Inf. Process. Lett. 188 (2025) 106522

Contents lists available at ScienceDirect

Information Processing Letters

journal homepage: www.elsevier.com/locate/ipl

Robust probabilistic temporal logics

Martin Zimmermann

Aalborg University, Aalborg, Denmark

ARTICLE INFO

Keywords:
Probabilistic temporal logics
Robustness
Model-checking

ABSTRACT

We robustify PCTL and PCTL*, the most important specification languages for probabilistic systems, and show that robustness does not increase the complexity of their model-checking problems.

1. Introduction

Specifications of reactive systems are typically implications $\varphi_a \rightarrow \varphi_g$ where φ_a is an environment assumption and φ_g is a system guarantee, both specified in a temporal logic. Such a specification is satisfied whenever the assumption is violated, independently of the system's behaviour. Assume, for example, that both the assumption and the guarantee are invariants $\varphi_a = \square\psi_a$ and $\varphi_g = \square\psi_g$ for propositional formulas ψ_g and ψ_a. Then, the specification $\square\psi_a \rightarrow \square\psi_g$ is satisfied if the formula ψ_a is violated just once, even if the formula ψ_g never holds. Such a behaviour is clearly undesirable, but the classical semantics of temporal logics are not sufficiently robust to deal with violations of the environment assumption.

Considerable effort has been put into overcoming this "defect" to provide robust semantics for temporal logics. However, the notion of robustness is hard to formalize, which is witnessed by the plethora of incomparable notions of robustness in the literature on verification (see, e.g., the introduction of [1] for a recent overview). Here, we further develop an approach due to Tabuada and Neider based on a novel, robust semantics for temporal logics, originally introduced for Linear Temporal Logic (LTL) [2]. They argue that there are four canonical degrees a formula of the form $\square\psi$ can be violated:

1. ψ is violated only finitely often.
2. ψ is violated infinitely often, but also holds infinitely often.
3. ψ is satisfied only finitely often.
4. ψ is never satisfied.

Note that there is a natural order between these cases. Consequently, their robust semantics uses five truth values, one for satisfaction and four more to capture the four degrees of violation. Furthermore, Tabuada and Neider defined the semantics of implication such that $\square\psi_a \rightarrow \square\psi_g$ is satisfied whenever the degree of violation of the guarantee $\square\psi_g$ is not more severe than the violation of the assumption $\square\psi_g$. Thus, the semantics indeed robustly handles violations of environment assumptions.

The resulting logic, called robust LTL (rLTL), has been extensively studied with very encouraging results: robustness can be added without increasing the complexity of model-checking and synthesis [1–3], robust semantics increases the usefulness of runtime monitoring [4], and rLTL can even be extended with increased expressiveness or timing constraints, again without an increase in complexity [5]. This approach towards robustness even extends to other temporal logics, e.g., branching-time logics like CTL and CTL* [6] and alternating-time logics like ATL and ATL* [7], where robustness can again be added without increasing the complexity of the most important verification problems.

Beyond the fact that this form of robustness comes for free (in terms of computational complexity), it only changes the semantics of the logics, but not the syntax. Furthermore, these logics are also evaluated over classical transition systems with the classical binary satisfaction relation for atomic propositions, i.e., robustness does not emerge from multi-valued semantics of the models (which might be hard to determine), but purely from the semantics. These aspects allow for a smooth transition from classical semantics to robust semantics for temporal logics. In conclusion, Tabuada and Neider introduced a natural and lightweight approach to add robustness that is applicable to a wide range of logics.

However, these logics capture only robustness in the temporal dimension, i.e., they are concerned with a single execution. Statements like "99% of the executions answer each request eventually" require robustness in terms of the whole set of executions, which is orthogonal to the capabilities of the robust logics studied thus far. To express such specifications, Hansson and Jonsson introduced probabilistic CTL

E-mail address: mzi@cs.aau.dk.

https://doi.org/10.1016/j.ipl.2024.106522
Received 9 June 2023; Received in revised form 30 July 2024; Accepted 31 July 2024
Available online 8 August 2024

(PCTL) [8], while Aziz, Singhal, and Balarin introduced probabilistic CTL* (PCTL*) [9]. PCTL and PCTL* replace the existential and universal quantification over paths in CTL and CTL* by the probabilistic operator $\mathcal{P}_I(\Phi)$, where $I \subseteq [0,1]$ is an interval with rational endpoints and Φ is a property of paths. Intuitively, $\mathcal{P}_I(\Phi)$ is satisfied in a state s if the probability that a path starting in s satisfies Φ is in the interval I. As CTL, PCTL requires each temporal operator to be preceded by a $\mathcal{P}$ while PCTL* (as CTL*) allows arbitrary nesting of Boolean connectives, temporal operators, and $\mathcal{P}$. For example, the property "99% of the executions answer each request eventually" is expressed by the PCTL* formula $\mathcal{P}_{\geq .99}(\Box(q \rightarrow \Diamond p))$, where q represents a request and p a response.

In this work, we further the study of robust semantics for temporal logics a la Tabuada and Neider by robustifying PCTL and PCTL*, obtaining the logics rPCTL and rPCTL*. In line with the design goals of the approach, the robust variants have (essentially) the same syntax as the non-robust variants and are evaluated over the same structures, simplifying the transition from the non-robust to the robust setting. The semantics of rPCTL and rPCTL* also follow the blueprint, i.e., they are five-valued employing the four degrees of violation described above. This simplifies the transition from robust semantics for linear, branching, and alternating time to the robust probabilistic setting.

As our main contribution, we show that this robustification comes again for free: the automata-based model-checking algorithms for rPCTL and rPCTL* can be generalized to the robust semantics. This result is in line with those on the robust temporal logics studied thus far, once more showing the versatility of robustness a la Tabuada and Neider.

2. Preliminaries

We denote the set of non-negative integers by $\mathbb{N}$. Throughout the paper, we fix a finite set AP of atomic propositions we use to label our models and to build our formulas. For algorithmic purposes, we assume that all probabilities used in the following are rational.

A discrete-time Markov chain (DTMC) $\mathcal{M} = (S, s_I, \delta, \ell)$ consists of a finite set S of states containing the initial state s_I, a (stochastic) transition function $\delta : S \times S \rightarrow [0,1]$ satisfying $\sum_{s' \in S} \delta(s,s') = 1$ for all $s \in S$, and a labelling function $\ell : S \rightarrow 2^{AP}$. The size $|\mathcal{M}|$ of $\mathcal{M}$ is defined as $\sum_{s,s' \in S} |\delta(s,s')|$, where $|p|$ denotes the length of the binary encoding of $p \in \mathbb{Q}$.

A path of $\mathcal{M}$ is an infinite sequence $\pi = s_0 s_1 s_2 \cdots \in S^\omega$ such that $\delta(s_n, s_{n+1}) > 0$ for all $n \in \mathbb{N}$. We say that π starts in s_0. For $n \in \mathbb{N}$, we write $\pi(n) = s_n$ for the n-th state of π and $\pi[n,\infty) = s_n s_{n+1} s_{n+2} \cdots$ for the suffix of π starting at position n. We write $\Pi(\mathcal{M}, s)$ for the set of all paths of $\mathcal{M}$ starting in $s \in S$ and define $\Pi(\mathcal{M}) = \bigcup_{s \in S} \Pi(\mathcal{M}, s)$.

The probability measure μ_s on sets of paths starting in some state $s \in S$ is defined as usual: Fix some non-empty path prefix $\rho = s_0 \cdots s_n$. The probability of the cylinder set $C_\rho = \{\pi \in \Pi(\mathcal{M}, s) \mid \rho \text{ is a prefix of } \pi\}$ is

$$\mu_s(C_\rho) = \begin{cases} 0 & \text{if } s_0 \neq s, \\ \prod_{j=0}^{n-1} \delta(\rho(j), \rho(j+1)) & \text{if } s_0 = s. \end{cases}$$

Using Carathéodory's extension theorem, we lift μ_s to a measure on the σ-algebra induced by the cylinder sets of path prefixes starting in s (see, e.g., [10, Theorem 1.41] for details. All sets of paths used in the following are ω-regular (see, e.g., [11, Chapter 1] for background on ω-regular languages) and therefore measurable.

3. Robust PCTL

In this section, we robustify PCTL [8]. Following the general design goals of the robustification a la Tabuada and Neider, robust PCTL (rPCTL) and PCTL share the same syntax (but for the dots to distinguish them), i.e., the formulas of rPCTL are given by the grammar

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \mathcal{P}_{\sim\lambda}(\odot\varphi) \mid \mathcal{P}_{\sim\lambda}(\diamondsuit\varphi) \mid \mathcal{P}_{\sim\lambda}(\boxdot\varphi) \mid \mathcal{P}_{\sim\lambda}(\varphi \,\mathfrak{U}\, \varphi) \mid \mathcal{P}_{\sim\lambda}(\varphi \,\mathfrak{R}\, \varphi)$$

where p ranges over AP, $\sim \in \{<, \leq, =, \geq, >\}$, and $\lambda \in [0,1]$ is a rational probability threshold. The size $|\varphi|$ of a formula φ is defined as the number of subformulas of φ plus the maximal length $|\lambda|$ of the binary encodings of the thresholds $\lambda \in \mathbb{Q}$ appearing in φ. Note that we have all Boolean operators in the grammar, as the semantics of negation is non-classical and implication cannot be derived from negation and disjunction. This is due to the five-valued robust semantics (see [2, Section 3.3] for a detailed discussion). For didactic reasons, we also prefer to explicitly have the operators eventually ($\diamondsuit$) and always ($\boxdot$) as they already capture the essence of the robust semantics. The robust semantics of until ($\mathfrak{U}$) and release ($\mathfrak{R}$) then generalize these.

Again, following the design goals of the robustification a la Tabuada and Neider, rPCTL is evaluated over the same structures as PCTL, i.e., over discrete-time Markov chains. Let $\mathcal{M} = (S, s_I, \delta, \ell)$ be a DTMC. The semantics of rPCTL is defined via an evaluation function $V_\mathcal{M}$ mapping a vertex s of $\mathcal{M}$ and a formula φ to a truth value in the (ordered) set $\mathbb{B}_4 = \{1111 \succ 0111 \succ 0011 \succ 0001 \succ 0000\}$. Given a truth value $t = b_1 b_2 b_3 b_4 \in \mathbb{B}_4$, we write $t[k]$ for b_k.

The evaluation function is defined inductively via

- $V_\mathcal{M}(s,p) = \begin{cases} 1111 & \text{if } p \in \ell(s), \\ 0000 & \text{if } p \notin \ell(s), \end{cases}$
- $V_\mathcal{M}(s,\neg\varphi) = \begin{cases} 1111 & \text{if } V_\mathcal{M}(s,\varphi) \prec 1111, \\ 0000 & \text{if } V_\mathcal{M}(s,\varphi) = 1111, \end{cases}$
- $V_\mathcal{M}(s,\varphi_0 \wedge \varphi_1) = \min(V_\mathcal{M}(s,\varphi_0), V_\mathcal{M}(s,\varphi_1))$,
- $V_\mathcal{M}(s,\varphi_0 \vee \varphi_1) = \max(V_\mathcal{M}(s,\varphi_0), V_\mathcal{M}(s,\varphi_1))$,
- $V_\mathcal{M}(s,\varphi_0 \rightarrow \varphi_1) = \begin{cases} 1111 & \text{if } V_\mathcal{M}(s,\varphi_0) \preceq V_\mathcal{M}(s,\varphi_1), \\ V_\mathcal{M}(s,\varphi_1) & \text{if } V_\mathcal{M}(s,\varphi_0) \succ V_\mathcal{M}(s,\varphi_1), \end{cases}$
- $V_\mathcal{M}(s,\mathcal{P}_{\sim\lambda}(\odot\varphi)) = b_1 b_2 b_3 b_4 \in \mathbb{B}_4$ where for all $k \in \{1,2,3,4\}$: $b_k = 1$ iff $\mu_s(\{\pi \in \Pi(\mathcal{M},s) \mid V_\mathcal{M}(\pi(1),\varphi)[k] = 1\}) \sim \lambda$,
- $V_\mathcal{M}(s,\mathcal{P}_{\sim\lambda}(\diamondsuit\varphi)) = b_1 b_2 b_2 b_4 \in \mathbb{B}_4$ where for all $k \in \{1,2,3,4\}$: $b_k = 1$ iff $\mu_s(\{\pi \in \Pi(\mathcal{M},s) \mid (V_\mathcal{M}(\pi(n),\varphi))[k] = 1 \text{ for some } n \in \mathbb{N}\}) \sim \lambda$, and
- $V_\mathcal{M}(s,\mathcal{P}_{\sim\lambda}(\boxdot\varphi)) = b_1 b_2 b_3 b_4 \in \mathbb{B}_4$ with
  - $b_1 = 1$ iff $\mu_s(\{\pi \in \Pi(\mathcal{M},s) \mid (V_\mathcal{M}(\pi(n),\varphi))[1] = 1 \text{ for all } n \in \mathbb{N}\}) \sim \lambda$,
  - $b_2 = 1$ iff $\mu_s(\{\pi \in \Pi(\mathcal{M},s) \mid (V_\mathcal{M}(\pi(n),\varphi))[2] = 1 \text{ for all but finitely many } n \in \mathbb{N}\}) \sim \lambda$,
  - $b_3 = 1$ iff $\mu_s(\{\pi \in \Pi(\mathcal{M},s) \mid (V_\mathcal{M}(\pi(n),\varphi))[3] = 1 \text{ for infinitely many } n \in \mathbb{N}\}) \sim \lambda$,
  - $b_4 = 1$ iff $\mu_s(\{\pi \in \Pi(\mathcal{M},s) \mid (V_\mathcal{M}(\pi(n),\varphi))[4] = 1 \text{ for some } n \in \mathbb{N}\}) \sim \lambda$,
- $V_\mathcal{M}(s,\mathcal{P}_{\sim\lambda}(\varphi \,\mathfrak{U}\, \psi)) = b_1 b_2 b_2 b_4 \in \mathbb{B}_4$ where for all $k \in \{1,2,3,4\}$: $b_k = 1$ iff $\mu_s(\{\pi \in \Pi(\mathcal{M},s) \mid \text{there exists } n \in \mathbb{N} \text{ s.t. } (V_\mathcal{M}(\pi(n),\psi))[k] = 1 \text{ and } (V_\mathcal{M}(\pi(n'),\varphi))[k] = 1 \text{ for all } n' < n\}) \sim \lambda$, and
- $V_\mathcal{M}(s,\mathcal{P}_{\sim\lambda}(\varphi \,\mathfrak{R}\, \psi)) = b_1 b_2 b_3 b_4 \in \mathbb{B}_4$ with
  - $b_1 = 1$ iff $\mu_s(\{\pi \in \Pi(\mathcal{M},s) \mid \text{for all } n \in \mathbb{N}\ (V_\mathcal{M}(\pi(n),\psi))[1] = 1 \text{ or } (V_\mathcal{M}(\pi(n),\varphi))[1] = 1 \text{ some } n' < n\}) \sim \lambda$,
  - $b_2 = 1$ iff $\mu_s(\{\pi \in \Pi(\mathcal{M},s) \mid (V_\mathcal{M}(\pi(n),\psi))[2] = 1 \text{ for all but finitely many } n \in \mathbb{N} \text{ or } (V_\mathcal{M}(\pi(n),\varphi))[2] = 1 \text{ some } n \in \mathbb{N}\}) \sim \lambda$,
  - $b_3 = 1$ iff $\mu_s(\{\pi \in \Pi(\mathcal{M},s) \mid (V_\mathcal{M}(\pi(n),\psi))[3] = 1 \text{ for infinitely many } n \in \mathbb{N} \text{ or } (V_\mathcal{M}(\pi(n),\varphi))[3] = 1 \text{ some } n' < n\}) \sim \lambda$, and
  - $b_4 = 1$ iff $\mu_s(\{\pi \in \Pi(\mathcal{M},s) \mid (V_\mathcal{M}(\pi(n),\psi))[4] = 1 \text{ for some } n \in \mathbb{N} \text{ or } (V_\mathcal{M}(\pi(n),\varphi))[4] = 1 \text{ some } n' < n\}) \sim \lambda$.

Here, the cases for Boolean connectives and temporal operators follow the blueprint introduced by Tabuada and Neider for rLTL (which also have been used for the robust variants of CTL, CTL*, ATL, and ATL*). For a detailed motivation and description, we refer to [2, Section 3]. On the other hand, the semantics of $\mathcal{P}$ generalizes the classical two-valued semantics of PCTL to five truth values, just as the path quantifiers in robust CTL [6] generalizes the path quantifiers of CTL and the

strategy quantifier of robust ATL [7] generalizes the strategy quantifier of ATL.

Example 1. Consider the formula $\varphi = \mathcal{P}_{\geq .9}(\boxdot a) \rightarrow \mathcal{P}_{\geq .95}(\boxdot g)$ expressing a robust assume-guarantee property. Assume φ evaluates to 1111 and consider the following cases:

- Assume $\mathcal{P}_{\geq .9}(\boxdot a)$ evaluates to 1111, i.e., with probability $\geq .9$, a holds at every position of a path. Then, by the semantics of the implication, with probability $\geq .95$, g holds at every position.
- Assume $\mathcal{P}_{\geq .9}(\boxdot a)$ evaluates to 0111, i.e., with probability $\geq .9$, a holds at all but finitely many positions of a path (but not at *every* position of a path with probability $\geq .9$). Then, by the semantics of the implication, with probability $\geq .95$, g holds at least at all but finitely many positions.
- Similar arguments hold for the truth values 0011 and 0001: Assume with probability $\geq .9$, a holds infinitely often (a holds at least once). Then, with probability $\geq .95$, g holds infinitely often) (g holds at least once).

Thus, the semantics of φ ensures that a violation of the assumption $\boxdot a$ is met with (at most) a proportional violation of the guarantee $\boxdot g$.

But we can even derive useful information if φ does not evaluate to 1111. Assume, φ evaluates to $t \prec 1111$. This can only be the case if the assumption $\mathcal{P}_{\geq .9}(\boxdot a)$ evaluates to some truth value strictly smaller than t and the guarantee $\mathcal{P}_{\geq .95}(\boxdot g)$ evaluates to t. Hence, even if the implication does not hold, it still yields the degree of satisfaction of the guarantee.

The above example shows that the robust semantics does indeed capture the intuition described in the introduction.

3.1. Expressiveness

In this section, we discuss the expressiveness of rPCTL; in particular, we compare it to the expressiveness of PCTL.

Our first result shows that rPCTL is at least as expressive as PCTL. It follows directly from the design goals of the robust semantics: they are defined such the first bit represents standard (non-robust) semantics.

Note that the restriction to implication-free formulas is just technical, as implications $\varphi \rightarrow \psi$ in PCTL formulas can always be rewritten as $\neg\varphi \vee \psi$. The need for the implication-removal stems from the fact that robust implication does not generalize classical implication [4, Footnote 3].

Lemma 1. *Let φ be a PCTL formula without implications, and let $\mathcal{M}$ be a DTMC with initial state s_I. Then, $\mathcal{M}, s_I \vDash \varphi$ iff $V_{\mathcal{M}}(s_I, \dot{\varphi}) = 1111$, where $\dot{\varphi}$ is the rPCTL formula obtained from φ by dotting all temporal operators.*

Proof. By induction over the construction of φ, formalizing the fact that the first bit of the robust semantics captures the classical semantics of PCTL. This can be seen by a careful inspection of the robust semantics. □

Corollary 1. *rPCTL is at least as expressive as PCTL.*

Let us briefly discuss the other inclusion, e.g., is rPCTL strictly more expressive than PCTL? This is true for the non-probabilistic setting, where rCTL (robust CTL) is strictly more expressive than CTL [6], as $V_{\mathcal{M}}(s_I, \forall\boxdot p) \geq 0111$ holds iff p holds at all but finitely many positions of every path starting in s. This property cannot be expressed in CTL [12, Theorem 6.21]. However, the analogous property "p holds at all but finitely many positions often with probability one" can be expressed in PCTL [12, Theorem 10.48] (when considering finite DTMCs), relying on the fact that a path ends up with probability one in a bottom strongly-connected component. We leave open the question whether similar arguments are sufficient to show that rPCTL can be embedded into PCTL (w.r.t. finite DTMCs).

Let us conclude this section with a consequence of the embedding proven in Lemma 1. rPCTL satisfiability asks, given a formula φ and a truth value t^* whether there is a DTMC $\mathcal{M}$ with initial state s_I such that $V_{\mathcal{M}}(s_I, \varphi) \geq t^*$. PCTL satisfiability has recently been shown to be undecidable [13]. So, due to Lemma 1, which allows us to embed PCTL in rPCTL, rPCTL satisfiability is also undecidable.

3.2. Model-checking

In this section, we prove that model-checking rPCTL is not harder than model-checking PCTL, which is in PTIME [8], i.e., robustness can be added for free. Formally, rPCTL model-checking is the following problem: Given a DTMC $\mathcal{M}$ with initial state s_I, an rPCTL formula φ, and a truth value $t^* \in \mathbb{B}_4$, is $V_{\mathcal{M}}(s_I, \varphi) \geq t^*$?

In the following, we prove that rPCTL model-checking is not harder than PCTL model-checking by combining techniques developed for robustified temporal logics with a generalization of an automata-based model-checking algorithm for PCTL.

Theorem 1. *rPCTL model-checking is in PTIME.*

Proof. Fix a DTMC $\mathcal{M} = (S, s_I, \delta, \ell)$ and an rPCTL formula φ, and let $\mathrm{cl}(\varphi)$ denote the set of subformulas of φ (which is defined as expected). We show how to inductively compute the satisfaction sets

$$\mathrm{Sat}(\psi, t) = \{s \in S \mid V_{\mathcal{M}}(s, \psi) \geq t\}$$

for $\psi \in \mathrm{cl}(\varphi)$ and $t \in \mathbb{B}_4$. Note that $\mathrm{Sat}(\psi, 0000) = S$ holds for all subformulas ψ. Hence, in the following, we only consider $t \succ 0000$. Also, the cases for atomic propositions and Boolean connectives are trivial, as they amount to Boolean combinations of already computed sets (see, e.g., [6]). For example, we have $\mathrm{Sat}(\psi' \wedge \psi'', t) = \mathrm{Sat}(\psi', t) \cap \mathrm{Sat}(\psi'', t)$ and $\mathrm{Sat}(\psi' \vee \psi'', t) = \mathrm{Sat}(\psi', t) \cup \mathrm{Sat}(\psi'', t)$. Hence, it only remains to consider subformulas ψ of the form $\mathcal{P}_{\sim\lambda}(\odot\psi')$, $\mathcal{P}_{\sim\lambda}(\diamondsuit\psi')$, $\mathcal{P}_{\sim\lambda}(\boxdot\psi')$, $\mathcal{P}_{\sim\lambda}(\psi' \,\mathrm{U}\, \psi'')$, or $\mathcal{P}_{\sim\lambda}(\psi' \,\mathrm{R}\, \psi'')$.

We begin with the next operator. Here, we have $s \in \mathrm{Sat}(\mathcal{P}_{\sim\lambda}(\odot\psi'), t)$ iff

$$\mu_s(\{\pi \in \Pi(\mathcal{M}, s) \mid \pi(1) \in \mathrm{Sat}(\psi', t)\}) = \left(\sum\nolimits_{s' \in \mathrm{Sat}(\psi', t)} \delta(s, s')\right) \sim \lambda.$$

The value $\sum_{s'} \delta(s, s')$ can be computed and compared to λ in polynomial time, as $\mathrm{Sat}(\psi, t)$ has already been computed by induction hypothesis.

For the remaining temporal operators, we rely on standard automata-theoretic characterizations of the sets of paths satisfying a temporal formula (see, e.g., [11, Section 1] for an introduction to automata on infinite words). We will then apply the following result due to Baier et al.: Given a DTMC $\mathcal{M}$, one of its states s, and an unambiguous Büchi automaton[1] with n states accepting a language L, the probability $\mu_s(L)$ can be computed in polynomial time in $|\mathcal{M}|$ and n [14, Theorem 2].

We begin by considering the always operator and then deal with the remaining operators, as we can reuse the machinery developed for the always operator to deal with them. By definition, we have $s \in \mathrm{Sat}(\mathcal{P}_{\sim\lambda}(\boxdot\psi'), t)$ iff

- $t = 1111$ and $\mu_s(\mathrm{Safe}(\mathrm{Sat}(\psi', 1111))) \sim \lambda$,
- $t = 0111$ and $\mu_s(\mathrm{CoB\ddot{u}chi}(\mathrm{Sat}(\psi', 0111))) \sim \lambda$,
- $t = 0011$ and $\mu_s(\mathrm{B\ddot{u}chi}(\mathrm{Sat}(\psi', 0011))) \sim \lambda$, and
- $t = 0001$ and $\mu_s(\mathrm{Reach}(\mathrm{Sat}(\psi', 0001))) \sim \lambda$,

where

[1] An automaton is unambiguous if it has at most one accepting run on every input.

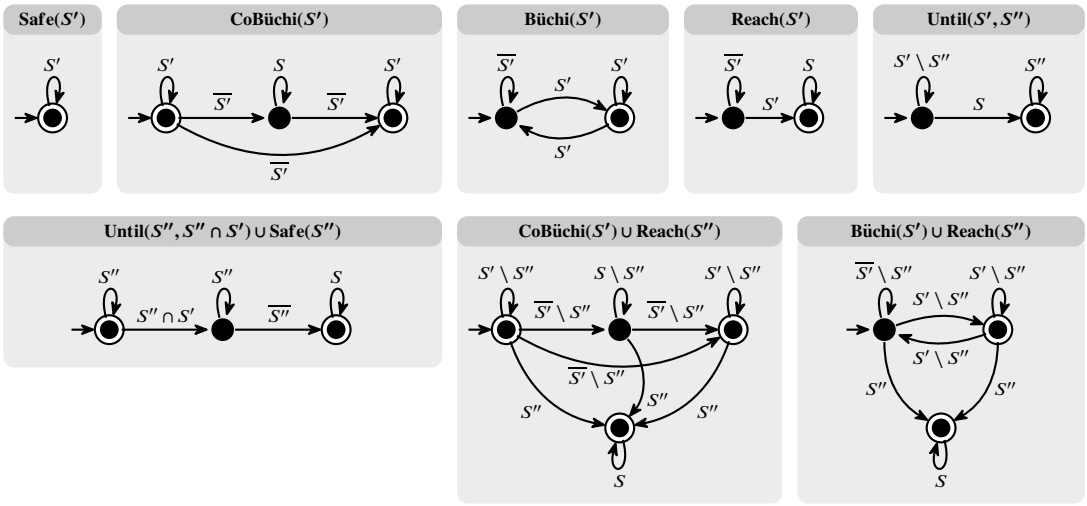

Fig. 1. The unambiguous Büchi automata for the path properties used in the rPCTL model-checking algorithm. Transitions are labelled by sets of states that represent all states in them (recall that S is the set of all states while S' and S'' are subsets of S). $\overline{S'}$ denotes the complement of S' w.r.t. S.

- $\text{Safe}(S') = \{\pi \in \Pi(\mathcal{M}) \mid \pi(n) \in S' \text{ for all } n \in \mathbb{N}\}$,
- $\text{CoBüchi}(S') = \{\pi \in \Pi(\mathcal{M}) \mid \pi(n) \in S' \text{ for all but finitely many } n \in \mathbb{N}\}$,
- $\text{Büchi}(S') = \{\pi \in \Pi(\mathcal{M}) \mid \pi(n) \in S' \text{ for infinitely many } n \in \mathbb{N}\}$,
- $\text{Reach}(S') = \{\pi \in \Pi(\mathcal{M}) \mid \pi(n) \in S' \text{ for some } n \in \mathbb{N}\}$.

All these sets are accepted by some unambiguous Büchi automaton with at most three states (see Fig. 1). As the satisfiability sets $\text{Sat}(\psi', t)$ are already computed by induction assumption, we only need to compute $\mu_s(L)$ for these languages and compare it to the given threshold λ. This can be achieved in polynomial time as argued above.

Now, let us consider the eventually operator. By definition, we have $s \in \text{Sat}(\mathcal{P}_{\sim\lambda}(\Diamond\psi'), t)$ iff $\mu_s(\text{Reach}(\text{Sat}(\psi', t))) \sim \lambda$, which we have just seen how to check in polynomial time. For the until operator, we have $s \in \text{Sat}(\mathcal{P}_{\sim\lambda}(\psi' \,\mathrm{U}\, \psi''), t)$ iff $\mu_s(\text{Until}(\text{Sat}(\psi', t), \text{Sat}(\psi'', t))) \sim \lambda$, where

$$\text{Until}(S', S'') = \{\pi \in \Pi(\mathcal{M}) \mid \text{there is an } n \in \mathbb{N} \text{ s.t. } \pi(n) \in S''$$
$$\text{and } \pi(n') \in S' \text{ for all } n' < n\}.$$

There is an unambiguous Büchi automaton with two states accepting this language (see Fig. 1). Thus, $\mu_s(\text{Until}(\text{Sat}(\psi', t), \text{Sat}(\psi'', t))) \sim \lambda$ can again be checked in polynomial time.

Finally, we consider the release operator. By definition, we have $s \in \text{Sat}(\mathcal{P}_{\sim\lambda}(\psi' \,\mathrm{R}\, \psi''), t)$ iff

- $t = 1111$ and $\mu_s([\text{Until}(\text{Sat}(\psi'', 1111), \text{Sat}(\psi', 1111) \cap \text{Sat}(\psi'', 1111))] \cup \text{Safe}(\text{Sat}(\psi'', 1111))) \sim \lambda$,
- $t = 0111$ and $\mu_s(\text{CoBüchi}(\text{Sat}(\psi'', 0111) \cup \text{Reach}(\text{Sat}(\psi', 0111)))) \sim \lambda$,
- $t = 0011$ and $\mu_s(\text{Büchi}(\text{Sat}(\psi'', 0011) \cup \text{Reach}(\text{Sat}(\psi', 0011))) \sim \lambda$, and
- $t = 0001$ and $\mu_s(\text{Reach}(\text{Sat}(\psi'', 0001) \cup \text{Reach}(\text{Sat}(\psi', 0001))) \sim \lambda$.

Note that $\text{Reach}(S') \cup \text{Reach}(S'') = \text{Reach}(S' \cup S'')$ for all sets S' and S'', i.e., we can rely on the results for Reach shown above.

Again, all these languages are accepted by unambiguous Büchi automata (see Fig. 1) with at most four states, which implies that we can again decide $\mu_s(L) \sim \lambda$ in polynomial time for these languages L.

Altogether, our algorithm inductively computes $5|\text{cl}(\varphi)|$ many satisfaction sets, each one in polynomial time (in $|\mathcal{M}|$), and then checks whether $s_I \in \text{Sat}(\varphi, t^*)$. Thus, the algorithm has polynomial running time. □

Again, this result is in line with previous work on robustifying temporal logics: The robustification comes for free (here in terms of computational complexity of the model-checking problem) and the algorithms for the classical semantics can be adapted to handle the robust semantics as well.

4. Robust PCTL*

In this section, we robustify PCTL* [9]. In line with the general approach, rPCTL* and PCTL* share the same syntax (but for the dots), i.e., the formulas of rPCTL are either state formulas or path formulas. State formulas are given by the grammar

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \mid \mathcal{P}_{\sim\lambda}(\Phi)$$

where p ranges over AP, $\sim \in \{<, \leq, =, \geq, >\}$, $\lambda \in [0, 1]$ is a rational probability threshold, and Φ ranges over path formulas. Path formulas are given by

$$\Phi ::= \varphi \mid \neg\Phi \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \Phi \to \Phi \mid \bigcirc\Phi \mid \Diamond\Phi \mid \Box\Phi \mid \Phi \,\mathrm{U}\, \Phi \mid \Phi \,\mathrm{R}\, \Phi$$

where φ ranges over state formulas. Formula size is defined as for rPCTL.

Also, rPCTL* is evaluated over discrete-time Markov chains, just as PCTL*. Let DTMC $\mathcal{M} = (S, s_I, \delta, \ell)$ be a DTMC. The semantics of rPCTL* is again defined via an evaluation function $V_{\mathcal{M}}$, this time mapping a vertex s of $\mathcal{M}$ and a state formula, or a path of $\mathcal{M}$ and a path formula to a truth value in $\mathbb{B}_4$.

As rPCTL* is designed to extend rPCTL, the definition of the rPCTL* semantics is (in some parts) very similar to that of rPCTL. This is in particular true for the Boolean connectives (both for state and path formulas), which is exactly the same as for rPCTL. However, to easily accommodate the arbitrary nesting of temporal operators in rPCTL*, we use an alternative definition of the semantics for path formulas, which mimics the original semantics for rLTL [2]. This follows the precedent of robust CTL* [6], where the semantics of path formulas is derived from the semantics of rLTL as well.

The rPCTL* evaluation function is defined inductively via

- $V_{\mathcal{M}}(s, p) = \begin{cases} 1111 & \text{if } p \in \ell(s), \\ 0000 & \text{if } p \notin \ell(s), \end{cases}$
- $V_{\mathcal{M}}(s, \neg\varphi) = \begin{cases} 1111 & \text{if } V_{\mathcal{M}}(s, \varphi) \prec 1111, \\ 0000 & \text{if } V_{\mathcal{M}}(s, \varphi) = 1111, \end{cases}$
- $V_{\mathcal{M}}(s, \varphi_0 \wedge \varphi_1) = \min(V_{\mathcal{M}}(s, \varphi_0), V_{\mathcal{M}}(s, \varphi_1))$,
- $V_{\mathcal{M}}(s, \varphi_0 \vee \varphi_1) = \max(V_{\mathcal{M}}(s, \varphi_0), V_{\mathcal{M}}(s, \varphi_1))$,

- $V_{\mathcal{M}}(s, \varphi_0 \to \varphi_1) = \begin{cases} 1111 & \text{if } V_{\mathcal{M}}(s,\varphi_0) \preceq V_{\mathcal{M}}(s,\varphi_1), \\ V_{\mathcal{M}}(s,\varphi_1) & \text{if } V_{\mathcal{M}}(s,\varphi_0) \succ V_{\mathcal{M}}(s,\varphi_1), \end{cases}$
- $V_{\mathcal{M}}(s, \mathcal{P}_{\sim\lambda}(\Phi)) = \max\{t \in \mathbb{B}_4 \mid \mu_s(\{\pi \in \Pi(\mathcal{M}, s) \mid V_{\mathcal{M}}(\pi, \Phi) \succeq t\}) \sim \lambda\}$ with the convention $\max \emptyset = 0000$,
- $V_{\mathcal{M}}(\pi, \varphi) = V_{\mathcal{M}}(\pi(0), \varphi)$,
- $V_{\mathcal{M}}(\pi, \neg\Phi) = \begin{cases} 1111 & \text{if } V_{\mathcal{M}}(\pi,\Phi) \prec 1111, \\ 0000 & \text{if } V_{\mathcal{M}}(\pi,\Phi) = 1111, \end{cases}$
- $V_{\mathcal{M}}(\pi, \Phi_0 \wedge \Phi_1) = \min(V_{\mathcal{M}}(\pi,\Phi_0), V_{\mathcal{M}}(\pi,\Phi_1))$,
- $V_{\mathcal{M}}(\pi, \Phi_0 \vee \Phi_1) = \max(V_{\mathcal{M}}(\pi,\Phi_0), V_{\mathcal{M}}(\pi,\Phi_1))$,
- $V_{\mathcal{M}}(\pi, \Phi_0 \to \Phi_1) = \begin{cases} 1111 & \text{if } V_{\mathcal{M}}(\pi,\Phi_0) \preceq V_{\mathcal{M}}(\pi,\Phi_1), \\ V_{\mathcal{M}}(s,\Phi_1) & \text{if } V_{\mathcal{M}}(\pi,\Phi_0) \succ V_{\mathcal{M}}(\pi,\Phi_1), \end{cases}$
- $V_{\mathcal{M}}(\pi, \odot\Phi) = V_{\mathcal{M}}(\pi[1,\infty), \Phi)$,
- $V_{\mathcal{M}}(\pi, \diamondsuit\Phi) = b_1b_2b_3b_4$ with $b_k = \max_{n\geq 0}(V_{\mathcal{M}}(\pi[n,\infty),\Phi))[k]$ for all $k \in \{1,2,3,4\}$,
- $V_{\mathcal{M}}(\pi, \boxdot\Phi) = b_1b_2b_3b_4$ with
  - $b_1 = \min_{n\geq 0}(V_{\mathcal{M}}(\pi[n,\infty),\Phi))[1]$,
  - $b_2 = \max_{m\geq 0}(\min_{n\geq m} V_{\mathcal{M}}(\pi[n,\infty),\Phi))[2]$,
  - $b_3 = \min_{m\geq 0}(\max_{n\geq m} V_{\mathcal{M}}(\pi[n,\infty),\Phi))[3]$, and
  - $b_4 = \max_{n\geq 0}(V_{\mathcal{M}}(\pi[n,\infty),\Phi))[4]$,
- $V_{\mathcal{M}}(\pi, \Phi \mathbin{\mathcal{U}} \Psi) = b_1b_2b_3b_4$ with $b_k = \max_{n\geq 0} \min\{(V_{\mathcal{M}}(\pi[n,\infty),\Psi))[k], \min\{(V_{\mathcal{M}}(\pi[n',\infty),\Phi))[k] \mid 0 \leq n' < n\}\}$ for all $k \in \{1,2,3,4\}$, and
- $V_{\mathcal{M}}(\pi, \Phi \mathbin{\mathcal{R}} \Psi) = b_1b_2b_3b_4$ with
  - $b_1 = \min_{n'\geq 0} \max\{(V_{\mathcal{M}}(\pi[n',\infty),\Psi))[1], \max_{n''<n'}(V_{\mathcal{M}}(\pi[n'',\infty),\Phi))[1]\}$,
  - $b_2 = \max_{n\geq 0} \min_{n'\geq n} \max\{(V_{\mathcal{M}}(\pi[n',\infty),\Psi))[2], \max_{n''<n'}(V_{\mathcal{M}}(\pi[n'',\infty),\Phi))[2]\}$,
  - $b_3 = \min_{n\geq 0} \max_{n'\geq n} \max\{(V_{\mathcal{M}}(\pi[n',\infty),\Psi))[3], \max_{n''<n'}(V_{\mathcal{M}}(\pi[n'',\infty),\Phi))[3]\}$, and
  - $b_4 = \max_{n'\geq 0} \max\{(V_{\mathcal{M}}(\pi[n',\infty),\Psi))[4], \max_{n''<n'}(V_{\mathcal{M}}(\pi[n'',\infty),\Phi))[4]\}$.

Note that while the definition of the semantics of the temporal operators differs from the one for rPCTL (to easily accomodate arbitrary nesting of temporal operators which is not possible in rPCTL), rPCTL is a fragment of rPCTL*.

Example 2. Consider the formula $\mathcal{P}_{\geq .9}(\boxdot a \to \boxdot g)$, a variant of the assume-guarantee property of Example 1. It evaluates to the largest truth value t such that $\boxdot a \to \boxdot g$ evaluates to t with probability $\geq .9$. Now, on a single path, $\boxdot a \to \boxdot g$ evaluates to

- 1111 if $\boxdot g$ evaluates to a larger or equal truth value than $\boxdot a$ and
- to $t \prec 1111$ if $\boxdot a$ evaluates to t and $\boxdot g$ evaluates to a truth value larger than t.

4.1. Expressiveness

As usual for temporal logics that allow arbitrary nesting of temporal operators (e.g., LTL, CTL*, and ATL*) rPCTL* has the same expressiveness as its non-robust version: the first bit of the five-valued semantics of rPCTL* again captures the semantics of non-robust PCTL* (as per design goals), thereby yielding the first embedding, while arbitrary nesting of temporal operators allows to mimic the five-valued semantics of rPCTL* explicitly in non-robust PCTL*, there yielding the second embedding.

Theorem 2. *rPCTL* is as expressive as PCTL*. Both translations can be computed in polynomial time.*

Proof. The translation from PCTL* to rPCTL* is a generalization of the analogous result for PCTL and rPCTL (see Lemma 1): Let φ be a PCTL* state formula without implications, and let $\mathcal{M}$ be a DTMC with initial state s_I. Then, $\mathcal{M}, s_I \vDash \varphi$ iff $V_{\mathcal{M}}(s_I, \dot{\varphi}) = 1111$, where $\dot{\varphi}$ is the rPCTL* state formula obtained from φ by dotting all temporal operators. This is again proven by induction over the construction of φ.

For the other direction, we inductively translate an rPCTL* state formula φ and a truth value $t \in \mathbb{B}_4$ into a PCTL* state formula φ_t such that $V_{\mathcal{M}}(s,\varphi) \geq t$ iff $\mathcal{M}, s \vDash \varphi_t$ for all DTMCs $\mathcal{M}$ and all states s of $\mathcal{M}$. This is in line with previous work on LTL [2], CTL* [6], and ATL* [7].

As we have $V_{\mathcal{M}}(s,\varphi) \geq 0000$ for all state formulas φ, we define φ_{0000} to be some tautology (say $p \vee \neg p$) and only consider $t \succ 0000$ in the following. We start with atomic propositions and define $p_t = p$. The translation for Boolean connectives is the same for state and path formulas. So, to avoid duplication, χ ranges in the following over state and path formulas.

- $(\neg\chi)_t = \neg\chi_t$,
- $(\chi_1 \vee \chi_2)_t = (\chi_1)_t \vee (\chi_2)_t$ and $(\chi_1 \wedge \chi_2)_t = (\chi_1)_t \wedge (\chi_2)_t$,
- $(\chi_1 \to \chi_2)_{1111} = \bigwedge_{t\succeq 0000}(\chi_2)_t \vee \neg(\chi_1)_t$, and
- $(\chi_1 \to \chi_2)_t = (\chi_1 \to \chi_2)_{1111} \vee (\chi_2)_t$ for $t \prec 1111$.

Next, we define $(\mathcal{P}_{\sim\lambda}(\Phi))_t = \mathcal{P}_{\sim\lambda}(\Phi_t)$ and consider the temporal operators:

- $(\odot\Phi)_t = \bigcirc\Phi_t$ and $(\diamondsuit\Phi)_t = \diamondsuit\Phi_t$,
- $(\boxdot\Phi)_{1111} = \Box\Phi_{1111}$, $(\boxdot\Phi)_{0111} = \diamondsuit\Box\Phi_{0111}$, $(\boxdot\Phi)_{0011} = \Box\diamondsuit\Phi_{0011}$, and $(\boxdot\Phi)_{0001} = \diamondsuit\Phi_{0001}$,
- $(\Phi \mathbin{\mathcal{U}} \Psi)_t = \Phi_t \mathbin{\mathcal{U}} \Psi_t$,
- $(\Phi \mathbin{\mathcal{R}} \Psi)_{1111} = \Phi_{1111} \mathbin{\mathcal{R}} \Psi_{1111}$, $(\Phi \mathbin{\mathcal{R}} \Psi)_{0111} = \diamondsuit\Box\Psi_{0111} \vee \diamondsuit\Phi_{1111}$, and $(\Phi \mathbin{\mathcal{R}} \Psi)_{0011} = \Box\diamondsuit\Psi_{0011} \vee \diamondsuit\Phi_{1111}$, and $(\Phi \mathbin{\mathcal{R}} \Psi)_{0001} = \diamondsuit\Psi_{0001} \vee \diamondsuit\Phi_{1111}$.

An induction over the construction of φ shows that φ_t has the desired properties. □

4.2. Model-checking

The model-checking problem for rPCTL* is defined as for rPCTL: Given a DTMC $\mathcal{M}$ with initial state s_I, an rPCTL* state formula φ, and a truth value $t^* \in \mathbb{B}_4$, is $V_{\mathcal{M}}(s,\varphi) \geq t^*$? It is PSPACE-complete, as is the PCTL* model-checking problem [9,15], i.e., robustness comes again for free. This result follows directly from the fact that rPCTL* can be (in polynomial time) translated into PCTL* and follows previous results on robust CTL* [6] and robust ATL* [7].

Theorem 3. *rPCTL* model-checking is PSPACE-complete.*

Proof. The result follows immediately from Theorem 2 and the PSPACE-completeness of PCTL* model-checking. □

5. Related work

There is a plethora of work on the verification of probabilistic systems and on robustifying verification. Due to space restrictions, we focus here on the intersection of these two areas, which is our concern in this work.

A major challenge in the modelling of probabilistic systems is the fact that determining exact transition probabilities is often impossible. Instead one resorts to statistical analyses of the system, which comes with uncertainties.[2] However, verification results are often highly sensitive to changes in the transition probabilities, i.e., modelling and verification are not robust to those changes. Hence, a large body of work is concerned with capturing uncertainty in probabilistic systems and their subsequent verification.

Various types of uncertain transition functions for Markov chains have been introduced, e.g., interval bounded DTMCs [16] where only

[2] In fact, even the work introducing PCTL* considered models with unknown transition probabilities [9].

upper and lower bounds on the transition probabilities are specified, and convex MDPs, Markov decision processes with convex uncertainties [17], and robust MDPs with rectangular ambiguity sets [18,19]. However, there are uncertainties beyond the transition probabilities, e.g., in the form of partial observability and adversarial behaviour. A recent position paper by Badings et al. [20] gives a thorough overview of the state-of-the-art in decision making under uncertainty, presenting a survey of uncertainty models that enable more robust modelling and verification. Finally, other approaches to handling uncertainty include simulation [21,22] and approximation [23].

6. Conclusion

We have shown how to robustify PCTL and PCTL*, obtaining the logics rPCTL and rPCTL*. The model-checking problems for these robust logics are as hard as the model-checking problems for the non-robust variants, i.e., robustness can be added for free. This is in line with previous work on robust variants of LTL [1] and its extensions [5], as well as CTL and CTL* [6], and ATL and ATL* [7].

Probably the most interesting problem left for future work concerns the expressiveness of rPCTL and PCTL. Note that in the non-probabilistic setting, it is known that rCTL is strictly more expressive than CTL [6, Section 3.3]. However, as discussed in Subsection 3.1, it is unclear whether this separation can be lifted to the probabilistic setting.

CRediT authorship contribution statement

Martin Zimmermann: Writing – review & editing, Writing – original draft, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

We want to thank Marco Muñiz for proposing to study rPCTL and rPCTL* and for many fruitful discussions, as well as the reviewers for their valuable feedback.

This work was supported by DIREC – Digital Research Centre Denmark.

References

[1] T. Anevlavis, M. Philippe, D. Neider, P. Tabuada, Being correct is not enough: efficient verification using robust linear temporal logic, ACM Trans. Comput. Log. 23 (2022) 8:1–8:39.
[2] P. Tabuada, D. Neider, Robust linear temporal logic, in: J. Talbot, L. Regnier (Eds.), CSL 2016, in: LIPIcs, vol. 62, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, pp. 10:1–10:21.
[3] S.P. Nayak, D. Neider, M. Zimmermann, Robustness-by-construction synthesis: adapting to the environment at runtime, in: T. Margaria, B. Steffen (Eds.), ISoLA 2022, Part I, in: LNCS, vol. 13701, Springer, 2022, pp. 149–173.
[4] C. Mascle, D. Neider, M. Schwenger, P. Tabuada, A. Weinert, M. Zimmermann, From LTL to rLTL monitoring: improved monitorability through robust semantics, Form. Methods Syst. Des. 59 (2021) 170–204.
[5] D. Neider, A. Weinert, M. Zimmermann, Robust, expressive, and quantitative linear temporal logics: pick any two for free, Inf. Comput. 285 (2022) 104810.
[6] S.P. Nayak, D. Neider, R. Roy, M. Zimmermann, Robust computation tree logic, Innov. Syst. Softw. Eng. (2024), https://doi.org/10.1007/s11334-024-00552-7, in press.
[7] A. Murano, D. Neider, M. Zimmermann, Robust alternating-time temporal logic, in: S.A. Gaggl, M.V. Martinez, M. Ortiz (Eds.), JELIA, in: LNCS, vol. 14281, Springer, 2023, pp. 796–813.
[8] H. Hansson, B. Jonsson, A logic for reasoning about time and reliability, Form. Asp. Comput. 6 (1994) 512–535.
[9] A. Aziz, V. Singhal, F. Balarin, It usually works: the temporal logic of stochastic systems, in: P. Wolper (Ed.), CAV 1995, in: LNCS, vol. 939, Springer, 1995, pp. 155–165.
[10] A. Klenke, Probability Theory - a Comprehensive Course, Universitext, Springer, 2008.
[11] E. Grädel, W. Thomas, T. Wilke (Eds.), Automata, Logics, and Infinite Games: A Guide to Current Research, LNCS, vol. 2500, Springer, 2002.
[12] C. Baier, J. Katoen, Principles of Model Checking, MIT Press, 2008.
[13] M. Chodil, A. Kucera, The general and finite satisfiability problems for PCTL are undecidable, arXiv:2404.10648, 2024.
[14] C. Baier, S. Kiefer, J. Klein, D. Müller, J. Worrell, Markov chains and unambiguous automata, J. Comput. Syst. Sci. 136 (2023) 113–134.
[15] M.Y. Vardi, P. Wolper, An automata-theoretic approach to automatic program verification (preliminary report), in: LICS 1986, IEEE Computer Society, 1986, pp. 332–344.
[16] K. Sen, M. Viswanathan, G. Agha, Model-checking Markov chains in the presence of uncertainties, in: H. Hermanns, J. Palsberg (Eds.), TACAS 2006, in: LNCS, vol. 3920, Springer, 2006, pp. 394–410.
[17] A. Puggelli, W. Li, A.L. Sangiovanni-Vincentelli, S.A. Seshia, Polynomial-time verification of PCTL properties of MDPs with convex uncertainties, in: N. Sharygina, H. Veith (Eds.), CAV 2013, in: LNCS, vol. 8044, Springer, 2013, pp. 527–542.
[18] G.N. Iyengar, Robust dynamic programming, Math. Oper. Res. 30 (2005) 257–280.
[19] A. Nilim, L.E. Ghaoui, Robust control of Markov decision processes with uncertain transition matrices, Oper. Res. 53 (2005) 780–798.
[20] T.S. Badings, T.D. Simão, M. Suilen, N. Jansen, Decision-making under uncertainty: beyond probabilities, Int. J. Softw. Tools Technol. Transf. 25 (2023) 375–391.
[21] P. Ashok, J. Kretínský, M. Weininger, PAC statistical model checking for Markov decision processes and stochastic games, in: I. Dillig, S. Tasiran (Eds.), CAV 2019, Part I, in: LNCS, vol. 11561, Springer, 2019, pp. 497–519.
[22] W. Wiesemann, D. Kuhn, B. Rustem, Robust Markov decision processes, Math. Oper. Res. 38 (2013) 153–183.
[23] M. Jaeger, G. Bacci, G. Bacci, K.G. Larsen, P.G. Jensen, Approximating Euclidean by imprecise Markov decision processes, in: T. Margaria, B. Steffen (Eds.), ISoLA 2020, Part I, in: LNCS, vol. 12476, Springer, 2020, pp. 275–289.