirc(\neg b\, \mathrm{U}\, \mathsf{T})) \wedge \Box((\mathsf{T} \wedge \bigcirc(\neg \mathsf{T}\, \mathrm{U}\, v_2)) \to \bigcirc(b\, \mathrm{U}\, v_2)) \wedge \\
&\forall p \forall q \Big(\varphi_{k-1,n}(p,q) \to \\
&\qquad \Box\Big[\big(v_1 \wedge (\neg v_2\, \mathrm{U}(p \wedge (0 \vee 1))) \wedge (\neg v_2\, \mathrm{U}\, q)\big) \to \\
&\qquad\quad \Big(\neg p\, \mathrm{U}(p \wedge b) \leftrightarrow \big(((\mathsf{T} \to \neg \bigcirc(b\, \mathrm{U}\, p))\, \mathrm{U}\, p) \leftrightarrow \neg q\, \mathrm{U}(q \wedge \bigcirc b)\big)\Big)\Big]\Big) \wedge \\
&\Box(v_1 \to (\neg(\mathsf{T} \wedge \bigcirc(b\, \mathrm{U}\, \mathsf{T}))\, \mathrm{U}(\mathsf{T} \wedge \bigcirc(b\, \mathrm{U}\, v_2))))\Big).
\end{aligned}$$

As mentioned above, we assume that the states from T are separated by $h(k-1,n)$ states from $\{0,1\}$. The variable b encodes counter values on exactly these blocks of $h(k-1,n)$ states. The first line ensures that b does not hold in the first block of bits after v_1; the second line that b always holds on all positions in the last block before v_2. The third line introduces p and q at distance $h(k-1,n)$ if the antecedent $\varphi_{k-1,n}(p,q)$ holds. The next lines state that if p and q hold somewhere between v_1 and v_2 and p marks a position in one of the blocks of bits from $\{0,1\}$, then the following condition has to hold: Either, the next p marks a position at which b holds; then the successor bit $h(k-1,n)+1$ steps later satisfies b iff b does not hold on all positions of bits before p. The successor bit is encoded one position after q. Or, the next p marks a position at which $\neg b$ holds. Then the condition for $\neg b$ one step after q is analogously. The last line ensures that the counter encoded by b reaches $1\ldots1$ only once before v_2. Hence, v_1 and v_2 are indeed placed on the encodings of vertically adjacent tiles.
Again, the formula $\varphi_{k-1,n}(p,q)$ appears in the scope of one negation. So, the subformula starting with the universal quantification of p,q is in $\Pi^{\mathrm{QLTL}}_{k-1}$. That means that the whole formula ψ_v is in $\Sigma^{\mathrm{QLTL}}_{k}$.
The formula $\mathit{vertical}$ is now stating that the two tiles marked with v_1 and v_2 satisfy the vertical condition if there are marked tiles:

$$\neg(\neg v_1\, \mathrm{U}\, \mathit{end}) \to \bigvee_{(t,t')\in V} ((\neg v_1\, \mathrm{U}(v_1 \wedge t)) \wedge (\neg v_2\, \mathrm{U}(v_2 \wedge t'))).$$

This completes the formula $\mathit{valid\_tiling}$:

$$\begin{aligned}
&\forall d \forall c \forall v_1 \forall v_2 \big((\psi_d \wedge \psi_c \wedge \psi_v) \to \Diamond\big(\mathit{start} \wedge \mathit{initial} \wedge \mathit{final} \wedge \mathit{horizontal} \wedge \\
&\qquad\qquad \mathit{correct\_dist}(d) \wedge \mathit{correct\_counter}(c) \wedge \mathit{vertical}(v_1,v_2)\big)\big)
\end{aligned}$$

As ψ_v is in Σ_k^{QLTL} and ψ_c and ψ_d are in Π_{k-1}^{QLTL} and these formulas occur in the scope of one negation while the remaining formulas are quantifier-free, the whole formula can be written as a Π_k^{QLTL} formula of size polynomial in the size of T, V, H, and n which was given in unary.
We claim that in the Markov chain $\mathcal{M}$ we have:

$\Pr_{\mathcal{M}}(valid\_tiling) = 1$ iff there is a valid tiling, and

$\Pr_{\mathcal{M}}(valid\_tiling) = 0$ iff there is no valid tiling.

So, suppose there is no valid tiling for the given instance of the tiling problem. A run of the Markov chain almost surely produces a concatenation of infinitely many words from $\textit{start}(T \cup \{0,1\})^+\textit{end}$. Consider one such run ρ. We can now simply choose positions for the quantified variables d, c, v_1, v_2, and b such that the formula does not holds on this run ρ. Each of the words from $\textit{start}(T \cup \{0,1\})^+\textit{end}$ either does not encode a tiling because the distance between the tiles or the counter are not correct or it does not encode a valid tiling. If the distance between the tiles is not correct, we pick a tile in the substring that is not followed by $h(k-1,n)$ bits and make d true at that position. Note that the formula is designed such that we can use each of the 'markers' on each of the finite words between *start* and *end*. So, the quantified variable d can be used to mark violations to the required distance in all infinitely many words in which that is necessary. If the distances are correct, but the counter is incorrect, we make c true at a bit with incorrect successor. If the finite word does encode a tiling, either one of the formulas $\neg intial$, $\neg final$, or $\neg horizontal$ holds when evaluated at the first position of this finite word, or the vertical condition is violated. In the first case, we do not have to mark any positions. In the second case, we mark two horizontally adjacent tiles within this finite word which are not compatible with v_1 and v_2. These positions are separated by $2^{h(k-1,n)}$ blocks of $h(k-1,n)$ bits. On these blocks, we make the variable b true such that it encodes a correct binary counter using $\neg b$ to represent 0 and b to represent 1. This counter will start with value $0\ldots0$ after v_1 and end with $1\ldots1$ before v_2. We conclude that $\rho \models \neg valid\_tiling$. So, $\Pr_{\mathcal{M}}(valid\_tiling) = 0$ in this case.
Now, suppose there is a valid tiling f and suppose that the finite word $w \in \textit{start}(T \cup \{0,1\})^+\textit{end}$ encodes this tiling. Almost all runs ρ contain this word at some point. Let ρ' be the suffix of a run that starts with w. By design, we have

$$\begin{aligned}\rho' \models & initial \wedge final \wedge horizontal \wedge correct\_dist(d)\\ & \wedge correct\_counter(c) \wedge vertical(v_1, v_2)\end{aligned}$$

for all possible choices of d, c, v_1, v_2, and b that satisfy $\psi_d \wedge \psi_c \wedge \psi_v$. Therefore,

$\Pr_{\mathcal{M}}(valid\_tiling) = 1.$

That means that deciding whether a Π_k^{QLTL}-formula holds almost surely and whether it holds with positive probability in a Markov chain are both k-EXPSPACE-complete. Hence, the same result holds for Σ_k^{QLTL} because the negations of Σ_k^{QLTL} are equivalent to Π_k^{QLTL} formulas of the same length. ◀

B Full proof of Theorem 4

▶ **Theorem 15** (Theorem 4)**.** *Given an MDP $\mathcal{M}$, a Π_k^{QLTL}-formula φ, and $\mathrm{opt} \in \{\max, \min\}$, deciding whether $\Pr_{\mathcal{M}}^{\mathrm{opt}}(\varphi) = 1$ and deciding whether $\Pr_{\mathcal{M}}^{\mathrm{opt}}(\varphi) > 0$ are $k+1$-EXPTIME-complete for any $k \geqslant 1$.*

We split up the proof into two parts in the sequel.

B.1 Proof of Theorem 4 for maximal satisfaction probabilities

First, we prove the result for opt = max which is somewhat easier.

▶ **Theorem 16.** *Given an MDP* $\mathcal{M}$ *and a* Π_k^{QLTL}*-formula* φ*, deciding whether* $\mathrm{Pr}_{\mathcal{M}}^{\max}(\varphi) = 1$ *and deciding whether* $\mathrm{Pr}_{\mathcal{M}}^{\max}(\varphi) > 0$ *are* $k+1$*-EXPTIME-complete for any* $k \geqslant 1$.

Proof. Again, the upper bounds are easily obtained: From the negation of a Π_k^{QLTL}-formula, i.e., a Σ_k^{QLTL}-formula, we can build a non-deterministic Büchi automaton in k-exponential time. A deterministic Rabin automaton (of $k+1$-exponential size) can hence be obtained in $k+1$-exponential time. Using this automaton all qualitative model-checking problems can then be solved in time polynomial in the size of the automaton.
For the hardness proof, we will encode alternating space-bounded Turing machines. Let us recall the definition of an alternating Turing machine (ATM). An ATM consists of a finite set of states Q, a finite tape alphabet Σ, a transition function

$$\delta : Q \times \Sigma \to \mathcal{P}(Q \times \Sigma \times \{L, S, R\}),$$

an initial state q_{init}, and a function g assigning to each state one of the types $\bigvee$ (existential), $\bigwedge$ (universal), *accept*, or *reject*. We can safely assume that there is exactly one state of type *accept* and one of type *reject*.
A computation of an ATM can be seen as a two player game between an existential and a universal player: Whenever the current state is existential, the existential player chooses a move from the possible moves given by the transition function that are interpreted as usual for a Turing machine. Likewise, the universal player chooses the move if the current state is universal. The existential player wins if the accepting state *accept* is reached. An input word is accepted if the existential player has a winning strategy starting from the initial configuration with the input word on the tape and the head in the initial state on the first letter of the input word. W.l.o.g. we assume that for each existential or universal state there are exactly two possible moves for each tape symbol. The accepting and rejecting state do not have any successors. More precisely, the transition function $\delta : Q \times \Sigma \to \mathcal{P}(Q \times \Sigma \times \{L, S, R\})$ has the property that the image of (q, a) has size two for all existential or universal states q and letters a. Hence, we regard δ as a function of the form

$$(Q \setminus \{\mathit{accept}, \mathit{reject}\}) \times \Sigma \times \{\mathsf{zero}, \mathsf{one}\} \to Q \times \Sigma \times \{L, S, R\}.$$

Furthermore, we can assume that it is impossible to repeat configurations in a computation of the ATM and that hence all computations end in an accepting or rejecting state.
It is well-known that the class of problems solvable by a k-exponentially space-bounded ATM coincides with the class $k+1$-EXPTIME [5]. So, let $\mathcal{T} = (Q, \Sigma, \delta, q_{init}, g)$ be an ATM satisfying the assumptions mentioned above. Assume that on an input word of length n the space used by $\mathcal{T}$ is bounded by $h(k, n)$. Recall that we defined $h(0, n) = n$ and $h(k+1, n) = 2^{h(k,n)} \cdot h(k, n)$ for all k and n. So, for each k, the function $h(k, n)$ is k-exponential in n.
Let $w \in \Sigma^n$ be an input word of length n. We will construct an MDP $\mathcal{M}$ and a Π_k^{QLTL}-formula $\mathsf{input\_accepted}$ of size polynomial in the size of $\mathcal{T}$ and n such that $\mathrm{Pr}_{\mathcal{M}}^{\max}(\mathsf{input\_accepted}) = 1$ if and only if $\mathcal{T}$ accepts w, i.e., if the existential player has a winning strategy on the initial configuration with w on the tape. Further, $\mathrm{Pr}_{\mathcal{M}}^{\max}(\mathsf{input\_accepted}) = 1$ will hold if and only if $\mathrm{Pr}_{\mathcal{M}}^{\max}(\mathsf{input\_accepted}) > 0$.
We define the extended tape alphabet to be

$$\Gamma = \Sigma \cup \Sigma \times Q \cup \{\mathit{blank}\}.$$

A configuration of the Turing machine $\mathcal{T}$ on input w can now be represented by a string of length $h(k, n)$ over the extended tape alphabet Γ that contains exactly one symbol from $\Sigma \times Q$. This symbol

marks the position of the head together with the current state. The symbol *blank* represents empty tape cells. The idea is to construct an MDP in which a scheduler can produce sequences over the extended tape alphabet separated by a marker *step* indicating a new step or *comp* indicating a new computation. A run of the MDP will consist of the concatenation of the encodings of potential computations. If the encoded computations are correct computations, a run will contain infinitely many finite encodings of computations. The formula we construct requires the scheduler to construct the correct initial configuration at the beginning of each computation. Afterwards, the scheduler has to construct the correct successor configurations in each step. The successor move here can be chosen by the scheduler if the state of a configuration is existential. If the state is universal, the successor move is chosen randomly. As soon as the accepting or rejecting state is reached, the scheduler has to end the computation and start a new one. If the scheduler can correctly construct all computations and always end in an accepting state, no matter which moves are randomly chosen in universal states, the formula holds. As almost surely all possible strategies for the universal player are played by chance at some point, the formula can only hold with positive probability, and then already with probability 1, if the input word w is accepted by the ATM $\mathcal{T}$.
The MDP $\mathcal{M}$, depicted in Figure 6 is quite simple: It consists of states *comp*, *step$_1$*, *step$_2$*, *step$_3$*, *step$_4$*, and one state for each extended tape symbol $\gamma \in \Gamma$. States are labeled with their name. Additionally, the states of the form *step$_i$* are labeled with *step* and have further additional lables: *step$_1$* is labeled with $\mathtt{zero}$ and E; *step$_2$* is labeled with $\mathtt{one}$ and E; *step$_3$* is labeled with $\mathtt{zero}$ and U; and *step$_4$* is labeled with $\mathtt{one}$ and U. The labels $\mathtt{zero}$ and $\mathtt{one}$ are used to indicate which of the two successor moves has been chosen. The labels E and U denote which player has made the choice of the move. The initial state is *comp*. In the states *comp*, and *step$_i$*, $i = 1, \ldots, 4$, there is one action enabled for each state in Γ, leading to that state with probability 1. In each state from Γ, there is an action to each state with probability 1 except for the states *step$_3$* and *step$_4$*. States *step$_3$* and *step$_4$* can be reached from each state from Γ via a randomized action leading to these states with probability $1/2$ each. These are the only probabilistic transitions.
We will now construct the formula $\mathtt{input\_accepted}$. The first requirement we impose on a run of the MDP is that a state labeled with *comp* or *step* is always followed by a block of exactly $h(k,n)$ symbols from Γ before the next state labeled with *comp* or *step*. We express this in the following formula $\mathtt{distance}$, again exploiting the Σ_k^{QLTL}-formulas $\varphi_{k,n}(p,q)$ from [21]:

$$\forall p \forall q \Big(\varphi_{k,n}(p,q) \to \\ \big[\Box\big(((\mathit{comp} \vee \mathit{step}) \wedge \bigcirc p) \to \bigcirc(\Gamma \, \mathrm{U}(q \wedge (\mathit{comp} \vee \mathit{step})))\big)\big]\Big).$$

Note that this formula is in Π_k^{QLTL} as the subformula $\varphi_{k,n}(p,q)$ which is in Σ_k^{QLTL} appears in the scope of one negation as the antecedent of an implication.
The next requirement is that the state *comp* is always followed by an encoding of the initial configuration. We write $w_1 \ldots w_n$ for the input word w. The requirement is expressed by the formula $\mathtt{initial}$:

$$\Box\big(\mathit{comp} \to \big[\bigcirc(w_1, q_{\mathit{init}}) \wedge \bigwedge_{2 \leqslant i \leqslant n} \bigcirc^i w_i \wedge \bigcirc^{n+1}(\mathit{blank}\, \mathrm{U}(\mathit{comp} \vee \mathit{step}))\big]\big).$$

Here (w_1, q_{init}) is the letter from the extended tape alphabet, that indicates that the head is on the first letter of the input word in the initial state. As n is given in unary, this formula is an LTL-formula polynomial in the size of the input.
The next requirement is that each block of extended tape symbols contains exactly one symbol from $\Sigma \times Q$ to ensure that it encodes a configuration. The formula $\mathtt{configuration}$ is:

$$\Box\big((\mathit{comp} \vee \mathit{step}) \to \bigcirc\big[\neg(\mathit{comp} \vee \mathit{step})\, \mathrm{U}(\Sigma \times Q \wedge \bigcirc(\neg \Sigma \times Q\, \mathrm{U}(\mathit{comp} \vee \mathit{step})))\big]\big).$$

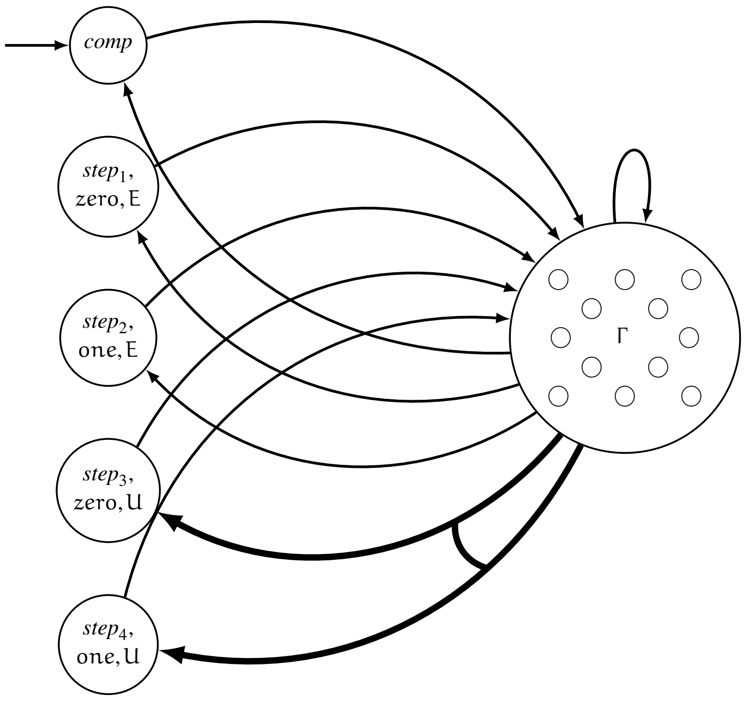


Figure 6 The MDP $\mathcal{M}$. The state depicted as Γ represents the behavior of each state $\gamma \in \Gamma$. I.e. from each state, there is one action to each state in Γ with probability 1. Further, from all states in Γ, there are actions leading to states *comp*, $step_1$ and $step_2$ with probability 1, as well as a randomized action (bold lines) leading to states $step_3$ and $step_4$ with probability $1/2$ each.

The additional label in the states labeled with *step* denotes which player, E or U, is allowed to move and which of the two possible successor moves $zero$ or one. If the universal player U is allowed to move, the decision is made randomly. The moves of the existential player E are determined by the scheduler. The formula $move$ checks whether the scheduler always chooses the right player to make the next move. Let *exists* denote all symbols of the extended tape alphabet containing an existential state from Q and *forall* denote all symbols containing a universal state. The formula $move$ is the following:

$$\Box(exists \to (\neg(comp \vee step)\, \mathsf{U}\, E)) \wedge \Box(forall \to (\neg(comp \vee step)\, \mathsf{U}\, U))$$

The crucial point to check now is that successive configurations are correct with respect to the transition function of the ATM $\mathcal{T}$ and the successor move, $zero$ or one, denoted in the state labeled *step* separating the two configurations. The (extended) content of a tape cell depends on the chosen move and on the content of the same cell in the previous configuration as well as its two neighbors because a tape cell denoted by a symbol of the extended tape alphabet can only change if the head was placed there or moves there. Let B denote the border of the tape, i.e., the set of the states labeled with *comp* or *step*. As the bit $zero$ or one determines the successor move, we can define two relations $\nabla_{zero}, \nabla_{one} \subseteq (\Gamma \cup B)^4$. A tuple of symbols $(\gamma_1, \gamma_2, \gamma_3, \rho)$ is in ∇_i if the content of a cell is ρ after move i if the cell itself contained γ_2 and its neighbors were γ_1 and γ_3 in the previous configuration. So, only γ_1 or γ_3 can be in B for a valid tuple. If ρ is not equal to γ_2, then one of the symbols γ_1, γ_2, and γ_3 has to contain the state of the ATM before the transition. Together with the knowledge whether move $zero$ or one was chosen, the new tape content ρ is then uniquely defined. In our encoding of a computation, the left neighbor of a cell in the previous configuration and the cell itself are placed $h(k,n)+2$ steps apart. The formula $\mathit{successor}$ checks the correctness of all successor configurations. We use universally quantified variables p and q that satisfy $\varphi_{k,n}(p,q)$. The cell at which p is placed is the cell whose successor has to be checked. The successor cell can be found at the position directly after q. The left neighbor of the cell marked with p satisfies $\bigcirc p$. Finally, the move that is chosen can be found by checking whether the next state labeled with *comp* or *step* is labeled with zero or one. If it is labeled with *comp* we are in the final configuration of a computation and no successor configuration has to be checked. The formula $\mathit{successor\_zero}$ checks whether the successor is correct if the chosen move is zero.

$$\forall p \forall q \Big(\varphi_{k,n}(p,q) \to \Big[\Box(\bigcirc(p \wedge \Gamma \wedge (\neg(\mathit{comp} \vee \mathit{step})\, \mathrm{U}\, \mathit{zero})) \to \bigvee_{(\gamma_1,\gamma_2,\gamma_3,\rho)\in\nabla_{\mathit{zero}}} (\gamma_1 \wedge \bigcirc\gamma_2 \wedge \bigcirc^2\gamma_3 \wedge \Diamond(q \wedge \bigcirc\rho)))\Big]\Big).$$

The formula $\mathit{successor\_one}$ works completely analogously with zero replaced by one. The formula $\mathit{successor}$ is simply $\mathit{successor\_zero} \wedge \mathit{successor\_one}$.
The last requirement on a valid sequence of encodings of computations we pose is that as soon as an accepting or rejecting state is reached, the computation is ended. This is indicated by the state *comp* following the configuration. Immediately afterwards a new computation is started. Let *final* be referring to all extended tape symbols that contain an accepting or rejecting state. The formula end requires that all configurations with a final state are followed by *comp*:

$$\Box(\mathit{final} \to (\neg \mathit{step}\, \mathrm{U}\, \mathit{comp})).$$

Finally, we can state that all computations encoded in a run lead to an accepting state. Let *reject* denote all extended tape symbols with a rejecting state. The formula $\mathit{input\_accepted}$ is the conjunction of all formulas we described:

$$\mathit{distance} \wedge \mathit{initial} \wedge \mathit{configuration} \wedge \mathit{move} \wedge \mathit{successor} \wedge \mathit{end} \wedge \Box\neg \mathit{reject}.$$

As all the formulas in the conjunction are contained in Π_k^{QLTL} this formula itself is contained in Π_k^{QLTL}.
We claim that $\Pr^{\max}_{\mathcal{M}}(\mathit{input\_accepted}) = 1$ iff the input word w is accepted by $\mathcal{T}$. Assume that w is accepted. So, the existential player has a winning strategy from the initial configuration. This strategy chooses a move for each configuration. It can be seen as a function from

$$\Gamma^{h(k,n)} \to \{\mathit{zero}, \mathit{one}\}.$$

All steps a scheduler has to take in order to satisfy $\mathit{input\_accepted}$ are uniquely determined by the history so far except for the choice between states step_1 and step_2 when a configuration with an existential state is completed. If the scheduler makes this choice of the next move according to the winning strategy and otherwise constructs correct successor configurations, all computations will lead to the accepting state no matter which moves are randomly chosen when it is the universal player's turn.
Assume now that the universal player has a winning strategy. As the length of possible computations is bounded by the number of possible configurations, in each of the infinitely many computations encoded in a run, there is a positive probability that the randomly chosen moves of the universal player agree with her winning strategy in each step. Hence, the probability that eventually a rejecting state is reached if all requirements on a correct encoding are satisfied is 1. So, $\Pr^{\max}_{\mathcal{M}}(\mathit{input\_accepted}) = 0$.
We also see that $\Pr^{\max}_{\mathcal{M}}(\mathit{input\_accepted}) = 1$ iff $\Pr^{\max}_{\mathcal{M}}(\mathit{input\_accepted}) > 0$.
Hence, checking for a Π_k^{QLTL}-formula whether the its maximal satisfiability probability is 1 (or > 0) in MDPs is $k+1$-EXPTIME-complete. ◀

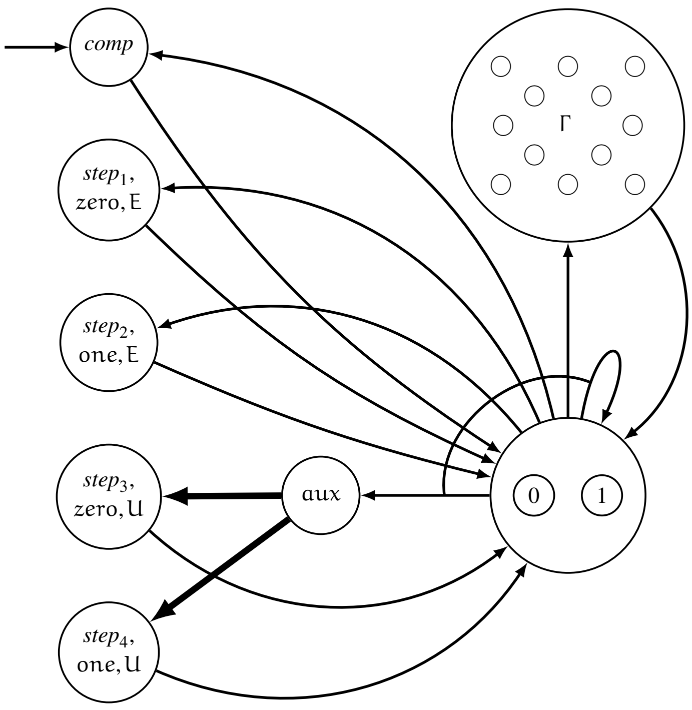


Figure 7 The MDP $\mathcal{M}$. The behavior is probabilistic except for the choice in the state aux. When entering the cluster of states Γ or the cluster with the two bits 0 and 1, one of the states in the cluster is chosen randomly. Further all states in Γ have only the outgoing transition randomly moving to 0 or 1. The state aux is only an auxiliary state for the graphical representation. That means that in states 0 and 1 two actions are enabled. The first moving randomly to any state except for *step*$_3$; the second moving randomly to any state except for *step*$_4$.

B.2 Proof of Theorem 4 for minimal satisfaction probabilities

Now, we prove the result for the minimal satisfaction probabilities.

▶ **Theorem 17.** *Given an MDP $\mathcal{M}$ and a Π_k^{QLTL}-formula φ, deciding whether* $\Pr^{\min}_{\mathcal{M}}(\varphi) = 1$ *and deciding whether* $\Pr^{\min}_{\mathcal{M}}(\varphi) > 0$ *are $k+1$-EXPTIME-complete for any $k \geqslant 1$.*

Proof. The upper bounds are obtained as in the proof of Theorem 4 directly via the construction of a deterministic Rabin automaton for a Π_k^{QLTL}-formula.
For the proof of the lower bound, let $\mathcal{T} = (Q, \Sigma, \delta, q_{init}, g)$ be an ATM that is k-exponentially space-bounded. More precisely, assume that on an input word of length n, the space used by $\mathcal{T}$ is bounded by $2^{h(k,n)} - 1$. Recall that we defined $h(0,n) = n$ and $h(k+1,n) = 2^{h(k,n)} \cdot h(k,n)$ for all k and n. So, for each k, the function $2^{h(k,n)} - 1$ is k-exponential in n. As in the proof of Theorem 4, we further assume that the transition function has the form

$$(Q \setminus \{accept, reject\}) \times \Sigma \times \{\mathsf{zero}, \mathsf{one}\} \to Q \times \Sigma \times \{L, S, R\}$$

and that it is impossible to repeat a configuration, so each computation ends in the accepting state *accept* or the rejecting state *reject* after finitely many steps.
As the minimal satisfaction probability corresponds to a statement about all schedulers, we cannot let a scheduler construct correct computations anymore. Instead potential sequences of configurations

are produced randomly. The scheduler now takes the role of the universal player trying to minimize the probability that eventually a correct accepting computation is produced. Checking that, e.g., eventually an encoding is produced in which the distances between the beginnings of the computation steps are correct now becomes more difficult. A pair of universally quantified variables p and q placed at the correct distance by the formula $\varphi_{k,n}(p,q)$ is not sufficient anymore. The reason is that the $\Diamond$-operator implicitly contains an existential quantification; for each pair of p and q a different starting point can be referred to by $\Diamond$. To circumvent this problem, we employ two ideas that we have seen in the hardness proof for Markov chains, Theorem 2, already: A binary counter of $(k-1)$-exponential length separating the content of the encoding and quantified variables as markers for violations to various conditions.

Again, we let $\Gamma = \Sigma \cup \Sigma \times Q \cup \{\textit{blank}\}$ be the extended tape alphabet. A computation will be represented similar to a tiling above by a function $f\colon \{1,\dots,2^{h(k,n)}-1\} \times \{0,\dots,m\} \to \Gamma$. Each row of the grid stands for one configuration. Such a function will then be encoded by a finite word of the following form:

$$\begin{aligned}
&\textit{comp},\ \overbrace{0,0,0,\dots,0}^{h(k-1,n)\text{ steps}},f(1,0),\ \overbrace{1,0,0,\dots,0}^{h(k-1,n)\text{ steps}},f(2,0),\ \overbrace{0,1,0,\dots,0}^{h(k-1,n)\text{ steps}},f(3,0),\ \dots,\\
&\qquad f(2^{h(k-1,n)}-1,0),\ \overbrace{1,1,1,\dots,1}^{h(k-1,n)\text{ steps}},\\
&\textit{step},\ \overbrace{0,0,0,\dots,0}^{h(k-1,n)\text{ steps}},f(1,1),\ \overbrace{1,0,0,\dots,0}^{h(k-1,n)\text{ steps}},f(2,1),\ \overbrace{0,1,0,\dots,0}^{h(k-1,n)\text{ steps}},f(3,1),\ \dots,\\
&\quad\vdots\\
&\textit{step},\ \overbrace{0,0,0,\dots,0}^{h(k-1,n)\text{ steps}},f(1,m),\ \overbrace{1,0,0,\dots,0}^{h(k-1,n)\text{ steps}},f(2,m),\ \overbrace{0,1,0,\dots,0}^{h(k-1,n)\text{ steps}},f(3,m),\dots,\\
&\qquad f(2^{h(k-1,n)}-1,m),\ \overbrace{1,1,1,\dots,1}^{h(k-1,n)\text{ steps}},\textit{comp}
\end{aligned}$$

The MDP $\mathcal{M}$ for the reduction, depicted in Figure 7, contains states *comp*, *step₁*, *step₂*, *step₃*, *step₄*, one state for each extended tape symbol $\gamma \in \Gamma$ as in the previous proof, and additionally two states 0 and 1. Starting from the state *comp*, the process randomly moves to one of the two states 0 and 1. In these states, two actions α_0 and α_1 are enabled. Action α_0 randomly leads to any state except for *step₄*; action α_1 to any state except for *step₃*. In this way, a scheduler can determine which of the states labeled with U indicating that it is the universal player's move to go to whenever it is probabilistically chosen that one of these states will be visited. From all other states, the process moves back to one of the states 0 and 1. A run of the MDP $\mathcal{M}$ almost surely is the concatenation of infinitely many finite words separated by the state labeled *comp*. These finite words are potential encodings of an accepting computation of $\mathcal{T}$.

The formula $\mathit{input\_accepted}'$ that we construct now expresses that eventually a correct accepting computation is encoded that respects the successor moves indicated by zero or one in the *step*-states. A correct computation also has to contain the correct labels E or U indicating which player's move it is after each configuration. In this way, a scheduler takes the role of the universal player U in a valid encoding of a computation.

The conditions for a valid encoding of a computation are very similar to the conditions for a valid encoding of tiling as in the proof of Theorem 2. Checking that the distance length of the binary counters is correct can be done completely analogously. We use a quantified variable d to mark violations. We only have to appropriately exchange the names of the propositions in the formula ψ_d used there. Then this formula expresses that d can only be placed at positions labeled *comp*, *step*,

or with $\gamma \in \Gamma$, that it can occur at most once between two states labeled *comp* and that a sequence of bits following a marked position is always of length not equal to $h(k-1,n)$. We denote the modified formula by ψ'_d. The formula $\mathtt{correct\_dist}'$ then becomes $\neg d \wedge \bigcirc(\neg d \, \mathrm{U} \, \mathit{comp})$. Evaluated at a state labeled *comp* it makes sure that no violation to the length of the counter is marked before the next state *comp*. If we now universally quantify d, all placements of d somewhere between any successive occurrences of *comp* where the correct length of the counters is not adhered to are considered. Similarly, we can adapt the formulas ψ_c and $\mathtt{correct\_counter}$ from the proof of Theorem 2 by minor changes exchanging the used atomic propositions appropriately. We denote the modified formulas by ψ'_c and $\mathtt{correct\_counter}'$.
The formula $\mathtt{config\_follows}$ expresses that there is exactly one symbol from $\Sigma \times Q$ before $\mathit{step} \vee \mathit{comp}$ holds:

$$\bigcirc\big[\neg(\mathit{step} \vee \mathit{comp}) \, \mathrm{U} (\Sigma \times Q \wedge \bigcirc(\neg \Sigma \times Q \, \mathrm{U} (\mathit{step} \vee \mathit{comp})))\big].$$

The formula $\mathtt{configuration}'$ evaluated at *comp* states that this condition holds on all encoded configurations before the next *comp*:

$$\mathtt{config\_follows} \wedge \bigcirc((\mathit{step} \rightarrow \mathtt{config\_follows}) \, \mathrm{U} \, \mathit{comp}).$$

The formula $\mathtt{initial}'$ checks that the correct initial configuration is encoded first:

$$\bigcirc((0 \vee 1) \, \mathrm{U} ((w_1, q_{init}) \wedge \bigcirc((0 \vee 1) \, \mathrm{U} (w_2 \wedge \cdots \wedge (w_n \wedge \bigcirc((0 \vee 1 \vee \mathit{blank}) \, \mathrm{U} \, \mathit{step}) \ldots).$$

The formula $\mathtt{end}'$ using *final* to denote an accepting or rejecting state makes sure that the computation ends after a final state:

$$\big[\mathit{final} \rightarrow (\neg \mathit{step} \, \mathrm{U} \, \mathit{comp})\big] \, \mathrm{U} \, \mathit{comp}.$$

Let again *accept* denote letters from Γ containing an accepting state. Then, $\mathtt{accept}'$ is the formula

$$\bigcirc(\neg(\neg \mathit{accept} \, \mathrm{U} \, \mathit{comp}))$$

stating that an accepting state occurs before the next *comp*. Next, we make sure that the correct player determines the next move by defining the formula $\mathtt{move}'$ using *exists* and *forall* as before to denote an extended tape alphabet with an existential or universal state, respecively:

$$\big[\mathit{exists} \rightarrow (\neg \mathit{step} \, \mathrm{U} \, \mathsf{E})\big] \, \mathrm{U} \, \mathit{comp} \wedge \big[\mathit{forall} \rightarrow (\neg \mathit{step} \, \mathrm{U} \, \mathsf{U})\big] \, \mathrm{U} \, \mathit{comp}.$$

Finally, we have to make sure that the successor configurations are correct. We use variables v_1 and v_2 to mark a cell and its incorrect successor. For the correct placement of these markes – at most once in the correct order between two states labeled *comp*, only on extended tape symbols, and at the correct distance – we reuse the formula ψ_v from the proof of Theorem 2. After the necessary renaming of propositions in the formula, we obtain the formula ψ'_v. This formula uses the existentially quantified variable b to encode an additional counter. Nevertheless ψ'_v is in Σ_k^{QLTL}. For the transition relation, we also need markers on the neighboring tape cells (or the borders marked by *step* or *comp* which we again denote by B in case v_1 is placed at the first or last cell of a configuration). We introduce universally quantified variables u_1 and u_2. The Formula ψ_u makes sure these are always placed on these neighboring cells:

$$\Box\big[((B \vee \Gamma) \wedge \bigcirc((0 \vee 1) \, \mathrm{U} \, v_1)) \rightarrow u_1\big] \wedge \Box\big[v_1 \rightarrow (\bigcirc((0 \vee 1) \, \mathrm{U} \, u_3))\big].$$

We again define the two relations $\nabla_{zero}, \nabla_{one} \subseteq (\Gamma \cup B)^4$. A tuple of symbols $(\gamma_1, \gamma_2, \gamma_3, \rho)$ is in ∇_i if the content of a cell is ρ after move i if the cell itself contained γ_2 and its neighbors were γ_1

and γ_3 in the previous configuration. The formula $\mathsf{successor\_zero}'$ states that the cells marked by u_1, v_1, u_2, and v_3 satisfy the relation ∇_{zero} if the move chosen at the corresponding step is $zero$:

$$\bigcirc\left[\neg comp\,\mathrm{U}(v_1 \wedge (\neg step\,\mathrm{U}\,zero))\right] \rightarrow$$
$$\Big[\bigwedge_{(\gamma_1,\gamma_2,\gamma_3,\rho)\in\nabla_{zero}} \neg u_1\,\mathrm{U}(u_1 \wedge \gamma_1) \wedge \neg v_1\,\mathrm{U}(v_1 \wedge \gamma_2) \wedge \neg u_2\,\mathrm{U}(u_2 \wedge \gamma_3) \wedge \neg v_2\,\mathrm{U}(v_2 \wedge \rho)\Big].$$

The formula $\mathsf{successor\_one}'$ again works complete analogously. Put together, we obtain the formula $\mathsf{successor}'$ that simply is $\mathsf{successor\_zero}' \wedge \mathsf{successor\_one}'$.
This completes the list of subformulas for the formula $\mathsf{input\_accepted}'$:

$$\forall d \forall c \forall v_1 \forall v_2 \forall u_1 \forall u_2$$
$$\big((\psi'_d \wedge \psi'_c \wedge \psi'_v \wedge \psi_u) \rightarrow \Diamond\big(comp \wedge \mathsf{correct\_dist}' \wedge \mathsf{correct\_counter}' \wedge$$
$$\mathsf{initial}' \wedge \mathsf{configuration}' \wedge \mathsf{move}' \wedge \mathsf{successor}' \wedge \mathsf{end}' \wedge \mathsf{accept}'\big)\big).$$

It remains to show that $\mathrm{Pr}^{\min}_{\mathcal{M}}(\mathsf{input\_accepted}') = 1$ iff the existential player has a winning strategy iff $\mathrm{Pr}^{\min}_{\mathcal{M}}(\mathsf{input\_accepted}') > 0$. The argument goes as before: If there is a winning strategy for the existential player, a correct accepting computation no matter which moves the scheduler as the universal player chooses is encoded with probability 1. On the encoding of a correct computation, no markers that satisfy $\psi'_d \wedge \psi'_c \wedge \psi'_v \wedge \psi_u$ can be placed. Also all LTL-expressible conditions hold and hence the formula holds with probability 1 under any scheduler.
If, however, there is a winning strategy for the universal player, the scheduler can follow this strategy. Each finite word between two states labeled *comp* then violates one condition. Either this violation is already detected by the LTL-expressible conditions, or a universally quantified marker indicating a violation of the remaining conditions can be placed on this finite word. So, no suffix satisfies the whole formula after the $\Diamond$-operator and the minimal satisfaction probability of $\mathsf{input\_accepted}'$ is 0. ◀

C Argument for the restriction to one quantifier in the proof of Theorem 10

We sketch that Π_1^{QLTL}-model checking can be reduced to model checking of formulas of the form $\forall x \varphi$ with only one quantifier in Markov chains and MDPs. Let $\mathcal{M}$ be an MDP with state space S and let ϑ be a Π_1^{QLTL}-formula of the form $\forall x_0 \dots \forall x_{n-1} \psi$. We construct an MDP $\mathcal{M}'$ by replacing each state s in S by a chain $s_0, \dots, s_{n-1}$ with transition from s_i to s_{i+1} for $i < n-1$. From the last state s_{n-1} of such a chain, the same transtions to states t_0 are available as between s and t in $\mathcal{M}$. States of the form s_0 are additionally labeled with #. We can now mimic the variables x_i by replacing them by $\bigcirc^i x$. In order to do that we translate the formula ψ by the following translation function T:

- $\mathsf{T}(a) = a$ for $a \in \mathsf{AP}$,
- $\mathsf{T}(x_i) = \bigcirc^i x$,
- $\mathsf{T}(\neg\chi) = \neg\mathsf{T}(\chi)$,
- $\mathsf{T}(\chi_1 \wedge \chi_2) = \mathsf{T}(\chi_1) \wedge \mathsf{T}(\chi_2)$,
- $\mathsf{T}(\bigcirc\chi) = \bigcirc^n \chi$,
- $\mathsf{T}(\chi_1\,\mathrm{U}\,\chi_2) = (\neg\# \vee \mathsf{T}(\chi_1))\,\mathrm{U}(\# \wedge \mathsf{T}(\chi_2))$.

So, also in the formula the transition from a step in the original MDP $\mathcal{M}$ to n steps through a chain in $\mathcal{M}'$ is respected after the translation. The marker # at the beginning of the chains makes sure that in the U-operator only the first positions in a chain are considered. Now, it is not hard to see that $\mathrm{Pr}^{\mathfrak{S}}_{\mathcal{M},s_{init}}(\forall x_0 \dots \forall x_{n-1}\psi) = \mathrm{Pr}^{\mathfrak{S}'}_{\mathcal{M},s_{init,0}}(\forall x.\mathsf{T}(\psi))$ where $\mathfrak{S}'$ and $\mathfrak{S}$ are schedulers that behave the same in $\mathcal{M}$ and $\mathcal{M}'$ when considereing the chains in $\mathcal{M}'$ as single states .
So, the restriction to one quantified variable does not decrease the complexity of the model-checking problems for MDPs or Markov chains.