On the Satisfiability of Some Simple Probabilistic Logics

Souymodip Chakraborty Joost-Pieter Katoen
RWTH Aachen University, Germany
{chakraborty, katoen}@cs.rwth-aachen.de

Abstract

This paper shows that the satisfiability problems for a bounded fragment of probabilistic CTL (called bounded PCTL) and an extension of the modal μ-calculus with probabilistic quantification over next-modalities (called PμTL) are decidable. For bounded PCTL we provide an NEXPTIME-algorithm for the satisfiability problem and show that the logic has a *small model property* where the model size is independent from the probability bounds in the formula. We show that the satisfiability problem of a simple sub-logic of bounded PCTL is PSPACE-complete. We prove that PμTL has a small model property and that a decision procedure using 2 player parity games can be employed for the satisfiability problem of PμTL. These results imply that PμTL and qualitative PCTL formulas with only thresholds >0 and $=1$—are incomparable.

Categories and Subject Descriptors F.4.1 [*Mathematical Logic and Formal Languages*]: Mathematical Logic.; G.3 [*Probability and Statistics*]: Markov Processes.

1. Introduction

Background and context. Probabilistic versions of CTL, such as PCTL [13] and PCTL* [2], are logics to express quantitative properties of models such as Markov chains (MCs) and Markov decision processes (MDPs). These logics are quite popular in the field of probabilistic verification. Whereas model checking amounts to checking whether a given finite structure satisfies a logic formula, satisfiability focuses on the problem whether a given formula f has a model, i.e., whether there exists a (possibly infinite) structure satisfying f. For logics such as PCTL, PCTL*, as well as the more expressive probabilistic μ-calculus [18], the model-checking problem is known to be decidable. In contrast, the satisfiability problem for these logics turns out to be a much more difficult endeavour. The satisfiability problems for PCTL and the probabilistic μ-calculus are long-standing open problems. Results so far are restricted to logical fragments or by considering variations of the satisfiability problem. For *qualitative* PCTL—a fragment of PCTL in which all probability bounds are $=1$ or >0—a decision procedure has been provided by Hart and Sharir [14], and showed that qualitative PCTL does not possess the finite model property. Brázdil *et al.* [5] proved qualitative PCTL satisfiability to be EXPTIME-complete. Bertrand *et al.* [1] encoded the *bounded* satisfiability problem for PCTL formulas, i.e., does there exist a "simple" model for a PCTL formula, as an SMT problem. (Here a simple model means a model of a given number of states with rational transition probabilities.) Recently, Weiwan *et al.* [17] showed the decidability of PμTL, an extension of the modal μ-calculus with a *probabilistic next-modality* by providing a 2-EXPTIME decision procedure.

Bounded PCTL. In our attempts to tackle the satisfaction problem, we have found that the difficulty of providing a decision procedure for probabilistic logics primarily lies in the presence of quantified recursively defined path formulas. This includes formulas of the form $[a \,\mathsf{U}\, b]_{\geq 1/2}$ expressing that a b-state is to be reached via a-states with probability at least $1/2$. This paper therefore considers probabilistic logics in which syntactic restrictions are imposed on recursively defined path formulas. We consider the satisfiability problem for two logical fragments (and some of their sub-fragments): bounded PCTL and PμTL. Bounded PCTL is a PCTL fragment in which until-modalities are bounded by the number of steps that can be taken; e.g., $[a \,\mathsf{U}^n b]_{\geq 1/2}$ expresses that a b-state is to be reached within n steps. Bounded PCTL thus abandons the unbounded until-modality. We show that the logic has a finite (tree) model property where the size of the model is independent from the probability bounds (like $\geq 1/2$) in the formula. To study the computational complexity of bounded PCTL satisfiability, we first show that the satisfiability problem of a simple sub-logic of bounded PCTL that (besides propositional logic) only contains nested quantified next-modalities is PSPACE-complete. The main result is an NEXPTIME-algorithm for the entire bounded PCTL satisfiability. This is based on a novel variable elimination method for solving the satisfiability problem for specific class of formulas in the theory of the reals. Finally, we show that the satisfiability of bounded PCTL-formula f is EXPTIME-hard in the encoding of f.

Simple probabilistic μ-calculus. In the second part of the paper we extend our results to a logic with recursion: PμTL, a version of the modal μ-calculus equipped with a probabilistic next-modality. This logic allows to express formulas like $\nu Y.\, \big(a \wedge [\mathsf{X}\, Y]_{\geq 1/3}\big)$ asserting that a holds and continues to hold with at least probability $1/3$ in the next state. The logics PμTL and (bounded) PCTL have incomparable expressive power. PμTL is subsumed by various probabilistic logics with recursion, such as μ-PCTL [7], Mio's probabilistic μ-calculus with independent product [18] and the μ^p-calculus [7]. PμTL has recently been introduced in [17]

where the satisfiability problem was shown to be decidable only for *finite models*. A 2-EXPTIME decision procedure was provided by reducing the problem to the emptiness problem of certain alternating tree automata. We extend this with the following results. We show that PμTL has a *small model property* in the sense that every satisfiable formula f has a model of size exponential in $|f|$ (and has a bounded out-degree at most $|f|+1$. These results imply (using [5]) that PμTL and qualitative PCTL are incomparable[1]. Our constructive proof exploits the results on (ordinary) parity games for the satisfiability problem of PμTL. Similar to results for the modal μ-calculus, we obtain that a PμTL-formula f is satisfiable iff player zero has a winning strategy in the game arena that corresponds to f. Using these results we establish that—in contrast to PCTL [5] with a priory fixed model size —every satisfiable PμTL-formula has a rational model, i.e., a model with rational probabilities only.[2] Our results show that one needs to solve a parity game of exponential size in order to decide PμTL satisfiability. This is the strongest possible bound since PμTL can encode μ-calculus, and our result shows that satisfiability of PμTL lies in the same complexity class as the satisfiability of μ-calculus. This sharpens the 2-EXPTIME complexity in [17] that used alternating tree automata, and the result is no longer restricted to finite models.

Organisation of this paper. Section 2 presents preliminary notions such as weighted covers and Markov chains. Section 3 considers bounded PCTL and some sub-logics, their satisfiability and complexity. Section 4 introduces PμTL, pre-models, and presents a decision problem of PμTL satisfiability using 2-player parity games. Section 5 concludes the paper. Omitted proof details are provided in the Appendix.

2. Preliminaries

2.1 Weighted covers

Let X^Y be the functions from the set Y to the set X.

Definition 1 ((Weighted) cover)**.** Let H be a set of objects. A *cover* c is a set of sets of objects of H, such that $\bigcup_{e\in c} e = H$. The cardinality of c is called the *width* of the cover c. A *weighted cover* of H is a cover c with a mapping $w : c \to (0, 1]$, such that $\sum_{e\in c} w(e) = 1$.

Proposition 1. A set of cardinality n has at most $\frac{2^{n\cdot(k+1)}-2^n}{2^n-1}$ different covers of width at most k.

Proof. Let H be a set with $|H| = n$, and c a cover of H with width $i \le k$. An object of H can be placed in every set of c. Given that c covers H, there are at most 2^i-1 possibilities. This holds for every object of H. The number of different covers of width i thus is $(2^i-1)^n$ (or, $\leqslant 2^{n\cdot i}$). Summing over all $1 \le i \le k$ gives the desired bound. □

For weighted cover (c, w) of $H = \{o_1, \ldots, o_n\}$, let $H(o_i) = \{e \in c : o_i \in e\}$ and $w(o_i) = \sum_{e\in H(o_i)} w(e)$.

Proposition 2 (Dual of Helly's theorem)**.** Let T be a countable set of vectors in $\mathbb{R}^n$. For every vector $\vec{v}$ being a convex combination of vectors from T, there exists a set $T' \subseteq T$ such that $\vec{v}$ is a convex combination of vectors from T' and $|T'| \le n+1$.

Proof. The vector $\vec{v}$ is inside the convex polytope defined by T. A *triangulation* of a polytope is a partitioning of the space inside the convex polytope using $(n+1)$-simplexes (tetrahedrons) in n-dimensions. Such a triangulation always exists even if the convex polytope is generated by a countable set of points. Thus, $\vec{v}$ is inside (or on) some $n+1$-simplex whose vertices are in $T' \subseteq T$. Which implies, $\vec{v}$ can also be defined as a convex combination of vectors in T'. □

2.2 Markov chains

Let $\mathbb{R}_+$ denote the set of non-negative reals. Let $\mathcal{D}_X$ be the set of probability distributions over the set X where $\vec{d} \in \mathcal{D}_X$ iff $\vec{d} \in \mathbb{R}_+^X$ and $\vec{d}^T \cdot \vec{1} = 1$, where $\vec{1}$ denotes the vector only containing ones.

Definition 2 ((Labelled) Markov chain)**.** A (labelled) Markov chain (MC) M is a quintuple $(S, P, \mathsf{AP}, L, s_{in})$ where S is a countable set of states, $P(s) \in \mathcal{D}_S$ for all $s \in S$, AP is a set of atomic propositions, $L : S \to 2^{\mathsf{AP}}$ is a labelling function, and $s_{in} \in S$ is the initial state.

An *infinite path* w through MC M is a sequence of states $w = \{w_i\}_{i\ge 0}$, where for all $i \ge 0$, $P(w_i, w_{i+1}) > 0$. Let $\mathsf{path}(s)$ denote the set of (finite or infinite) paths starting from state s. For a path w, let $\mathsf{last}(w)$ denote the last state of w if this exists (i.e., if w is finite) and $|w|$ denote the length of w. Let $\mathsf{succ}(s) = \{t : P(s, t) > 0\}$ be the set of successors of state s. A probability measure $(\Omega_s, \mathcal{F}, \Pr)$ where Ω_s is the set of infinite paths from state s is obtained by a standard cylinder set construction [3]. Details are omitted here.

3. Bounded PCTL

This section is concerned with the satisfiability problem for bounded PCTL, i.e., PCTL without unbounded until-modalities. We first show that the satisfiability problem for this PCTL fragment is decidable, and is in NEXPTIME in the *size* of the formula, and EXPTIME-hard in its *encoding*. In addition, we consider the complexity for the satisfiability problem for some other bounded fragments of PCTL.

3.1 Bounded PCTL: syntax and semantics

Probabilistic CTL [13], or PCTL for short, is a branching time temporal logic interpreted on infinite probabilistic computation trees obtained by unfolding Markov chains. We consider a bounded fragment of PCTL, obtained by omitting the unbounded until-modality. Let $a \in \mathsf{AP}$ be a proposition, $p \in \mathbb{Q} \cap [0, 1]$ a probability, $\succ \in \{>, \ge\}$ a binary comparison operator, and $n \in \mathbb{N}$.

Definition 3 (Bounded PCTL syntax)**.** The syntax of bounded PCTL is given by the following BNF grammar:

$$f ::= a \mid \sim f \mid f \wedge f \mid [g]_{\succ p} \quad \text{and} \quad g ::= \mathsf{X} f \mid f \,\mathsf{U}^n f.$$

The semantics of bounded PCTL is defined on labelled Markov chains; for state sentence f this is standard, where the semantics of $[g]_{\succ p}$ is given for state s by:

$$s \models [g]_{\succ p} \text{ iff } \Pr\{w \in \Omega_s : w \models g\} \succ p$$

where Pr is the probability measure defined on cylinder sets of the Markov chain to which state s belongs. The semantics of the next-operator is standard and omitted. For an infinite

[1] This contradicts the result in [17] that qualitative PCTL is strictly less expressive than qualitative PμTL for finite models.

[2] We don't know that such a property holds for PCTL in general.

path $w = (w_0, w_1, \ldots)$, the semantics of the bounded until-operator is defined by:

$$w \models f_1 \,\mathsf{U}^n f_2 \text{ iff } \begin{cases} w_0 \models f_2 & \text{for } n = 0 \\ w_0 \models f_2 \text{ or } (w_0 \models f_1 \text{ and } \\ \quad w_1 \models f_1 \,\mathsf{U}^{n-1} f_2) & \text{for } n > 0 \end{cases} \tag{1}$$

A path thus satisfies $f_1 \,\mathsf{U}^n f_2$ whenever an f_2-state is reached within n steps while all preceding states satisfy f_1. Let $\mathsf{F}^n g$ denote $\mathsf{true}\,\mathsf{U}^n g$ and $\mathsf{G}^n g \equiv {\sim}\mathsf{F}^n {\sim} g$.

Example 1. The sentence $[\mathsf{F}^3[\mathsf{G}^{10}\, b]_{=1}]_{>1/3}$ in bounded PCTL expresses that the probability to reach a state within three steps from which almost surely b holds for at least the next ten steps exceeds $1/3$.

3.2 Bounded PCTL satisfiability: decidability

An MC whose underlying graph is a tree is called a *tree* Markov chain. Every MC M induces a tree MC M_s by unfolding M from state s. Let $M_{s,n}$ be the MC obtained from M by unfolding n steps starting from state s. The leaves of $M_{s,n}$ are absorbing by equipping them with self-loops of probability one. $M_{s,n}$ is thus a finite tree MC of depth n rooted at s. For bounded PCTL-sentence f, let $\mathsf{ord}(f)$ be recursively defined as follows:

$$\begin{array}{rcl} \mathsf{ord}(a) & = & 1 \text{ for } a \in \mathsf{AP} \\ \mathsf{ord}({\sim}f) & = & \mathsf{ord}(f) \\ \mathsf{ord}(f_1 \wedge f_2) & = & \max\{\mathsf{ord}(f_1), \mathsf{ord}(f_2)\} \\ \mathsf{ord}([\mathsf{X}\, f]_{\succ p}) & = & 1 + \mathsf{ord}(f) \\ \mathsf{ord}([f_1 \,\mathsf{U}^n f_2]_{\succ p}) & = & n + \max\{\mathsf{ord}(f_1), \mathsf{ord}(f_2)\}. \end{array}$$

It follows by a straightforward structural induction on f, that the satisfiability of bounded PCTL over finite trees obey the monotonicity property, i.e., $M_{s,n} \models f$ implies $M_{s,m} \models f$ for every $m \geqslant n \geqslant \mathsf{ord}(f)$. This provides the basis for the following result.

Proposition 3. For bounded PCTL-sentence f and MC M: $M, s \models f$ iff $M_{s,n}, s \models f$ with $n = \mathsf{ord}(f)$.

Proof. If $M_{s,n}, s \models f$ with $n = \mathsf{ord}(f)$ then obviously $M, s \models f$. For the other direction, we proceed as follows. Assume $M, s \models f$ and let $n = \mathsf{ord}(f)$. We prove that $M_{s,n}, s \models f$ by induction on the structure of f.

1. $f = a$. Then, $n = 1$. By definition, $M_{s,n}$ consists of a single node s equipped with a self-loop. Given $M, s \models f$ it follows $a \in L(s)$. Hence, $M_{s,n}, s \models f$.
2. $f = f_1 \wedge f_2$. Let $n_1 = \mathsf{ord}(f_1)$ and $n_2 = \mathsf{ord}(f_2)$. By induction hypothesis, $M_{s,n_1}, s \models f_1$ and $M_{s,n_2}, s \models f_2$. As $n \geq n_1$ and $n \geq n_2$, by monotonicity, $M_{s,n}, s \models f_1$ and $M_{s,n}, s \models f_2$. Thus, $M_{s,n}, s \models f$.
3. $f = {\sim}g$. By induction hypothesis, $M_{s,n}, s \not\models g$, thus $M_{s,n}, s \models f$.
4. $f = [\mathsf{X}\, g]_{\succ p}$. Let $S' = \{t : M, t \models g \text{ and } P(s,t) > 0\}$ and $m = \mathsf{ord}(g)$, i.e., $n = m{+}1$. By induction hypothesis, $M_{t,m}, t \models g$ for every $t \in S'$. As $t \in S'$ is a direct successor of s, $M_{t,m}$ is a subtree of $M_{s,m+1}$. As $M, s \models f$, we have $\sum_{t \in S'} P(s,t) \succ p$. Thus, $M_{s,m+1}, s \models f$.
5. $f = [f_1 \,\mathsf{U}^k f_2]_{\succ p}$. Let $n_1 = \mathsf{ord}(f_1)$ and $n_2 = \mathsf{ord}(f_2)$. If $M, s \models f_2$, then the statement follows from the induction hypothesis and monotonicity. Assume $M, s \not\models f_2$. Let w be a path starting in s with $w \models f_1 \,\mathsf{U}^k f_2$. Thus, $M, w_i \models f_2$ for some $0 < i \leq k$, and, $M, w_j \models f_1$, for every $j < i$. By the induction hypothesis, $M_{w_i,n_2}, w_i \models f_2$ and $M_{w_j,n_1}, w_j \models f_1$ for every $j < i$. By monotonicity, $M_{w_{i-1},n'}, w_{i-1} \models f_1$ and $M_{w_i,n'}, w_i \models f_2$ where $n' = \max\{1{+}\mathsf{ord}(f_1), \mathsf{ord}(f_2)\}$. For $n = n'{+}k$, $M_{w_{i-1},n'}$ is a sub-tree of $M_{s,n}$, therefore $M_{s,n}, w_{i-1} \models f_1$ and $M_{s,n}, w_i \models f_2$. This is true for any path $w \models f_1 \,\mathsf{U}^k f_2$. Thus, $M_{s,n}, s \models f$. □

The set $\mathsf{sub}(f)$ of *sub-formulas* of bounded PCTL-formula f is defined by:

$$\begin{array}{rcl} \mathsf{sub}(a) & = & \{a\} \\ \mathsf{sub}({\sim}f) & = & \{{\sim}f\} \cup \mathsf{sub}(f) \\ \mathsf{sub}(f_1 \wedge f_2) & = & \{f_1 \wedge f_2\} \cup \mathsf{sub}(f_1) \cup \mathsf{sub}(f_2) \\ \mathsf{sub}([\mathsf{X}\, f]_{\succ p}) & = & \{[\mathsf{X}\, f]_{\succ p}\} \cup \mathsf{sub}(f) \\ \mathsf{sub}([f_1 \,\mathsf{U}^n f_2]_{\succ p}) & = & \{[f_1 \,\mathsf{U}^n f_2]_{\succ p}\} \cup \mathsf{sub}(f_1) \cup \mathsf{sub}(f_2). \end{array}$$

Similar to Theorem 4 for PμTL, we obtain:

Proposition 4. Every satisfiable bounded PCTL-sentence f has a tree model with bounded out-degree at most $|\mathsf{sub}(f)|{+}1$.

Proof. Let M be a model of f. As the statement trivially holds for propositional formulas, we focus on path sentences. Consider state s in M and let $H = \{g \in \mathsf{sub}(f) : s \models g\}$. Assume w.l.o.g. that no two sub-sentences are syntactically identical. Let $\{1, \ldots, n\}$ be an enumeration of H, i.e., each formula in $\mathsf{sub}(f)$ is assigned a unique index. Assume s has more than $n{+}1$ descendants, i.e., $\mathsf{succ}(s) = \{t_1, \ldots, t_k\}$ for $k > n{+}1$. Let for path sentence g, $\Pr(s \models g)$ abbreviate $\Pr\{w \in \Omega_s : w \models g\}$. Define the vectors $\{\vec{s}, \vec{t}_1, \ldots, \vec{t}_k\}$ in the Euclidean space $[0,1]^n$ as follows:

1. for $[\mathsf{X}\, g]_{\succ p}$ with index i, $\vec{s}(i) = q$ where $\Pr(s \models \mathsf{X}\, g) = q$, and $t(i) = 1$ if $t \models g$ else $t(i) = 0$, for each $t \in \mathsf{succ}(s)$.
2. for $[f_1 \,\mathsf{U}^k f_2]_{\succ p}$ with index i and $s \not\models f_2$, $\vec{s}(i) = q$ where $\Pr(s \models f_1 \,\mathsf{U}^k f_2) = q$, and $t(i) = q$ with $\Pr(t \models f_1 \,\mathsf{U}^{k-1} f_2) = q$, for each $t \in \mathsf{succ}(s)$.
3. for any other index i, $\vec{s}(i) = \vec{t}(i) = 0$.

For the semantics of a path sentence of the form $[g]_{\succ p}$, we obtain the following relation:

$$\vec{s} \;=\; \sum_{t \in \mathsf{succ}(s)} P(s,t) \cdot \vec{t}.$$

That is, $\vec{s}$ is a linear combination of the vectors $\{\vec{t}_1, \ldots, \vec{t}_k\}$. By Proposition 2 (see page 2), there exists a set $G \subseteq \mathsf{succ}(s)$ with $|G| \leqslant n{+}1$ and a distribution $P'(s)$ such that:

$$\vec{s} \;=\; \sum_{t \in G} P'(s,t) \cdot \vec{t}.$$

It is easy to see that using G as set of direct successors (rather than $\mathsf{succ}(s)$) s still satisfies H. Applying this procedure to every state of M yields a model with out-degree at most $n{+}1$. □

Propositions 3 and 4 are the ingredients to obtain a small model theorem for bounded PCTL.

Theorem 1. Every satisfiable bounded PCTL-sentence f has a finite tree model of depth $\mathsf{ord}(f)$ and out-degree $|\mathsf{sub}(f)|{+}1$.

Let the *size* of bounded PCTL-sentence f be defined as $\mathsf{size}(f) = |\mathsf{ord}(f)|{+}|\mathsf{sub}(f)|$. The small model theorem thus asserts that for every satisfiable sentence f, there exists a tree MC model of f whose number of nodes is exponential in $\mathsf{size}(f)$ (but not the space needed to encode f).

3.3 Bounded PCTL satisfiability: complexity

In this section, we analyse the theoretical complexity of the satisfiability problem for bounded PCTL. We do so by first analysing the complexity for the satisfiability problem of the sub-logic Px_ω that supports arbitrary nesting of next-modalities but has no bounded until-modality. Subsequently, we treat bounded PCTL.

The PCTL-fragment Px_ω is defined as follows. For $n \in \mathbb{N}$, we inductively define the sub-logic Px_n:

$$\begin{array}{lll} \mathsf{Px}_0 & : & f ::= a \mid {\sim}f \mid f \wedge f \\ \mathsf{Px}_n & : & f ::= a \mid {\sim}f \mid f \wedge f \mid g \mid [\mathsf{X}\, g]_{\succ p} \text{ for } n > 0, \end{array}$$

where $a \in \mathsf{AP}$, $p \in \mathbb{Q} \cap [0,1]$, and $g \in \mathsf{Px}_{n-1}$. With little abuse of notation, let Px_n denote the set of all sentences of the logic Px_n. Let Px_ω be the set of sentences with an unbounded number of nested next modalities.

Proposition 5. The satisfiability problem for Px_ω is in PSPACE.

Proof. We show that the satisfiability problem for sentences in Px_n is in Σ_n^P of the polynomial-time hierarchy. Let T_n be a non-deterministic Turing machine (NTM) with an oracle Ω_{n-1}. Oracle Ω_n can foretell whether a set of sentences in Px_n is satisfiable[3]. W.l.o.g. we assume that Px_n sentences are in negated normal form. Let H be the set of Px_n sentences that the input to NTM T_n. T_n proceeds as follows: First, H is replaced by $\mathsf{closure}(H)$ (Alg. 1). This can be done in linear time in $|H|$.

Secondly, if $H \cap \mathsf{Px}_0$ is unsatisfiable, then T_n rejects. Otherwise, T_n selects a weighted cover (see Def. 1) (c, w) of $\{g : [\mathsf{X}\, g]_{\succ p} \in H\}$ with

1. $w(g) \succ p$ for each $[\mathsf{X}\, g]_{\succ p} \in H$, and
2. $\bigwedge_{g \in G} g \not\equiv$ false for each $G \in c$.

By Proposition 4, we restrict to covers whose widths are at most $|H|+1$. Checking (1) can be done by solving linear equations, and (2) satisfiability of formulas for each $G \in c$ is delegated to the oracle Ω_{n-1}. This is possible since the set G only contains sentences in Px_{n-1}. The NTM T_n accepts if such a weighted cover exists, else it rejects.

The correctness of the above algorithm is straightforward. The algorithm generates a model (and accepts H) iff H is satisfiable. We omit the details. The satisfiability of a set of Px_n sentences can thus be solved by an NTM with an oracle Ω_{n-1} in polynomial time. Hence, the satisfiability problem for Px_ω is in $\mathsf{NP}^{\mathsf{NP}^{\mathsf{NP}^{\cdots}}}$, and hence in PSPACE. □

PSPACE-hardness is proven by exploiting the fact that the satisfiability of modal formulas in the K-logic is PSPACE-hard [16]. A detailed reduction is provided in Appendix 6.1.

Proposition 6. The satisfiability problem for Px_ω is PSPACE-hard.

We now consider the complexity of the satisfiability problem for bounded PCTL. We use the following machinery to solve the satisfiability problem.

Proposition 7. For finite tree T and bounded PCTL-formula f, it can be decided in NP whether $M \models f$ for some tree MC M with T as its underlying graph.

[3] See [15] for background information on oracle Turing machines and the polynomial time-hierarchy.

Algorithm 1 closure(H)

```
1: for each f ∈ H do
2:     if f = a ∈ AP or f = ∼a then skip
3:     if f = f1 ∧ f2 then H := (H \ {f}) ∪ {f1, f2}
4:     if f = f1 ∨ f2 then
5:         H := (H \ {f}) ∪ {fi}, where i = 1 or 2
6:     if f = [X g]≻p then skip
7:     if f = [f1 U^0 f2]≻p then H := (H ∪ {f2}) \ {f}
8:     if f = [f1 U^n f2]≻p and n > 0 then
9:         either H := (H ∪ {f2}) \ {f} or H := H ∪ {f1}
10: end for
11: return H
```

Algorithm 2 label(s)

```
1: if L(s) ⊆ Px0 then
2:     return true iff L(s) is satisfiable
3: else L(s) = closure(L(s))
4: end if;
5: for each f ∈ L(s) do
6:     if f = [X g]≻p then
7:         choose non-deterministically S' ⊆ succ(s)
8:         for each t ∈ S' do
9:             L(t) := L(t) ∪ {g}
10:            H := H ∪ (∑_{t∈S'} x_(s,t) ≻ p)
11:                    extend H with constraint for s and S'
12:        end for
13:    elseif f = [f1 U^n f2]≻p
14:        choose non-deterministically S' ⊆ succ(s);
15:        for each t ∈ S' do
16:                (L(t) := L(t) ∪ {[f1 U^{n-1} f2]=pt})
17:            or (L(t) := L(t) ∪ {f2} and pt = 1)
18:                                where pt is a new variable
19:            H := H ∪ (∑_{t∈S'} x_(s,t)·pt ≻ p)
20:        end for
21:    end if
22: end for
23: for each t ∈ succ(s) do label(t) od
```

Proof. Let tree $T = (V, E, s_0)$, where V is a set of vertices, $E \subseteq V \times V$ a set of directed edges, and s_0 is the root. Every edge $e \in E$ is assigned a variable x_e denoting the weight of e. Let $\mathcal{P} = \{x_e : e \in E\}$ be the set of weights in T. To construct an MC M with T as underlying graph, we non-deterministically select a labelling function L using Alg. 2. Function L labels every vertex in T with a set of bounded PCTL-formulas. This goes as follows. We initialize $L(s_{in})$ to $\{f\}$, and invoke *label*(s_{in}) (see Alg. 2.). Line 2 covers the case when s is only labeled with propositional formulas. If s is labeled with a non-propositional formula, its labelling is adapted to the *Hintikka set* of $L(s)$ (line 3). The computation of the Hintikka set is done using Alg. 1. This procedure is non-deterministic (see lines 5 and 9). After labelling s, a non-deterministic selection of its direct successors is labeled in the for-loop (line 5–22). During this loop, a set H (initially empty) of multi-variate polynomial inequations is computed over the variables x_e and newly introduced variables p_t for vertex t (line 16–17). Each vertex of T is visited twice: once to calculate the polynomial inequations and once in the recursive call. Thus, the labelling algorithm is in NP.

Formula f holds in s_{in} iff the set of (real non-linear) inequations H, with variables in $\mathcal{P}$ is satisfiable. The number of inequations is in $O(|V| \cdot |\mathsf{sub}(f)|)$ and the number of

variables is $|E|$, i.e., polynomial in the size of the input. Using the *existential theory of the reals* [6], we can determine the feasibility of the inequations in PSPACE. This complexity can be improved by exploiting the special structure of the inequations. Observe that after some simplification (and removal of new variables introduced in lines 16–17 of Alg. 2 [4]) every equation has the following form: $a_0{\cdot}\sigma_0 + a_1{\cdot}\sigma_1 + \ldots + a_k{\cdot}\sigma_k \succ b$, where $a_0, \ldots, a_k, b \in \mathbb{Q}$ and σ_i $(0 \leq i \leq k)$ is a term of a polynomial of the type $x_{e_{1,i}} x_{e_{2,i}} \ldots x_{e_{n,i}}$ where $e_{1,i} e_{2,i} \ldots e_{n,i}$ is a path in the tree T. Furthermore, the edges $e_{1,i}$ for every $0 \leq i \leq k$ have the same source vertex. In Appendix 6.2 it is shown how to solve the satisfiability problem of such a system of (in)equations in NP. □

The complexity for the satisfiability problem for bounded PCTL is now straightforward.

Proposition 8. The satisfiability problem for bounded PCTL is in NEXPTIME in the *size* of the formula.

Proof. Let f be a bounded PCTL-formula. Theorem 1 and Proposition 7 suggest the following algorithm to solve the satisfiability problem. We non-deterministically guess a tree T of size $2^{O(\mathsf{size}(f))}$. Then we check whether there is an MC with the underlying graph T that satisfies f. The algorithm works in $\mathsf{NTIME}(2^{O(\mathsf{size}(f)^2)}) \subseteq \mathsf{NEXPTIME}$ in the size of the formula. □

Proposition 9. The satisfiability of bounded PCTL-formula f is EXPTIME-hard in the encoding of f.

Proof. By a reduction from the acceptance problem for an alternating polynomial-space Turing machine to the satisfiability of bounded PCTL-formula. Details can be found in Appendix 6.1. □

4. Simple probabilistic μ-calculus

This section considers a modal μ-calculus extended with a probabilistic next-modality. After introducing the logic, we define the notions of rank and signature. We then show that satisfiable PμTL-sentences have a model of bounded out-degree. Finally, we provide a decision procedure for PμTL satisfiability using parity games—in the same vein as for the modal μ-calculus—and yield a small model as well as a rational model property.

4.1 Syntax and semantics of PμTL

The logic PμTL, originally proposed in [17], embeds a probabilistic next-modality in the modal μ-calculus.

Definition 4 (PμTL syntax)**.** The syntax of PμTL is given by the following BNF grammar:

$$f ::= a \mid {\sim}a \mid Z \mid f \wedge f \mid f \vee f \mid [\mathsf{X}\, f]_{\succ p} \mid \mu Z.f \mid \nu Z.f.$$

Sentences are assumed to be in negative normal form, i.e., negations only occur adjacent to propositions. The μ-sentence $\mu Z.f$ is the least fixed point and the ν-sentence $\nu Z.f$ stands for the greatest fixed point. The logical variable Z is used in fixed points. The μ- and ν-sentences can intuitively be understood by considering Z as a set of states. The sentence $\mu Z.f$ is valid for all those states in the smallest set Z that satisfies the equation $Z{=}f$, where Z generally occurs in f. Similarly, $\nu Z.f$ is valid for the largest set Z satisfying $Z{=}f$. An occurrence of variable Z is *bound* in a sentence f if it occurs within a subsentence of f having either the form $\mu Z.f$ or $\nu Z.f$. In all other cases, Z occurs *free* in f. We sometimes write $\mu Z.f(Z)$ instead of $\mu Z.f$ to make the dependence of f on Z explicit. A sentence f is *closed* if all variables occurring in f are bound. In the sequel, all sentences are assumed to be closed unless stated otherwise.

Definition 5 (PμTL semantics)**.** The semantics of PμTL is defined on labelled Markov chains. The pointed satisfaction of a PμTL sentence for a Markov chain M with state space S and state $s \in S$ is defined by the following rules:

$$\begin{array}{lll}
s \models a & \text{iff} & a \in L(s) \\
s \models {\sim}a & \text{iff} & a \notin L(s) \\
s \models g \wedge h & \text{iff} & s \models g \text{ and } s \models h \\
s \models g \vee h & \text{iff} & s \models g \text{ or } s \models h \\
s \models [\mathsf{X}\, g]_{\succ p} & \text{iff} & \sum_{s' : s' \models g} P(s, s') \succ p \\
s \models \mu Z.g & \text{iff} & s \in \bigcap \{S : g(S) \subseteq S\} \\
s \models \nu Z.g & \text{iff} & s \in \bigcup \{S : S \subseteq g(S)\}
\end{array}$$

With little abuse of notation, a PμTL-sentence f also denotes set of states satisfying f. It should be clear from the context, when f is considered as a sentence and when as a set of states. In particular, $\mu Z.g$ is valid in state s whenever s is contained in the smallest set of states such that $g(S)$—the sentence g where Z is replaced by S—is a subset of S.

Example 2. The logic PμTL can be used to model probabilistic programs where probability distributions are fixed and variables have a finite domain. Consider the probabilistic program `while(c == heads) toss(c)` where c is a fair coin that initially equals `heads` or `tails`. The following PμTL sentence where proposition `t` stands for `tails` and `h` abbreviates `heads`:

$$(\mathtt{t} \rightarrow [\mathsf{X}\, \mathtt{t}]_{=1}) \wedge \left(\mathtt{h} \rightarrow \nu Z.([\mathsf{X}\, \mathtt{t}]_{\geq 1/2} \wedge [\mathsf{X}\, \mathtt{h} \wedge Z]_{\geq 1/2})\right).$$

expresses that `c` being initially `t` implies `c` stays `t` almost surely, and that `c` being initially `h` implies that it turns into `t` with at least probability $1/2$ or stays `h` and continues with the same threshold.

Sub-sentences of PμTL-sentence f are defined in the standard way, e.g., the sub-sentences of $\mu Z.f$ are $\mu Z.f$ and all sub-sentences of f, and the sub-sentences of $f \wedge g$ are f, g and $f \wedge g$. A Markov chain M is a *model* of PμTL-sentence f whenever $s_{in} \models f$.

4.2 Ranks and signatures

Let f be a function on subsets of a universe U, i.e., $f : 2^U \to 2^U$. If f is monotonic, then by the Knaster-Tarski theorem, least and greatest fixed points of f exist. For ordinal α, the least fixed point $\mu(f) = \bigcup_\alpha \mu_\alpha(f)$, and the greatest fixed point $\nu(f) = \bigcap_\alpha \nu_\alpha(f)$, where:

$$\begin{array}{l}
\mu_0(f) = \varnothing \;\text{ and }\; \mu_{\alpha+1}(f) = f(\mu_\alpha(f)) \text{ and} \\
\nu_0(f) = U \;\text{ and }\; \nu_{\alpha+1}(f) = f(\nu_\alpha(f)).
\end{array}$$

Recall that we can view a PμTL sentence f as characterising a set of states satisfying f. Hence, we denote $\eta_{\alpha+1}(f) = \{s : s \models f(\eta_\alpha(f))\}$ for $\eta \in \{\mu, \nu\}$ where $\mu_0(f) = \bot$ and $\nu_0(f) = \top$. With little abuse of notation, we denote $s \models \eta_\alpha(f)$ iff $s \in \eta_\alpha(f)$. The satisfaction relation of PμTL (see Def. 5) can now be rephrased as follows:

$$\begin{array}{l}
s \models \mu Z.f(Z) \text{ iff for some ordinal } \alpha,\ s \models \mu_\alpha(f) \\
s \models \nu Z.f(Z) \text{ iff for all ordinals } \alpha,\ s \models \nu_\alpha(f).
\end{array}$$

[4] Note that the variable p_t is the lvalue of a single equation of the form $p_t = \ldots$. Thus p_t can be easily substituted.

No state satisfies $\mu_0(f)$, and every state satisfies $\nu_0(f)$.

Definition 6 (Rank)**.** The μ-sentence $\mu Z.f(Z)$ has *rank* α at state s if α is the least ordinal such that $s \models \mu_\alpha(f)$. If there is no ordinal $\alpha < \omega$ such that $s \models \mu_\alpha(f)$, then the rank of $\mu Z.f(Z)$ at s is ω. [5]

Example 3. Consider the following Markov chain:

$$s_1 \xrightarrow{1} s_2 \xrightarrow{1} s_3 \xrightarrow{1} s_4 \xrightarrow{1} \dots$$

where s_4 satisfies a and s_i satisfies $\sim a$ for $i < 4$. The sentence $\mu Y.\,(a \vee [\mathsf{X}\, Y]_{>0})$ has rank 4 at s_1, 3 at s_2, 2 at s_3 etc.

Definition 7 (Signature)**.** A *signature* is a sequence of ordinals. Let $<$ be the lexicographical ordering on signatures. Over a set of *bounded* length signatures, the lexicographical ordering is total and well defined.

Definition 8 (μ-height)**.** The *μ-height* of $\mathsf{P}\mu\mathsf{TL}$-sentence f is the nesting depth of closed μ-sub-sentences (including f) of f.

Example 4. Formula $\mu Z.([\mathsf{X}\, Z]_{>0} \vee \mu Y.(b \wedge [\mathsf{X}\, Y]_{>0}))$ has μ-height 2. The μ-height of $\mu Z.(a \vee \mu Y.(b \wedge [\mathsf{X}\, Z]_{=1} \vee [\mathsf{X}\, Y]_{>0}))$ is 1, since $\mu Y.(b \wedge [\mathsf{X}\, Z]_{=1} \vee [\mathsf{X}\, Y]_{>0})$ is not closed.

Definition 9. Let f be a $\mathsf{P}\mu\mathsf{TL}$-sentence of μ-height n. Sentence f *has the signature* $\sigma = \alpha_1, \dots, \alpha_n$ at state s if σ is the (lexicographically) least signature such that $s \models f'$ where f' is obtained by replacing every μ-sub-sentence $\mu Z.g$ in f of μ-height i by $\mu_{\alpha_i}(g)$.

Observe that ordinal α_i is used only for least fixed point sentences of μ-height i. Greatest fixed point sentences play no role for this notion.

Example 5. Let the sentence $f = \mu Z.([\mathsf{X}\, Z]_{>0} \vee (b \wedge \mu Y.(a \vee [\mathsf{X}\, Y]_{>0})))$ with μ-height 2. Consider the MC:

$$s_1 \xrightarrow{1} s_2 \xrightarrow{1} s_3 \xrightarrow{1} s_4 \xrightarrow{1} s_5 \xrightarrow{1} \dots$$

where only s_5 satisfies a and only s_3 satisfies b. Sentence f has signature $(3, 3)$ at s_1, $(3, 2)$ at s_2, $(3, 1)$ at s_3, $(2, \omega)$ at s_4, and $(1, \omega)$ at s_5.

Proposition 10. Signatures of $\mathsf{P}\mu\mathsf{TL}$ sentences satisfy:

1. If $f \vee g$ has signature σ at s, then either f or g has signature $\leq \sigma$ at s.
2. If $f \wedge g$ has signature σ at s, then f and g both have signatures $\leq \sigma$ at s.
3. If $[\mathsf{X}\, g]_{\succ p}$ has signature σ at s, then there is a set H of successors of s, such that $\sum_{t \in H} P(s, t) \succ p$, and g has a signature $\leq \sigma$ at t, for every $t \in H$.
4. If $\mu Z.f(Z)$ has signature σ at s, then $f(\mu Z.f(Z))$ has signature $\sigma' < \sigma$ at s.
5. If $\nu Z.f(Z)$ has signature σ at s, then $f(\nu Z.f(Z))$ has signature σ' with prefix σ at s.

Proof. Cases 1 and 2. Suppose $\varphi = f \vee g$ has a signature $\sigma = (\alpha_1, \dots, \alpha_n)$ at s. Let φ' be the formula obtained by replacing each occurrence of μ-sentence $\mu Z.f(Z)$ of μ-height i by $\mu_{\sigma_i}(f)$. Then $s \models \varphi'$. Observe that each μ-sub-sentence of φ belongs either to f or g. Thus the sentence obtained by replacing every μ-sub-sentence $\mu Z.f(Z)$ of either f or g of height j by $\mu_{\sigma_j}(f)$ is also satisfied by s. Thus, either f or g has signature σ' at most σ. Similar arguments hold for $f \wedge g$.

[5] ω denotes the first infinite ordinal.

Case 3. Assume $[\mathsf{X}\, g]_{\succ p}$ has signature $\sigma = (\alpha_1, \dots, \alpha_n)$. Let g' be the sentence obtained by replacing every occurrence of μ-sentence $\mu Z.f(Z)$ of height i by $\mu_{\sigma_i}(f)$ in g. Then $s \models [\mathsf{X}\, g']_{\succ p}$. This implies that there exists a set H of successors of s such that $\sum_{t \in H} P(s, t) \succ p$ and for each $t \in H$, either $t \models g'$, or g has a signature at most σ at t.

Case 4. Let $\varphi = \mu Z.f(Z)$, and the signature at s be $\sigma = (\alpha_1, \dots, \alpha_n)$. We assume without loss of generality that φ is the only formula with μ-height n. Let ψ be a μ-sub-sentence in $f(\varphi)$. We distinguish:

1. ψ occurs either properly inside φ or does not contain Z. Then the μ-height of ψ in φ and $f(\varphi)$ coincide. Thus, the rank α_i, say, of ψ in φ is the rank of ψ in $f(\varphi)$ too.
2. $\psi = \varphi$. The μ-height of φ in $f(\varphi)$ remains n. By definition of $\mu_\alpha(f)$, the rank of sub-sentence φ (of $f(\varphi)$) is $\alpha_n - 1$.
3. ψ contains Z. Let j be the μ-height of ψ (when occurring) in φ. Then it has μ-height $n+j$ in $f(\varphi)$. The signature of $f(\varphi)$ at s is thus $\sigma' = \{\alpha_1, \dots, \alpha_{n-1}, (\alpha_n - 1), \alpha'_{n+1}, \dots \alpha'_{n+k}\}$, where $n+k$ is the μ-height of $f(\varphi)$. Lexicographically, $\sigma' < \sigma$.

Case 5. concerns greatest fixed points, which does not change the signature of any sub-sentence of $f(\nu Z.f(Z))$ whose μ-height is $\leq n$. □

4.3 Pre-model and derivations

Throughout the rest of this section, we will assume that every sub-sentence of a given sentence is unique.

Definition 10 (FL closure)**.** The *Fisher-Ladner closure* of $\mathsf{P}\mu\mathsf{TL}$-sentence f is the smallest set $\mathsf{FL}(f)$ satisfying:

1. $f \in \mathsf{FL}(f)$.
2. If $g \vee h \in \mathsf{FL}(f)$ then $g, h \in \mathsf{FL}(f)$.
3. If $g \wedge h \in \mathsf{FL}(f)$ then $g, h \in \mathsf{FL}(f)$.
4. If $[\mathsf{X} g]_{\succ p} \in \mathsf{FL}(f)$ then $g \in \mathsf{FL}(f)$.
5. If $\eta Z.g \in \mathsf{FL}(f)$ then $g(\eta Z.g) \in \mathsf{FL}(f)$ for $\eta \in \{\mu, \nu\}$.

Example 6. For $\varphi = \nu Z.(a \wedge [\mathsf{X}\, Z]_{=1})$, we have $\mathsf{FL}(\varphi) = \{\varphi, a \wedge [\mathsf{X}\,(\nu Z.a \wedge [\mathsf{X}\, Z]_{=1})]_{=1}, a, [\mathsf{X}\,(\nu Z.a \wedge [\mathsf{X}\, Z]_{=1})]_{=1}\}$.

Remark 1 ([11])**.** For every $\mathsf{P}\mu\mathsf{TL}$-formula f, $|\mathsf{FL}(f)| \in O(|f|)$.

Here $|f|$ denotes the length of sentence f. We now introduce the notion of pre-model of formula f.

Definition 11 (Pre-model)**.** A *pre-model* of $\mathsf{P}\mu\mathsf{TL}$ sentence f is an MC $M_f = (S, P, 2^{\mathsf{FL}(f)}, L, s_{in})$ satisfying:

1. $f \in L(s_{in})$.
2. If $f \in L(s)$ then $\sim f \notin L(s)$.
3. If $f \vee g \in L(s)$ then $f \in L(s)$ or $g \in L(s)$.
4. If $f \wedge g \in L(s)$ then $f, g \in L(s)$.
5. If $[\mathsf{X}\, g]_{\succ p} \in L(s)$ then $\sum_{s' : g \in L(s')} P(s, s') \succ p$.
6. If $\eta Z.f \in L(s)$ then $f(\eta Z.f) \in L(s)$ with $\eta \in \{\mu, \nu\}$.

Each pre-model defines a specific choice of *derivation* rules.

Definition 12 (Derivation)**.** The *derivation relation induced* by pre-model $M_f = (S, P, 2^{\mathsf{FL}(f)}, L, s_{in})$ of $\mathsf{P}\mu\mathsf{TL}$ sentence f is defined by:

1. If $\varphi = h \vee g \in L(s)$ and $h \in L(s)$, then φ *derives* h (at s). Similar holds if $g \in L(s)$.
2. If $\varphi = h \wedge g \in L(s)$, then φ *derives* h and g (at s).

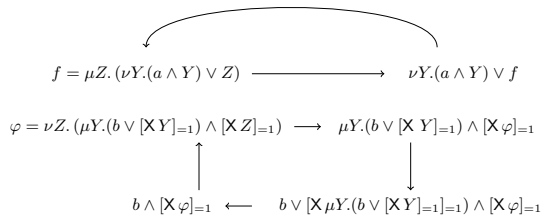

Figure 1. Derivation sequence for two formulas. The derivation sequence in the upper part shows a regeneration sequence; the derivation in the lower part is not a regeneration sequence.

3. If $\varphi = [\mathsf{X}\,g]_{\succ p} \in L(s)$ and $g \in L(t)$ for some successor t of s, then φ (at s) *derives* g (at t).
4. If $\varphi = \eta Z.h \in L(s)$ with $\eta \in \{\mu, \nu\}$, then φ *derives* $h(\eta Z.h)$ (at s).

The *derivation relation* of f is the union of derivation relations induced by all pre-models of f, i.e., g *derives* h iff for some states s, t of a pre-model of f, g in s derives h in t.

Note that the derivation relation of f only relates sentences in $\mathsf{FL}(f)$. An intuitive way to understand the derivation relation is to consider it as a logical implication. Not all pre-models of f are models of f. For instance, sentence $\varphi = \mu Z.f(Z)$ could be derived forever (by clause 4 of Def. 12). For a pre-model to be a model, a μ-sentence cannot be derived over and over again without ever satisfying it. But it is possible that a μ-sub-sentence appears infinitely often in a derivation sequence. So we have to be careful about which derivation sequences we are referring to. We will use the following definition.

Definition 13 (Regeneration)**.** The μ-sentence φ is *regenerated* by a sequence of derivations, if starting from φ we end up again with φ, and φ is a μ-sub-sentence for every sentence in any intermediate derivation.

Example 7. Sentence $f = \mu Z.\,(\nu Y.(a \wedge Y) \vee Z)$ derives $\nu Y.(a \wedge Y) \vee \mu Z.\,(\nu Y.(a \wedge Y) \vee Z)$ that contains f, and which can derive $\mu Z.\,(\nu Y.(a \wedge Y) \vee Z)$ which equals f. Thus the μ-sentence f is regenerated. (See Fig. 1, upper part).

Example 8. Let $\varphi = \nu Z.\,(\mu Y.(b \vee [\mathsf{X}\,Y]_{=1}) \wedge [\mathsf{X}\,Z]_{=1})$. φ derives $\mu Y.(b \vee [\mathsf{X}\,Y]_{=1}) \wedge [\mathsf{X}\,\varphi]_{=1}$. This derives a formula containing $\mu Y.(b \vee [\mathsf{X}\,Y]_{=1}) \wedge [\mathsf{X}\,\varphi]_{=1}$ (see Fig. 1, lower part). This in turn derives $b \wedge [\mathsf{X}\,\varphi]_{=1}$ which derives φ. Although the μ-sentence $\mu Y.(b \vee [\mathsf{X}\,Y]_{=1})$ is witnessed again, it is not regenerated, since it is not a sub-sentence of every derived sentence.

Definition 14 (Well-foundedness)**.** A pre-model M_f of $\mathsf{P}\mu\mathsf{TL}$-sentence f is *well-founded* if every μ-sub-sentence φ of f is regenerated *finitely* often.

Theorem 2. Every model of $\mathsf{P}\mu\mathsf{TL}$-sentence f is a well-founded pre-model of f.

Proof. Let M be a model of f. We first generalize the state-labeling; $\forall f \in \mathsf{FL}(\varphi) :\ s \models f \iff f \in L(s)$. Note that with the generalized labeling, the model satisfies the conditions in Def. 11. The rest of the reasoning is as follows: If $M, s_{in} \models \varphi$, then φ has a signature at initial state s_{in}, and we can ensure that every μ-sub-sentence of φ is regenerated finitely often.

Let $\psi = \mu Z.f(Z)$ be a sub-sentence of φ with μ-height n that is regenerated following a sequence of derivations from s to t (s and t can be identical). We will show that the signature $\sigma = (\alpha_1, \ldots, \alpha_n)$ of ψ from s to t decreases. As per definition, the derivation step begins by deriving $f(\mu Z.f(Z))$ from $\mu Z.f(Z)$. By Proposition 10, the sentence $f(\mu Z.f(Z))$ has lexicographically smaller signature at s. It remains to show that this decrease is not violated by other derivation rules between s and t.

For disjunction $h \vee g$, suppose (w.l.o.g.) $s \models g$. By Def. 12, g is derived. By Proposition 10, the signature cannot increase. As the derivation relation for a conjunction does not affect the signature, for this case no decrease occurs. The derivation rule for $[\mathsf{X}g]_{\succ p}$ derives g at some successor state t. If $f(\psi)$ is a sub-sentence, then the signature at t cannot increase; the other case is trivial.

A derivation may involve other fixed point sentences. Derivations of fixed point sentences that are sub-sentences of ψ do not affect the μ-height of $f(\varphi)$. For example, consider $\psi = \mu Z.f(Z, \mu Y.g(Y))$ with μ-height n. Applying the derivation for φ yields $\psi' = f(\varphi, \mu Y.g(Y))$ which has μ-height n too. The derivation steps for $\mu Y.g(Y)$ in $f(\varphi, \mu Y.g(Y))$ give $f(\varphi, g(\mu Y.g(Y)))$. The μ-height of $\mu Z.f(Z, g(\mu Y.g(Y)))$ does not change, hence the ordinal α_n in the signature of $f(\varphi)$ is unaffected.

Now, consider $\phi = \eta Y.g(Y)$ (where $\eta \in \{\mu, \nu\}$) with μ-subsentence $\psi = \mu Z.f(Z)$. Distinguish two cases:

1. The derivation of ϕ does not affect the μ-height of ψ. In such a case, deriving $\mu Y.g(Y)$ decreases the signature (by Proposition 10). For example, consider the derivation steps for $\varphi = \mu Y.g(Y, \mu Z.f(Z))$. This gives $g(\mu Y.g(Y, \mu Z.f(Z)), \mu Z.f(Z))$. The μ-height of $\mu Z.f(Z)$ is not affected, and the signature of φ decreases.
2. The derivation of ϕ increases the μ-height of ψ. For example, $\phi = \eta Y.g(\psi)$, where $\psi = \mu Z.f(Z, Y)$. Observe that a derivation of ψ can only occur after a derivation of ϕ. That would make ϕ a subsentence of ψ, namely $\mu Z.f(Z, \phi)$. The case where ϕ is a subsentence of ψ has already been considered.

Thus, we have derivations where each μ-sentence reduces its corresponding rank. Since the derivation sequence from φ has bounded length (by Observation 1, pp. 6), a regeneration can only happen finitely often. Thus the pre-model is well-founded. □

Theorem 3. Each well-founded pre-model of $\mathsf{P}\mu\mathsf{TL}$-sentence f is a model of f.

Proof. (*sketch*) Let MC M be a well-founded pre-model of f. Then the regeneration relation for every μ-subsentence of f terminates. Each μ-sentence thus has a (finite) rank at every state in M, and hence there exists a signature for $\varphi \in \mathsf{FL}(f)$ at each state. Let $\sigma = \alpha_1, \ldots, \alpha_n$ be the lexicographically smallest such signature. Replace each occurrence of the μ-sentence $\mu Z.f(Z)$ of height i by $\mu_{\alpha_i}(f)$. It follows by structural induction on sentences that $\varphi \in L(s)$ implies $s \models \varphi$. □

Theorem 4. If $\mathsf{P}\mu\mathsf{TL}$-sentence f is satisfiable, then it has a model of bounded out-degree at most $|f|+1$.

Proof. Similar to the proof of Proposition 4 (pp. 3). □

4.4 Decision procedure for PμTL satisfiability

This section presents a decision procedure for determining the satisfiability of PμTL sentence f. The procedure is based on a parity game obtained as cross-product of a game graph and a deterministic parity automaton.

Deterministic parity automaton

We first focus on the parity automaton. The starting point is a Büchi-automaton A_φ for each μ-sentence φ of f. The automaton A_φ accepts the regeneration sequences for φ, i.e., derivation sequences that derive φ infinitely often and for which φ is a sub-sentence of every sentence in the derivation.

Definition 15 (Büchi automaton for an μ-sentence)**.** Let f be a PμTL-sentence with φ a μ-sentence in $\mathsf{FL}(f)$. The *nondeterministic Büchi automaton (NBA)* A_φ is a quintuple $(Q_\varphi, \Sigma_\varphi, Q_{\varphi,in}, \delta_\varphi, F_\varphi)$ where:

1. $Q_\varphi = \{\psi \in \mathsf{FL}(f) : \varphi \text{ is a } \mu\text{-sentence of } \psi\}$, the state-set
2. $\Sigma_\varphi = 2^{\mathsf{FL}(f)}$, the alphabet
3. $Q_{\varphi,in} = Q_\varphi$, the set of initial states,
4. $\delta_\varphi(q, q') = q'$, if q' is obtained from q by a derivation
5. $F_\varphi = Q_\varphi$, the set of final states.

The transition relation is represented in a compressed form, that is, $\delta_\varphi(q, a) = q'$ implies that for all $A \in \Sigma$ with $a \in A$, $\delta_\varphi(q, A) = q'$.

For PμTL-sentence f, let A_f be a deterministic parity automaton (DPA) which is the complement of the union of the automata A_φ for μ-sentence φ of f:

$$\overline{L(A_f)} = \bigcup_{\varphi \in \mathsf{FL}(f) : \varphi \text{ is a } \mu\text{-sentence}} L(A_\varphi)$$

A_f thus accepts all terminating regeneration sequences for every μ-sentence in $\mathsf{FL}(f)$. The union of $L(A_\varphi)$ can be described by an NBA of maximal size in $O(kn)$ where k is the number of μ-sentences in $\mathsf{FL}(f)$ and $|\mathsf{FL}(f)| = n$. The DPA A_f then has size at most $O(2^{(kn)^2})$ [19].

A two-player game

Our next aim is to define a two-player game where player 0 aims to show that PμTL-sentence f is satisfiable, while its opponent wants to refute this claim. The vertices of the game graph are sets of subsets of $\mathsf{FL}(f)$. A vertex v is called *transitive* iff:

- For all $g \vee h \in v$ either $g \in v$ or $h \in v$.
- For all $g \wedge h \in v$, $g, h \in v$.
- For all $\eta X.g(X) \in v$, $g(\eta X.g(X)) \in v$.
- There exists $[\mathsf{X} g]_{\succ p} \in v$.

Definition 16 (Two-player game)**.** The two-player game G_f for PμTL-sentence f is the triple (V, E, v_0) where $V = V_0 \uplus V_1$ with $V_0 \subseteq 2^{\mathsf{FL}(f)}$ the set of Player 0 vertices, V_1 the set of Player 1 vertices (defined below), $v_0 = \{f\} \in V_0$ the starting vertex, and E is defined by:

1. $(v, v \cup \{g_i\}) \in E$ for i=1, 2, if $g = g_1 \vee g_2 \in v$
2. $(v, v \cup \{g_1, g_2\}) \in E)$ if $g = g_1 \wedge g_2 \in v$
3. $(v, v \cup \{g(\mu X.g)\}) \in E$ if $\mu X.g \in v$, and
4. $(v, v_c) \in E$ if v is a transitive vertex and v_c represents weighted cover (c, w) of $V_v = \{g : [\mathsf{X} g]_{\succ p} \in v\}$ with $w(g) \succ p$ for all $g \in V_v$. In addition, $(v_c, v') \in E$ for each $v' \in c$. If no such cover c exists, then $(v, \bot) \in E$. Let $V_1 = \{v_c : (v, v_c) \in E\}$ with v_c as above.

Example 9. Consider the game in Figure 2. Let us clarify one of its weighted covers. Consider the vertex $\langle \neg \mathtt{t},\ [\mathsf{X}\,\mathtt{t}]_{\geqslant \frac{1}{2}} \wedge [\mathsf{X}\,\mathtt{h} \wedge \nu Z.\varphi(Z)]_{\geqslant \frac{1}{2}} \rangle$. There are two possible weighted covers: (c_1, w_1) and (c_2, w_2). Let $c_1 = \{v_1, v_2\}$ and $c_2 = \{v_3\}$, where $v_1 = \langle \mathtt{h} \wedge \nu Z.\varphi(Z) \rangle$, $v_2 = \langle \mathtt{t} \rangle$ and $v_3 = \langle \mathtt{h} \wedge \nu Z.\varphi(Z), \mathtt{t} \rangle$. The weights are $w_1(v_1) = w_1(v_2) = 1/2$ and $w_2(v_3) = 1$.

Let $V_\bot$ be the set of vertices which contain propositional contradictions (like a and $\sim a$). The game graph G_f is of size at most $2^{O(n^2)}$ where $n = |\mathsf{FL}(f)|$. Player 0 looses if the finite play reaches $V_\bot$. An infinite play is winning for player 0 if it is accepted by the DPA A_f. To accomplish this, we define the parity game $\mathcal{G}_f$ as the (synchronous) cross product of game G_f and DPA A_f; for details we refer to Appendix 6.3.

Remark 2. There are some crucial differences between the game $\mathcal{G}_f$ and the tree-automaton construction for PμTL in [17]. For vertices with formula $g \wedge h$, the set of formulas is not split into two vertices (one containing g and one containing h); instead they are kept together. The key difference is the distribution of formulas as solutions of weighted covers.

Proposition 11. Player 0 has a winning strategy in parity game $\mathcal{G}_f$ for every satisfiable PμTL-sentence f.

Proof. Let f be satisfiable. By Theorem 2, f has a well-defined pre-model, say MC $M_f = (S, P, \mathsf{AP}, L, s_{in})$. The proof is by constructing a winning strategy $\pi : V^+ \to V$ in game $\mathcal{G}_f$ for player 0 against any strategy of player 1. This is done using the auxiliary function $\Gamma : V^+ \to S$ that maps finite plays in $\mathcal{G}_f$ onto states of M_f. Define $\Gamma(v_0) = s_{in}$. Consider a finite play ρ of $\mathcal{G}_f$ with $s = \Gamma(\rho)$ and $v = \mathsf{last}(\rho)$. Distinguish the following cases:

- v is a vertex with $g_1 \vee g_2 \in v$. If $g_i \in L(s)$ then let $\pi(\rho) = v_i$ (see Def. 16) and $\Gamma(\rho \cdot v_i) = s$, for $i \in \{1, 2\}$.
- v is a transitive vertex. Assume s has direct successors $\{t_1, \ldots, t_k\}$. By Theorem 4, it follows $k \leq |\mathsf{FL}(f)|+1$. Define $\pi(\rho) = v_c$ (i.e., a cover vertex, see Def. 16) for cover $c = \{L(t_1), \ldots, L(t_k)\}$. (Note that such cover always exists.) If player 1 selects $L(t_i)$, then let $\Gamma(\rho \cdot v_c \cdot L(t_i)) = t_i$.
- in any other case, v has at the most one successor, say v'. Define $\pi(\rho) = v'$ and $\Gamma(\rho \cdot v') = s$.

It remains to show that π is a winning strategy. For any strategy σ of Player 1, consider the resulting path ρ from the pair of strategies (π, σ). Path ρ cannot terminate in a $\bot$-vertex, as otherwise the label of $\Gamma(\rho)$ should contain a propositional contradiction. If ρ is infinite, then every regenerating μ-sub-sentence in the vertices of ρ is terminating. Hence $\rho \in L(A_f)$. □

Proposition 12. If there exists a winning strategy for player 0 in parity game $\mathcal{G}_f$, then f is satisfiable.

Proof. Let π be a winning strategy of player 0 in $\mathcal{G}_f$. Applying the strategy π to the game $\mathcal{G}_f$ yields the digraph $\mathcal{G}_f^\pi$. Let Π be the set of all finite paths $\sigma = (\sigma_0 \ldots \sigma_n)$ in $\mathcal{G}_f^\pi$ such that

1. σ_0 is a player 0 configuration, and it is either the initial configuration or has a player 1 configuration as a parent.
2. σ_n is a player 1 configuration.

Consider a path $\sigma \in \Pi$. Observe that each configuration except the last configuration (σ_n) of σ has at most one descendant. Path σ is said to *lead* to σ', if the last configuration

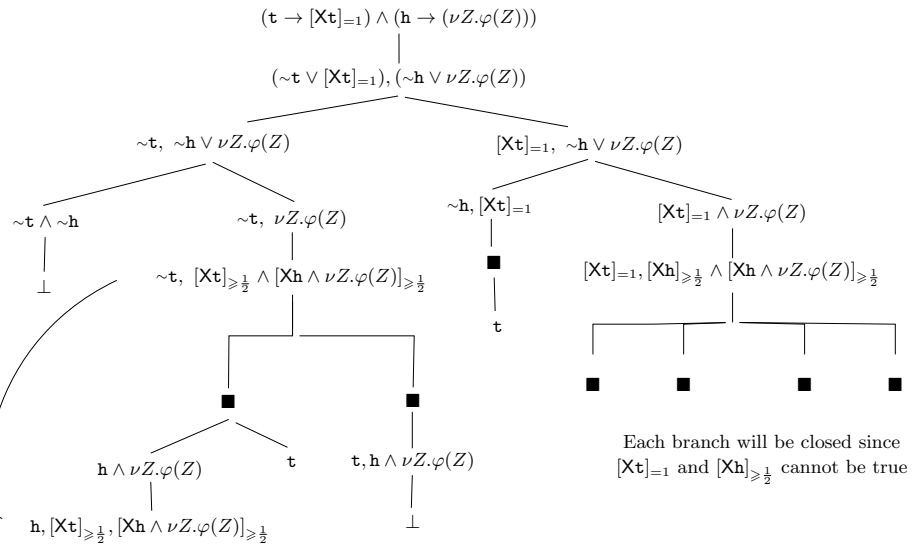

Figure 2. The satisfiability game for $(\mathtt{t} \to [\mathsf{X}\mathtt{t}]_{=1}) \wedge (\mathtt{h} \to (\nu Z.\varphi(Z)))$, where $\varphi = [\mathsf{X}\mathtt{t}]_{\geqslant\frac{1}{2}} \wedge [\mathsf{X}\mathtt{h} \wedge Z]_{\geqslant\frac{1}{2}}$. The black squares indicate Player 1 vertices, others are Player 0 vertices. Since there are no μ sub-sentences, any infinite path (or ending in no $\bot$-vertex) is winning for Player 0.

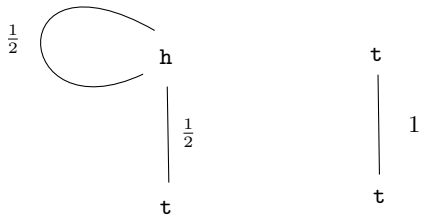

Figure 3. Two models generated by two different winning strategies of Player 0 in the game given in Fig. 2

of σ has an edge to every configuration of σ'. Let pre-model $M_f = (S, P, \mathsf{AP}, L, s_0)$ be obtained from digraph $\mathcal{G}_f^\pi$ in the following way:

- $S = \{s_\sigma : \sigma \in \Pi\}$.
- $P(s_\sigma, s_{\sigma'}) > 0$, if σ *leads* to σ'. The exact value is the weight defined by the weighted cover σ_n (see also Example 9).
- $L(s_\sigma) = \bigcup_{i=0}^{k-1} \sigma_i$, where $k = |\sigma|$.
- $s_0 = s_\sigma$, where $\sigma = (\sigma_0, \ldots, \sigma_n)$ such that σ_0 is the initial configuration of the game.

It is easy to see that the pre-model M is well-founded. From Proposition 3 it follows that f is satisfiable. □

Theorem 5. Every satisfiable $\mathsf{P}\mu\mathsf{TL}$ sentence f has a model of size exponential in $|f|$.

Proof. By Propositions 11 and 12 it follows that $\mathsf{P}\mu\mathsf{TL}$-sentence f is satisfiable iff Player 0 has a winning strategy in $\mathcal{G}_f$. It is well-known that if a winning strategy for a parity game exists, then there exists a pure memoryless winning strategy [12]. The size of the well-founded pre-model defined by a pure memoryless strategy is $2^{O((kn)^2)}$. □

Remark 3. Our last result together with the fact that qualitative PCTL has no finite model property [5] yields that qualitative PCTL and $\mathsf{P}\mu\mathsf{TL}$ have incomparable expressive power. The $\mathsf{P}\mu\mathsf{TL}$-formula $\nu Y.(a \wedge [\mathsf{X}\, Y]_{>0})$ cannot be expressed in qualitative PCTL. Vice versa, the PCTL-formula $[\mathsf{G}\,([\mathsf{X}\, a]_{>0} \wedge \sim a)]_{>0}$ cannot be expressed in $\mathsf{P}\mu\mathsf{TL}$.

Corollary 1. Every satisfiable $\mathsf{P}\mu\mathsf{TL}$ sentence has a model whose transition probabilities are rational.

Proof. The weight function for the weighted cover is determined by linear constraints. Thus if it is satisfiable, then it has a rational solution (recall that the probability bounds on the formula are rational). □

4.5 Discussion

Compared to the alternating tree automaton approach in [7], our game-based approach has the following advantages:

- It provides a clear separation of recursive sub-sentences and sentences with probability bounds. The probability bounds in the decision procedure only affect the existence of the *cover* vertices (v, v_c) in the game.
- An (ordinary) non-stochastic parity game was used for the satisfiability of a probabilistic logic.
- The game graph of a $\mathsf{P}\mu\mathsf{TL}$-sentence f captures all models of f.

The latter property enables querying some quantitative properties, e.g., the maximum probability of reaching certain states in the models of f. For instance, consider the game graph $G_f = (V, E, v_0)$ in Fig. 2, where all vertices that are losing for player 0 are omitted. The maximum probability of reaching specific vertices can then be obtained as follows. The cover vertices (player 1 vertices) act as probabilistic vertices whose transition relations are defined by linear equations. Using algorithms for determining reacha-

bility probabilities in convex MDPs [8, 21][6], we can calculate the maximum probability of reaching certain vertices.

This observation is a step towards considering an extension of $\mathsf{P}\mu\mathsf{TL}$ with until-modalities (that do neither occur as sub-sentence of another until-modality nor as part of a μ-sentence). Our decision procedure can be extended as follows for $[f_1 \,\mathsf{U}\, f_2]_{\succ p}$. We first extend the transition relation in Def. 16 by:

5.1 $(v, v') \in E$ if $[f_1 \,\mathsf{U}\, f_2]_{\succ p} \in v$ and $f_2 \vee (f_1 \wedge [\mathsf{X}\,\mathsf{U}^*(f_1, f_2)]_{>0}) \in v'$, where $\mathsf{U}^*(f_1, f_2)$ denotes $[f_1 \,\mathsf{U}\, f_2]_{\succ p'}$ for some p' (which is not relevant here).

5.2 $(v, v') \in E$ if $\mathsf{U}^*(f_1, f_2) \in v$ and $f_2 \vee (f_1 \wedge [\mathsf{X}\,\mathsf{U}^*(f_1, f_2)]_{>0}) \in v'$.

Intuitively, $[f_1 \,\mathsf{U}\, f_2]_{\succ p}$ is dealt as $\mu Z.\,(f_2 \vee (f_1 \wedge [\mathsf{X}\, Z]_{>0}))$. This yields the game graph $G_f = (V, E, v_0)$ and the winning condition is obtained as in Section 4.4. We remove all vertices that are not winning for player 0 and the sentence is satisfiable if for each (remaining) vertex w with $[f_1 \,\mathsf{U}\, f_2]_{\succ p} \in w$, the supremum probability of reaching vertices with f_2 only via vertices containing f_1 is $\succ p$.

This raises the question whether this technique can be extended to nested until-modalities. By a similar mechanism as above we annotate vertices with sentences $[f_1 \,\mathsf{U}\, f_2]_{\succ p}$ with *obligation* $\succ p$. This then amounts to decide the reachability problem for MDPs with *obligations* [10].[7] To our knowledge, such obligatory games can only be solved under strong structural restrictions. Nonetheless we believe this indicates that tying the satisfiability problem of a recursive probabilistic logic (with unbounded until) to a *finite* obligatory game is a promising avenue.

5. Conclusion

This paper considered the satisfiability problem of bounded PCTL and $\mathsf{P}\mu\mathsf{TL}$. Both logics possess the finite model property, whereas $\mathsf{P}\mu\mathsf{TL}$ is shown to also have the rational model property. Our results for $\mathsf{P}\mu\mathsf{TL}$ show that $\mathsf{P}\mu\mathsf{TL}$ and qualitative PCTL have incomparable expressive power. We have shown that the satisfiability problem for bounded PCTL formula f is in NEXPTIME in the size of f, and EXPTIME-hard in its encoding. The satisfiability problem for $\mathsf{P}\mu\mathsf{TL}$ is shown to be in the same complexity class as the satisfiability problem for the modal μ-calculus, i.e., in $\mathsf{UTIME}(2^{O(|f|)}) \cap \text{co-}\mathsf{UTIME}(2^{O(|f|)})$. This improves the 2-EXPTIME algorithm recently provided in [17]. Table 1 summarises the current situation. The satisfiability of PCTL [13] and μ-PCTL [7] remain open problems.

Acknowledgement. This work is supported by the Excellence Initiative of the German federal and state government and the EU FP7-project Sensation.

Logic	Finite model	Small model	Satisfiability checking
Px_ω	yes	$O(\|f\|)$	PSPACE-c.
bounded PCTL	yes	$2^{O(size(f))}$	NEXPTIME EXPTIME-hard
qualitative PCTL	no	–	EXPTIME-c.
PCTL	no	–	?
$\mathsf{P}\mu\mathsf{TL}$	yes	$2^{O(\|f\|)}$	$\mathsf{UTIME}(2^{O(\|f\|)}) \cap$ co-$\mathsf{UTIME}(2^{O(\|f\|)})$
μPCTL	no	–	?

Table 1. Overview of known satisfiability results (where $size(f)$ equals $|\mathsf{ord}(f)|+|\mathsf{sub}(f)|$). The first two rows and the fifth row summarise this paper.

References

[1] Nathalie Bertrand, John Fearnley, and Sven Schewe. Bounded satisfiability for PCTL. In *CSL*, volume 16 of *LIPIcs*, pages 92–106. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[2] Andrea Bianco and Luca de Alfaro. Model checking of probabalistic and nondeterministic systems. In *FSTTCS*, volume 1026 of *LNCS*, pages 499–513. Springer, 1995.

[3] Patrick Billingsley. *Probability and Measure*. Wiley & Sons, 1995.

[4] Garrett Birkhoff and Saunders Mac Lane. *A Survey of Modern Algebra*. AKP classics. A.K. Peters, 1997.

[5] Tomás Brázdil, Vojtech Forejt, Jan Kretínský, and Antonín Kucera. The satisfiability problem for probabilistic CTL. In *LICS*, pages 391–402. IEEE CS, 2008.

[6] John Canny. Some algebraic and geometric computations in PSPACE. In *STOC*, pages 460–467. ACM, 1988.

[7] Pablo F. Castro, Cecilia Kilmurray, and Nir Piterman. Tractable probabilistic mu-calculus that expresses probabilistic temporal logics. In *STACS*, volume 30 of *LIPIcs*, pages 211–223. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[8] Souymodip Chakraborty and Joost-Pieter Katoen. Model checking of open interval Markov chains. In *ASMTA*, volume 9081 of *LNCS*, pages 30–42. Springer, 2015.

[9] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.

[10] Krishnendu Chatterjee and Nir Piterman. Obligation Blackwell games and p-automata. *CoRR*, abs/1206.5174, 2012.

[11] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. of Comput. and Syst. Sci.*, 18(2):194 – 211, 1979.

[12] Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *STOC*, pages 60–65. ACM, 1982.

[13] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[14] Sergiu Hart and Micha Sharir. Probabilistic propositional temporal logics. *Inf. and Control*, 70(2/3):97–155, 1986.

[15] Dexter C. Kozen. *Theory of Computation*. Springer, 2006.

[16] Richard E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comput.*, 6(3):467–480, 1977.

[17] Wanwei Liu, Lei Song, Ji Wang, and Lijun Zhang. A simple probabilistic extension of modal mu-calculus. In *IJCAI*, pages 882–888. AAAI Press, 2015.

[18] Matteo Mio. Probabilistic modal μ-calculus with independent product. *Logical Methods in Computer Science*, 8(4), 2012.

[19] Nir Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3), 2007.

[20] Sally Popkorn. *First Steps in Modal Logic*. Cambridge University Press, 1994.

[21] Alberto Puggelli, Wenchao Li, Alberto Sangiovanni-Vincentelli, and Sanjit Seshia. Polynomial-time verification of PCTL properties of MDPs with convex uncertainties. In *CAV*, volume 8044 of *LNCS*, pages 527–542. Springer, 2013.

[6] Even in the presence of non-strict in-equalities [8].

[7] Formal definition of obligatory games is beyond the scope of this paper.

6. Appendix

6.1 Hardness proofs for bounded PCTL

Proposition 6. The satisfiability problem for Px_ω is PSPACE-hard.

Proof. The proof is by a reduction from the quantified Boolean formula (QBF) problem which is PSPACE-hard. The main idea behind the reduction is to give a logspace transducer to convert every instance of a QBF f to an equivalent sentence g in Px_ω. The proof goes along the lines of proving the PSPACE-hardness of the modal logic K [20], as provided in [16]. This follows from the similarity of the Px_ω-sentence $[\mathsf{X}\, g]_{=1}$ and the $\Box$-modality in the logic K [8]. Let $\Box g$ denote $[\mathsf{X}\, g]_{=1}$ and $\Diamond g$ denote $\sim[\mathsf{X} \sim g]_{=1}$, i.e., $\Diamond g$ is equivalent to $[\mathsf{X}\, g]_{>0}$. Let QBF f of the form $Q_1 x_1 . \cdots Q_m x_m . \varphi(x_1, \ldots, x_m)$ with quantifier $Q_i \in \{\exists, \forall\}$, Boolean variable x_i and quantifier-free Boolean formula $\varphi(x_1, \ldots, x_m)$ with variables $x_1, \ldots, x_m$.

We use the new propositions $y_0, \ldots, y_m$ to uniquely encode the index $0 \le i \le m$. To that end, let $z_1, \ldots, z_n$, where $n = \lceil \log m \rceil$ be new propositions such that

$$y_i \;\equiv\; \beta_{i,1} z_1 \wedge \ldots \wedge \beta_{i,n} z_n \text{ for } 0 \le i \le m,$$

where $\beta_{i,j} = \sim$ if the j^{th} bit of (binary) i is zero, else $\beta_{i,j}$ is the empty string ($1 \le j \le n$). Let g' denote the conjunction of all such equivalences. We now define the Px_ω-sentence g as the conjunction of the following sentences:

$$\begin{array}{ll}
\text{(F1)}: & \Box^m g' \\
\text{(F2)}: & y_0 \\
\text{(F3)}: & \Box^m (y_i \to \Diamond y_{i+1}) \;\text{ for } 0 \le i < m \\
\text{(F4)}: & \Box^m (y_i \to ((x_i \to \Box^{m-i} x_{i+1}) \\
 & \qquad \wedge (\sim x_i \to \Box^{m-i} \sim x_{i+1}))) \text{ for } 0 < i < m \\
\text{(F5)}: & \Box^m (y_i \to (\Diamond(y_{i+1} \wedge x_{i+1}) \wedge (\Diamond(y_{i+1} \wedge \sim x_{i+1})))) \\
 & \quad \text{if } Q_i = \forall, \text{ for } 0 \le i < m \\
\text{(F6)}: & \Box^m (y_m \to \varphi),
\end{array}$$

where $\Box^m h = h \wedge \Box(\Box^{m-1} h)$ for $m > 0$ and $\Box^0 h = h$. Intuitively, $\Box^m h$ holds at s if h is true at every state reachable from s within m steps. The idea behind the reduction is that any model of Px_ω-sentence g *simulates* the QBF f. This can be seen as follows. Assume g is satisfiable. By Theorem 1, g is satisfiable by a tree MC with root s_{in}, say. The variable y_i marks the states of the tree MC at depth i, as ensured by (F1) through (F3). If the i^{th} quantifier Q_i is universal, then (F5) guarantees that there are two descendants, one of which makes x_i true and the other makes $\sim x_i$ true. Once x_i (or $\sim x_i$) is chosen at a branch, (F4) guarantees that it remains unaltered for every descendant. Finally, by (F6) the quantifier free Boolean sentence $\varphi(x_1, \ldots, x_m)$ is evaluated at depth m.

To see that logspace suffices to obtain the Px_ω-sentence g from the QBF f, observe that at each step we need to be able to count the index i ($0 \le i \le m$). This can be stored in logspace of the working tape, while the corresponding string (the sentence as defined by (F1)–(F6)) can be written on the output tape. □

Proposition 9. The satisfiability of bounded PCTL-formula f is EXPTIME-hard in the encoding of the formula f.

Proof. We will show EXPTIME-hardness by encoding the runs of an alternating Turing machine in bounded PCTL. A similar proof technique was used in [11] to show EXPTIME hardness for PDL (Propositional Dynamic Logic). An *alternating Turing machine* (ATM) is a non-deterministic Turing machine (NTM) equipped with a function type indicating whether a state is an *and-state* or an *or-state*. An ATM with only or-states behaves exactly like an NTM. Formally, an ATM $A = (Q, \Sigma, \Gamma, \delta, q_0, \mathsf{type}, F)$ where Q is a finite set of states, Σ is a finite set of input symbols, Γ is a finite set of tape symbols (with $\Sigma \subseteq \Gamma$), $\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L, R\}$ is a transition relation, $q_0 \in Q$ is the initial state, $\mathsf{type} : Q \to \{\wedge, \vee\}$, and $F \subseteq Q$ is the set of accepting states.

Configurations of an ATM are elements in $\Gamma^* \times Q \times \Gamma^+$; configuration $\sigma = xqay$ denotes that the current tape content is $xay = \mathsf{tape}(\sigma) \in \Gamma^+$ with $a \in \Gamma$, the head is at position $|x|+1$, presently reading input a and the current state is $q = \mathsf{state}(\sigma)$. A configuration σ is an and-configuration (or-configuration) iff type of $\mathsf{state}(\sigma)$ is $\wedge$ ($\vee$, respectively). Configuration σ is *accepting* iff $\mathsf{state}(\sigma) \in F$. The successor configuration $\sigma' = x'q'a'y'$ of $\sigma = xqay$ is defined as follows:

- If $(q, a, q', b, L) \in \delta$ then $x'a' = x$ and $y' = by$.
- If $(q, a, q', b, R) \in \delta$ then $x' = xb$ and $y = a'y'$.

A *run* C of ATM A for input $x_{in} \in \Sigma^*$ is a tree of configurations rooted at $q_0 x_{in} \in C$. For every $\sigma \in C$ with $\mathsf{state}(\sigma) \notin F$ it holds: if $\mathsf{type}(\mathsf{state}(\sigma)) = \vee$, then one of the successor configurations σ' of σ is in C, if $\mathsf{type}(\mathsf{state}(\sigma)) = \wedge$ then every successor configuration of σ is in C. Pictorially, C is a tree where each node is a configuration and edges are defined by the *next* relation. A run C for input x is accepting if it is finite and the state of each leaf configuration of C is accepting. Let:

$$L(A) = \{x \in \Sigma^* : \text{ there is an accepting run } C \text{ for } x \}$$

For function $S : \mathbb{N} \to \mathbb{N}$, an ATM A is in $\mathsf{ASPACE}(S(n))$ iff every configuration of all runs for x requires at most $S(n)$ space, for every input $x \in \Sigma^*$ with $n = |x|$. (Here it is assumed that a configuration occurs at most once in run C of x. This can be achieved by enumerating every reachable configuration while encoding the numbering in $S(|x|)$ cells of the tape.) Thus, the number of steps of an accepting (or rejecting) run is at most $|\Gamma|^{2S(n)}$ or in $2^{O(S(n))}$. We will need the following identity [9]:

$$\mathsf{ASPACE}(S(n)) = \bigcup_k \mathsf{DTIME}(2^{kS(n)}). \tag{2}$$

Now consider an input x of length n of ATM $A \in \mathsf{ASPACE}(S(n))$, where $m = S(n)+2$ and the maximum number of steps needed by the ATM to accept (or reject) is $k = 2^m$. Observe that k can be encoded in m bits.

We will construct a bounded PCTL-formula f from ATM A and input x such that every model of f encodes a computation of A with input x iff $x \in L(A)$. Each node of the model will encode a configuration of the computation and the successor relation is simulated by $\Box g$ or $\Diamond g$, i.e., $[\mathsf{X}\, g]_{=1}$ or $[\mathsf{X}\, g]_{>0}$, respectively. We use the following set of propositions AP:

1. Cells: for each $a \in \Gamma$ and $0 \le i \le m$, $C_{a,i} \in \mathsf{AP}$.
2. States: for each $q \in Q$, $Q_q \in \mathsf{AP}$.
3. Head: for each $0 \le i \le n$, $H_i \in \mathsf{AP}$.

Intuitively, $C_{a,i}$ denotes that the i^{th} cell of the tape contains symbol a, Q_q denotes that the current state of A is q and

[8] The more appropriate modal logic system would be with K and serial axioms.

H_i denotes that the head is on the i^{th} cell. We will use the following formulas to capture the behaviour of ATM A:

- At every node of the model, A is at exactly one state:

$$g_1 \;:=\; \bigvee_{q\in Q}\Big(Q_q \wedge \bigwedge_{q'\in Q\setminus\{q\}} \sim Q_{q'}\Big)$$

- Every tape cell contains one symbol:

$$g_2 \;:=\; \bigwedge_{i=0}^{m}\bigvee_{a\in\Gamma}\Big(C_{a,i} \wedge \bigwedge_{a'\in\Gamma\setminus\{a\}} \sim C_{a',i}\Big)$$

- The head points to one position on the tape:

$$g_3 \;:=\; \bigvee_{i=1}^{m-1}\Big(H_i \wedge \bigwedge_{j\neq i} \sim H_j\Big) \wedge \sim H_0 \wedge \sim H_m$$

- Unread cells are unchanged in the next configuration:

$$g_4 \;:=\; \bigwedge_{i=0}^{m}\bigwedge_{a\in\Gamma}\Big(\sim H_i \wedge C_{a,i} \rightarrow \Box C_{a,i}\Big)$$

- Transition relation for and-states:

$$g_5 := \bigwedge_{i=1}^{m-1}\bigwedge_{a\in\Gamma}\bigwedge_{\mathsf{type}(q)=\wedge}\Big(H_i \wedge C_{a,i} \wedge Q_q \rightarrow \bigwedge_{(q,a,q',b,R)\in\delta} \Diamond(H_{i+1} \wedge C_{b,i} \wedge Q_{q'}) \wedge \bigwedge_{(q,a,q',b,L)\in\delta} \Diamond(H_{i-1} \wedge C_{b,i} \wedge Q_{q'})\Big)$$

- Transition relation for or-states:

$$g_6 := \bigwedge_{i=1}^{m-1}\bigwedge_{a\in\Gamma}\bigwedge_{\mathsf{type}(q)=\vee}\Big(H_i \wedge C_{a,i} \wedge Q_q \rightarrow \bigvee_{(q,a,q',b,R)\in\delta} \Diamond(H_{i+1} \wedge C_{b,i} \wedge Q_{q'}) \vee \bigvee_{(q,a,q',b,L)\in\delta} \Diamond(H_{i-1} \wedge C_{b,i} \wedge Q_{q'})\Big)$$

- The accepting states:

$$g_F \;:=\; \bigvee_{q\in F} Q_q$$

- The initial configuration:

$$g_{in} \;:=\; Q_{q_0} \wedge H_1 \wedge \bigwedge_{i=1}^{n} C_{a_i,i} \wedge C_{\sqcup,0} \wedge \bigwedge_{i=n+1}^{m} C_{\sqcup,i}$$

where input $x = a_0 \ldots a_n$ and $\sqcup$ is the blank space.

Let $g = \bigwedge_{i=1}^{6} g_i$ and bounded PCTL-formula f be:

$$f \;:=\; g_{in} \wedge [g\,\mathsf{U}^k g_F]_{=1}.$$

(Recall that $k = 2^m$.) Due to the one-to-one correspondence between the models of f and runs of A on input x, it follows that f is satisfiable iff ATM A accepts input x. Observe that the encoding takes $O((|\Gamma|+|Q|+|\delta|)\cdot S(n))$ time. If $S(n)$ is a polynomial function, then f is constructed in polynomial time in terms of the size of A and $|x| = n$. Furthermore, if $S(n)$ is polynomial in n, then $A \in \mathsf{EXPTIME}$ (equation (2)), which yields the desired result. □

6.2 Variable elimination procedure

Consider the ring of polynomials $D[X]$ in the *integral domain* D for the set X of variables. A polynomial $p(x_1, \ldots, x_n)$, with variables $x_1, \ldots, x_n \in X$, is seen as a sum of products with non-zero coefficients in D. Each $x_1^{d_1} \ldots x_n^{d_n}$ is called a *term*; together with its coefficient it is a *monomial*. The *degree* of the term $x_1^{d_1} \ldots x_n^{d_n}$ is $d_1 + \ldots + d_n$; the degree of a polynomial is the maximum degree of its terms. A polynomial is multivariate if $|X| > 1$ and univariate if $|X| = 1$. The ring of multivariate polynomials $D[X]$ can be viewed as a ring of univariate polynomials $D[X\setminus\{x\}][x]$ with coefficients in the integral domain $D[X \setminus \{x\}]$, see [4, Theorem 2, pp. 63]. The degree of a term of a polynomial in $D[X \setminus \{x\}][x]$ is the power of x in that term.

Let $E(D[X])$ be the set of (in)equations (e.g., $x_1^2 - x_2 \geq 0.4$) where the left hand side is a polynomial (such as $x_1^2 - x_2$) in $D[X]$ and the right hand side (e.g. 0.4) is in D. A variable x is *independent* of $H \subseteq E(D[X])$ iff $H = H \cap E(D[X \setminus \{x\}])$, else it is *dependent*. The *quotient domain* $\mathcal{Q}(D)$ is the rational form of the type $\frac{f}{g}$ where $f, g \in D$.

A weighted tree T is a triple (V, E, w), where V is the set of vertices, $E \subseteq V \times V$ is the set of edges and w is an injective weight function from $E \rightarrow \mathcal{V}$, where $\mathcal{V}$ is a set of variables. Let $X = \mathsf{img}(w)$. Define relations next and parent as follows; for $x, y \in X$, $v, v', v_1, v_2 \in V$, with $w^{-1}(x) = (v_1, v)$ and $w^{-1}(y) = (v', v_2)$, $(x, y) \in \mathsf{next}$ iff $v = v'$, and $(x, y) \in \mathsf{parent}$ iff $v_1 = v'$. Let next^+ be the transitive closure of next. Consider a term $\sigma = x_1 \ldots x_k$ such that for every $1 \leq i < k$, $(x_i, x_{i+1}) \in \mathsf{next}$. Define $\mathsf{head}(\sigma) = x_1$, $\mathsf{tail}(\sigma) = x_k$ and $x_i \ldots x_k$ as a suffix of σ, for $1 \leq i \leq k$. Let $H \subseteq E(\mathbb{Q}[X])$ be a set of (in)equations satisfying for each $\xi \in H$:

P1. For all $x \in X$, $\mathsf{lhs}(\xi) \in \mathbb{Q}[X \setminus \{x\}][x]$ implies $\mathsf{degree}(\xi) \leq 1$.

P2. For each term $\sigma = x_1 \ldots x_k$ in ξ, $(x_i, x_{i+1}) \in \mathsf{next}$.

P3. If $\mathsf{lhs}(\xi) = a_1\sigma_1 + \ldots + a_k\sigma_k$ where $a_i \in \mathbb{Q}$ and σ_i are terms, then for all $1 \leq i, j \leq k$, $(\mathsf{head}(\sigma_i), \mathsf{head}(\sigma_j)) \in \mathsf{parent}$.

Suppose $H \subseteq E(\mathbb{Q}[X])$ satisfies properties P1 through P3. Let $n = |X|$ and m be the number of (in)equations in H. We only consider positive variable valuations, i.e., for every variable x the in-equation $x > 0$ is in H. We present a non-deterministic algorithm to decide whether H is satisfiable. We begin by setting $H_0 = H$ and at each iteration i, we eliminate a (particular) variable, say x, and transform the set $H_i \subseteq E(\mathbb{Q}[X])$ of equations to $H_{i+1} \subseteq E(\mathbb{Q}[X\setminus\{x\}])$. Let binary comparison operator $\bowtie$ to be of the type $\{\geq, =, \leq\}$. (Strict inequalities can be removed by adding very small positive quantity ϵ. For example $f < g$ can be transformed to $f + \epsilon \leq g$.) The algorithm proceeds in the following steps:

1. If H_i is independent of all variables, then each (in)equation involves only rational numbers (and $\epsilon \rightarrow^+ 0$)[9]. Return true iff each (in)equality in H_i is true.
2. Choose a variable x such that every variable y with $(x, y) \in \mathsf{next}^+$, is independent of H_i.
3. H_x is the largest subset of H_i such that every formula in H_x is dependent on x. If H_x is empty, then $H_{i+1} = H_i$. Suppose H_x is not empty, every inequation $\xi \in H_x$ can be transformed to a form $(\sigma x \bowtie a_0 + a_1\sigma_1 + \ldots +$

[9] ϵ tends to 0 from the positive side.

$a_k\sigma_k)$, where $\sigma, \sigma_1, \ldots, \sigma_k$ are terms in $\mathbb{Q}[X \setminus \{x\}]$ and $a_0, \ldots, a_k \in \mathbb{Q}$. We will denote this form by $f{\cdot}x \bowtie g$. Set $H_{i+1} = H_i \setminus H_x$.

4. Define $\Lambda_{\bowtie} \subseteq \mathcal{Q}(\mathbb{Q}[X \setminus \{x\}])$, for $\bowtie \in \{\geq, =, \leq\}$ as follows:
   - $\Lambda_{\leq} := \{\frac{g}{f} \mid (f{\cdot}x \leq g) \in H_x\} \cup \{1\}$, quotients that are at least as large as x.
   - $\Lambda_{=} := \{\frac{g}{f} \mid (f{\cdot}x = g) \in H_x\}$, quotients that are equal to x.
   - $\Lambda_{\geq} := \{\frac{g}{f} \mid (f{\cdot}x \geq g) \in H_x\} \cup \{\epsilon\}$ quotients that are at least as small as x.
5. Non-deterministically choose an ordering of elements in $\Lambda_{\leq}$ and $\Lambda_{\geq}$. Then we have the following set of (in)equations:
$$\frac{g_1}{f_1} \leq \ldots \leq \frac{g_{n_1}}{f_{n_1}} \leq \frac{g_{n_1+1}}{f_{n_1+1}} = \ldots = \frac{g_{n_2}}{h_{n_2}} \leq \frac{g_{n_2+1}}{f_{n_2+1}} \leq \ldots \leq \frac{g_{n_3}}{f_{n_3}} \quad (3)$$
   where, $\frac{g_i}{f_i}$ is in $\Lambda_{\leq}$ for $1 \leq i \leq n_1$, in $\Lambda_{=}$ for $n_1 + 1 \leq i \leq n_2$ and in $\Lambda_{\geq}$ for $n_2 + 1 \leq i \leq n_3$.
6. For each $1 \leq j \leq n_3$, we have $\xi_j := (g_j f_{j+1} \bowtie g_{j+1} f_j)$. ξ'_j is obtained from ξ_j by canceling variables that are common divisors of the polynomials in the left hand side and in the right hand side of ξ_j. Add ξ'_j to H_{i+1} for each ξ_j $(1 \leq j \leq n_3)$. Go to step 1.

First we will show that H_{i+1} created in step 6, satisfies P1 through P3. Consider:
$$\frac{a_0 + a_1\sigma_1 + \ldots + a_k\sigma_k}{\sigma} \bowtie \frac{b_0 + b_1\sigma'_1 + \ldots + b_l\sigma'_l}{\sigma'} \quad (4)$$
Let $\xi := (\sigma{\cdot}x \bowtie a_0 + a_1\sigma_1 + \ldots + a_k\sigma_k)$, $\xi' := (\sigma'{\cdot}x \bowtie b_0 + b_1\sigma'_1 + \ldots + b_l\sigma'_l)$ and $\xi, \xi' \in H_i$ satisfy P1–P3. From the choice of the variable x (step 2), it is evident that either $\sigma|\sigma'$ or $\sigma'|\sigma$ ($a|b$ means a divides b). W.l.o.g assume $\sigma''\sigma' = \sigma$. The crucial observation is that if $\sigma'|\sigma$, then σ' is a suffix of σ, as there exists a variable y, such that $(x, y) \in \mathsf{next}$ and y is not independent of H_i.

Therefore, equation (4) can be rewritten as:
$$a_0 + a_1\sigma_1 + \ldots + a_k\sigma_k \bowtie b_0\sigma'' + b_1\sigma''\sigma'_1 + \ldots + b_l\sigma''\sigma'_l. \quad (5)$$
Condition P3 holds for equation (5) as $\mathsf{head}(\sigma) = \mathsf{head}(\sigma_i) = \mathsf{head}(\sigma'')$ for $1 \leq i \leq k$. We have $(\mathsf{tail}(\sigma''), \mathsf{head}(\sigma')) \in \mathsf{next}$, since $\sigma = \sigma''\sigma'$ and $(\mathsf{head}(\sigma'_i), \mathsf{head}(\sigma'_j)) \in \mathsf{parent}$ for all $1 \leq i, j \leq l$. Thus, the new equations added to H_{i+1} (after cancelling common variables) also satisfy P1 and P2 (cancellation is valid since variables can only take positive value).

The correctness of the algorithm follows from the following arguments:

1. Suppose H_i is feasible and let ν be a satisfying valuation of the variables. Then there exists some order among the rational numbers obtained by substituting the values of the variables in the quotients $\{\frac{g(x_1,\ldots,x_n)}{f(x_1,\ldots,x_n)}\}$ in $\Lambda_{\leq}$ and $\Lambda_{\geq}$. If we choose this order as the ordering in the equation (3) and obtain H_{i+1} subsequently, then ν is also a satisfying valuation for the (in)equations H_{i+1}.
2. If H_{i+1} is satisfiable, then the (in)equations (3) are true for some value of $X \setminus \{x\}$. If $\Lambda_{=}$ is not empty, then set $x = \frac{g_{n_2}}{f_{n_2}}$, else choose a value for x such that $\frac{g_{n_1}}{f_{n_1}} \leq x \leq \frac{g_{n_2+1}}{f_{n_2+1}}$. The value thus chosen is strictly greater than 0, since $\epsilon \in \Lambda_{\geq}$. (Hence, the rational form and the cancellation of variables as defined in step 5 and step 6, respectively is valid.) This gives us a satisfying valuation of H_i.

Observe that at iteration i, the size of H_i is $O(|H|)$ and in every iteration we remove one variable and spend $O(mn)$ in obtaining H_{i+1} (modulo division of rational numbers). As the maximum number of iterations is n, this yields a total time complexity in $O(mn^2)$. Thus, the satisfiability of a set of polynomial equation satisfying P1 through P3 is in NP.

6.3 Product of parity automaton and game

The cross product between the deterministic parity automaton $A_f = (Q, \Sigma, q_0, \delta, F)$ and game graph $G_f = (V, E, v_0)$ yields the parity game $\mathcal{G}_f = (U, R, u_0, \Omega)$ where

- $U \subseteq V \times Q$
- $u_0 = (v_0, q_0)$
- The transition relation R is defined by:
  - If v is not a transitive vertex then: $(v, q), (v', q') \in R$ iff $q' = \delta(q, v)$.
  - If v is a transition vertex then: $(v, q), (v', \blacktriangle) \in R$ and $(v', \blacktriangle), (v'', q') \in R$ iff $q' = \delta(q, v')$.
- $\Omega : U \to \mathsf{Img}(\mathsf{F})$ such that $\Omega(v, q) = F(q)$. Recall F is a parity condition of A_f.

Note that $\blacktriangle$ is simply used as a placeholder and has no special meaning.