

Counterexamples in Probabilistic LTL Model Checking for Markov Chains

Matthias Schmalz¹ Daniele Varacca² Hagen Völzer³

¹ETH Zurich, Switzerland

²PPS - CNRS & Univ. Paris Diderot, France

³IBM Zurich Research Laboratory, Switzerland

Abstract. We propose a way of presenting and computing a counterexample in probabilistic LTL model checking for discrete-time Markov chains. In qualitative probabilistic model checking, we present a counterexample as a pair (α, γ) , where α, γ are finite words such that all paths that extend α and have infinitely many occurrences of γ violate the specification. In quantitative probabilistic model checking, we present a counterexample as a pair (W, R) , where W is a set of such finite words α and R is a set of such finite words γ . Moreover, we suggest how the counterexample presented helps the user identify the underlying error in the system by means of an interactive game with the model checker.

1 Introduction

A counterexample in LTL model checking is an execution path that violates the LTL specification. This counterexample path should help the user identify and repair an error in the system. However, a counterexample path is in general infinite, and therefore if we want to show it to the user, we must find a finite representation. In classical LTL model checking, we can exploit the fact that a *periodic* counterexample always exists (see e.g. [17]), i.e., an execution path of the form $\alpha\gamma^\omega$, where α and γ are finite words.

In the probabilistic LTL model checking problem that we consider here, we are given an LTL formula Φ and a discrete-time finite-state Markov chain generating a probability measure \mathbb{P} , and we want to check whether $\mathbb{P}[\Phi] > t$ (or $\mathbb{P}[\Phi] \geq t$). A counterexample witnessing the violation of this assertion is therefore a set Y of execution paths violating Φ such that $\mathbb{P}[Y] \geq 1 - t$ (or $\mathbb{P}[Y] > 1 - t$). In general such a set is not only infinite, but almost all of its paths are aperiodic. How can such a counterexample be presented to the user to provide useful debugging information?

In this paper, we show how a counterexample can be presented and computed and suggest how the user should interact with the model checker to find the error.

We start by considering the special case of qualitative probabilistic model checking, i.e., the question whether $\mathbb{P}[\Phi] = 1$. We propose to represent a qualitative counterexample as a pair (α, γ) , where

- α is a finite path such that *almost all* paths extending α violate the specification and hence the specification is violated with at least the probability of α . Therefore, α shows where the probability is lost.

This work was partially supported by the EU project Deploy (N. 214158).

- γ is a finite word in a bottom strongly connected component such that *all* paths that extend α and that have infinitely many occurrences of γ violate the specification. The word γ witnesses that almost all paths extending α violate the specification.

The pair (α, γ) is presented to the user in an interactive game with the model checker. The user tries to construct a path extending α and satisfying the specification, whereas the model checker ensures that γ occurs infinitely often. By failing to construct such a path the user finds an error in the system.

We then show that this approach can be extended to the quantitative case ($t < 1$), where in general a set W of such finite paths α and a set R of such finite words γ has to be considered.

Finally, we show how such a counterexample can be computed; we build on a model checking algorithm by Courcoubetis and Yannakakis [8], which however has to be substantially complemented for our purposes.

We discuss related work in Section 6. Missing proofs can be found in [19].

2 Preliminaries

We assume that the reader is familiar with Kripke structures, discrete-time Markov chains, linear temporal logic (LTL) and ω -regular languages. We briefly recall the basic definitions to introduce conventions and fix the notation. References for further reading are also provided.

2.1 Words

Let Q be a set of *states*. The sets of infinite, finite and nonempty finite words over Q are denoted Q^ω , Q^* and Q^+ , respectively. Usually q, p denote elements of Q ; $\alpha, \beta, \gamma, \delta$ elements of Q^* , x an element of Q^ω , z an element of $Q^\omega \cup Q^*$, and λ the empty word.

We write $\alpha \sqsubseteq z$ if α is a prefix of z . If $\alpha \sqsubseteq z$, z is called an *extension* of α . We define $\alpha \uparrow := \{x \mid \alpha \sqsubseteq x\}$ and $z \downarrow := \{\alpha \mid \alpha \sqsubseteq z\}$. Similarly, given $W \subseteq Q^*$ and $Y \subseteq Q^\omega \cup Q^*$, $W \uparrow := \bigcup \{\alpha \uparrow \mid \alpha \in W\}$ (the set of *extensions of W*) and $Y \downarrow := \bigcup \{z \downarrow \mid z \in Y\}$.

2.2 Probabilistic Systems

A *system* (Kripke structure) $\Sigma = (Q, S, \rightarrow, v)$ consists of a finite set Q of *states*, a nonempty set $S \subseteq Q$ of *initial states*, a *state relation* $\rightarrow \subseteq Q \times Q$ and a *valuation function* $v : Q \rightarrow 2^{AP}$ mapping each state q to a set $v(q) \subseteq AP$ of *atomic propositions*. We assume here that for each $q \in Q$ there is a $p \in Q$ so that $q \rightarrow p$. The *size* of Σ is $|\Sigma| := |Q| + |\rightarrow|$.

A *path fragment* (of Σ) is a word $q_0 q_1 \dots \in Q^\omega \cup Q^*$ such that $q_{i-1} \rightarrow q_i$, $i > 0$. A *path* (of Σ) is a path fragment qz with $q \in S$. The empty word is also a path and a path fragment of Σ . The set $path_{fin}(\Sigma)$ contains all finite, and $path_\omega(\Sigma)$ all infinite paths of Σ .

Often we view Σ as the directed graph (Q, \rightarrow) . A set $K \subseteq Q$ is a *strongly connected component* (of Σ) (scc for short) if it is a strongly connected component of (Q, \rightarrow) (see e.g. [7]). A *bottom strongly connected component* (of Σ) (bscc) is an scc K without outgoing edges, i.e., if $q \in K$ and $q \rightarrow p$, then $p \in K$.

A (*labelled discrete-time*) *Markov chain* (see e.g. [6, 15]) is a system $\Sigma = (Q, S, \rightarrow, v)$ equipped with *transition probabilities* given by $P : Q \times Q \rightarrow [0, 1]$ and *initial probabilities* given by $P_{ini} : Q \rightarrow [0, 1]$, where $P(q, p) > 0$ iff $q \rightarrow p$, $P_{ini}(q) > 0$ iff $q \in S$,

$\sum_{p \in Q} P(q, p) = 1$, and $\sum_{q \in Q} P_{ini}(q) = 1$. It is well known (see e.g. [6, 15]) that a Markov chain induces a measure \mathbb{P} on the σ -algebra $\mathcal{B}(Q^\omega)$ induced by the basic cylinder sets $\alpha \uparrow$, $\alpha \in Q^*$ with the property $\mathbb{P}[q_0 \dots q_n \uparrow] = P_{ini}(q_0) \prod_{i=1}^n P(q_{i-1}, q_i)$, $q_0 \dots q_n \in Q^+$. A measure induced by a Markov chain is called *Markov measure*. We later refer to a Markov chain simply as Σ, \mathbb{P} .

2.3 Temporal Properties

A (*linear-time temporal*) *property*, denoted Y, Z , is a subset of Q^ω . We mainly consider properties expressible in LTL (linear temporal logic [16]); we use the notation introduced in [11]. A formula in LTL is built from atomic propositions in AP , *true*, *false* and the boolean and temporal connectives $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$ and X, U, G, F . The *size* $|\Phi|$ of a formula is the number of its temporal and boolean connectives.

An LTL formula Φ is interpreted in the context of a system $\Sigma = (Q, S, \rightarrow, \nu)$ over words $x \in Q^\omega$. For $i \in \mathbb{N}$, $x, i \models \Phi$ means that x *satisfies* Φ *at position* i (in the usual sense [11]). Moreover, $x \models \Phi$ (“ x satisfies Φ ”) abbreviates $x, 0 \models \Phi$; $\Sigma \models \Phi$ (“ Σ satisfies Φ ”) means $x \models \Phi$ for all $x \in \text{path}_\omega(\Sigma)$. We write $\text{Sat}(\Sigma, \Phi)$ for the set of all infinite paths of Σ satisfying Φ . For convenience, we often write $\text{Sat}(\Phi)$ or Φ instead of $\text{Sat}(\Sigma, \Phi)$. In particular, $\text{Sat}(\text{true}) = \text{path}_\omega(\Sigma)$. A formula ϕ without temporal connectives is a *state formula*. For $q \in Q$, $q \models \phi$ (“ q satisfies ϕ ”) iff $qx \models \phi$ for all (or equivalently some) $x \in Q^\omega$.

To simplify the presentation, we suppose that for each $q \in Q$ there is an atomic proposition a_q that holds in q and only there. In our examples, we do not explicitly mention such atomic propositions a_q . For better readability of formulas, we write q instead of a_q ; $q_0 q_1 \dots q_n$ instead of $q_0 \wedge X(q_1 \wedge \dots \wedge X(q_n) \dots)$ and λ instead of *true*. These assumptions do not affect the results of the paper.

An ω -*regular property* is a property that is accepted by some *Büchi automaton*. Any LTL formula expresses an ω -regular property. Any ω -regular property is *measurable*, i.e., a member of $\mathcal{B}(Q^\omega)$ (see [21]).

3 Qualitative Counterexamples

In this section, we consider the question whether $\mathbb{P}[\Phi] = 1$, where Φ is an LTL formula and \mathbb{P} a Markov measure. Probabilistic satisfaction can be seen as a special form of quantification, but our traditional understanding of a counterexample is tightly connected with universal quantification. Say, we want to understand why the CTL* formula $A.\Phi$ does not hold, where Φ is an LTL formula. We display a classical linear counterexample path in this case. The system has more behaviour than expected, and the model checker displays the path as an example of the additional behaviour. The user can then replay the path to see where the actual behaviour deviates from her expectation, which is where she should find the error in the system.

The situation is different for existential quantification. Say we want to understand why a CTL* formula $E.\Phi$ is violated, again Φ being an LTL formula. For example, $E.\Phi$ could express the property that there exists a run of the system such that ‘someone wins the jackpot’. If the formula is false, the answer of the model checker is just ‘no’. The information to be returned could be the entire system showing the absence of a path satisfying Φ . Of course, that is not very informative. To find the error, we suggest

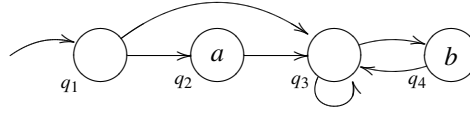
that the proof burden should be reversed, i.e., the user should try to display a witness for the formula. The user should have an idea on what the path should look like; in the example: she knows *how* someone could win the jackpot in the system. She can then try to replay that path. In doing so, she will find a point where the behaviour of the system deviates from her expectations, because the desired path does not exist in the system.

The interaction between user and model checker that we propose for the probabilistic case will be a mixture of the universal and the existential case.

3.1 Examples of Counterexamples

To approach the problem for Markov chains, let us now consider some examples. By default, the examples are based on the system $\Sigma = (Q, S, \rightarrow, v)$ below. For the qualitative case the particular transition probabilities of the Markov chain are not relevant (see e.g. [20]); hence we do not display them.

Each example considers an LTL formula Φ for which $\mathbb{P}[\Phi] < 1$, and in each case we will discuss how a counterexample should look.



Example 3.1. Let $\Phi = G \neg a$. As Φ is a safety property, if it is violated, there is a finite path α such that each extension of α violates Φ . This is the case for the path $\alpha = q_1 q_2$. Because $\mathbb{P}[q_1 q_2 \uparrow] > 0$, we have $\mathbb{P}[\Phi] < 1$. As in classical model checking, it is sufficient to display the violating finite path α to the user as a counterexample.

Example 3.2. Let $\Phi = F a$. There is a finite path $\alpha := q_1 q_3$ in the system such that each extension of α into $path_\omega(\Sigma)$ violates Φ , i.e., no extension of α into $path_\omega(\Sigma)$ contains an a -state. This clearly proves that $\mathbb{P}[\Phi] < 1$. In contrast to the previous example, not all extensions of α but only the extensions into $path_\omega(\Sigma)$ violate Φ . Hence, the inspection of α may not be sufficient to find the error; the user also has to take the structure of Σ into account. Similar to the CTL* case discussed above, the user who designed the system should have an idea on how to reach an a -state, once α has been executed. By trying to play such a path, which does not exist, she will eventually find the point in the system where the actual and the expected behaviour deviate.

Example 3.3. Let $\Phi = FG \neg b$. We recall that in any Markov chain a path eventually enters a bsc with probability one. For each reachable bsc K , a path eventually enters K with nonzero probability, and then, with probability 1, visits all states of K infinitely often. (These facts are well known and also follow from Lemma 4.3.) Any run that infinitely often visits a b -state violates Φ . The system above has a reachable bsc that contains a b -state, and therefore the specification is violated with nonzero probability.

To show that to the user, we propose that the model checker returns a b -state within a bsc, namely q_4 . The user then convinces herself that (i) the b -state indeed belongs to a reachable bsc and (ii) repeatedly visiting the b -state violates Φ . The latter point (ii) is straightforward in this case. To convince herself of (i), the user plays the following interactive game with the model checker: She tries to find a finite path fragment $q_4 \beta$ so that q_4 is unreachable after β . If she believes that Φ has probability 1, she has an idea of how to do so. The model checker then goes back to q_4 . The system must deviate from the expected behaviour in at least one of these two moves.

Example 3.4. Let $\Phi = GFb \Rightarrow Fa$. Repeatedly visiting a b -state without visiting an a -state violates Φ . The specification does not have probability 1, as there is a bsc containing a b -state but no a -state, and that bsc can be reached without passing through an a -state. We propose that the model checker outputs q_4 and $\alpha := q_1q_3$. The user then convinces herself that (i) q_4 belongs to a bsc and that α leads to that bsc, and (ii) any path starting with α , and visiting q_4 infinitely often violates Φ . To this end, she plays the following game with the model checker: The model checker plays α . To convince herself of (i), the user tries to extend α so that q_4 becomes unreachable; the model checker then goes back to q_4 . If that does not help discover the error, she tries to refute (ii) by extending α to $\alpha\beta \in path_{fin}(\Sigma)$ so that β visits an a -state.

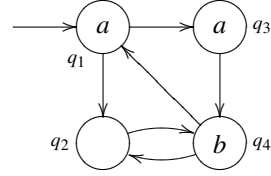
In Examples 3.1 and 3.2, a counterexample is represented by a finite path α such that $\Sigma \models \alpha \Rightarrow \neg\Phi$. This representation is not sufficiently expressive for Examples 3.3 and 3.4. Therefore we use the more general representation (α, q) , $\alpha \in path_{fin}(\Sigma)$, $q \in Q$ such that $\Sigma \models \alpha \wedge GFq \Rightarrow \neg\Phi$. Note that, in the above examples, the formula Φ is violated after α with probability one, and q witnesses that. The state q is in particular important when α leads to a large bsc.

However, there still are situations, in which counterexamples of the form (α, q) cannot be found, and we need a path fragment instead of the single state q :

Example 3.5. Consider $\Phi = FG(a \Rightarrow aUb)$ with the following system:

There are no α, q with $\Sigma \models (\alpha \wedge GFq) \Rightarrow \neg\Phi$, but any path of Σ that infinitely often visits $\gamma := q_1q_2$ violates Φ .

We therefore consider counterexamples of the form $\alpha \wedge GF\gamma$, $\alpha \in path_{fin}(\Sigma)$, $\gamma \in Q^*$. Below we prove that such a counterexample always exists when Φ has probability less than one.



3.2 Presenting a Qualitative Counterexample

According to the discussion in the preceding section, we propose to represent a qualitative counterexample as a pair (α, γ) , where α is a finite path of the system and γ is a finite path fragment within some bsc of the system.

Definition 3.6. A finite path fragment belonging to a bsc of a system Σ is called a recurrent word (of Σ). Let α be a finite path and γ a recurrent word of Σ . We say that γ refutes a property Y in context α iff the following conditions hold:

1. If $\gamma \neq \lambda$, then α leads to the bsc of γ , i.e., the bsc of γ is the unique bsc reachable after α .
2. $\alpha \uparrow \cap Sat(GF\gamma) \cap Y = \emptyset$, i.e., any path starting with α and repeating γ infinitely often violates Y .

If γ refutes Y in context α , then the pair (α, γ) represents the set of paths $\alpha \uparrow \cap Sat(GF\gamma)$ violating Y ; α describes how the violations begin and γ restricts their behaviour in the long run. The pair (α, γ) represents a qualitative counterexample because $\alpha \uparrow \cap Sat(GF\gamma)$ has nonzero probability, as we will see in Section 3.3. In particular, almost all paths that extend α violate Y . In this sense, α is a ‘bad’ prefix of the system. The word γ witnesses that α is ‘bad’ in this sense.

We propose to use this representation of a qualitative counterexample in an interaction between the user and the model checker as follows. First the model checker outputs α and γ and claims that γ is a recurrent word refuting Φ in context α . Then the user can challenge that claim in the following ways:

1. If $\gamma = \lambda$, the user tries to construct a path that extends α and satisfies Φ . In failing to do so, she will find a point where the actual and the expected behaviour deviate.
- 2.1. She challenges that $\gamma \neq \lambda$ belongs to any bsc at all or that after α only that bsc is reachable by constructing a path $\alpha\beta$ after which, in her opinion, γ is unreachable. The model checker refutes this challenge by returning δ such that $\alpha\beta\delta\gamma \in \text{path}_{\text{fin}}(\Sigma)$.
- 2.2. She challenges $\alpha\uparrow \cap \text{Sat}(GF\gamma) \cap \text{Sat}(\Phi) = \emptyset$, where $\gamma \neq \lambda$, by constructing a path $x = \alpha\beta_1\gamma\beta_2\gamma\dots$, which she believes to satisfy Φ . In failing to construct such a path, she will observe that the expected and the actual behaviour of the system differ.

The path x can be constructed interactively: The model checker starts with α . The user wants to extend α to a path that ultimately satisfies Φ , but she may only append a finite word at a time, allowing the model checker to append γ in between. If the user appends a word that allows the model checker to append γ directly, the model checker will do so. Otherwise the model checker suggests some extension of the current finite path that allows it to append γ afterwards. This interaction continues until the user has found some unexpected behaviour of the system.

In practice, the user cannot play forever. But she can try to generate a periodic path, i.e., a path of the form $\alpha\beta_1(\gamma\beta_2)^\omega$. It is well known that an LTL formula is violated only if it has a periodic counterexample.

3.3 Soundness and Completeness

Let $\Sigma = (Q, S, \rightarrow, v), \mathbb{P}$ be a Markov chain and Y a property. In this section, we show that our proposal to present qualitative counterexamples is sound and complete, i.e., the existence of (α, γ) implies $\mathbb{P}[Y] < 1$ and vice versa. In fact, using results from [20], we can show that our proposal is sound for arbitrary properties and complete if the specification is ω -regular.

Theorem 3.7.

1. *If γ is a recurrent word refuting Y in context α , then $\mathbb{P}[Y \mid \alpha\uparrow] = 0$ and $\mathbb{P}[Y] < 1$.*
2. *Suppose Y is ω -regular. If $\mathbb{P}[Y] < 1$, then there is an $\alpha \in \text{path}_{\text{fin}}(\Sigma)$ such that $\mathbb{P}[Y \mid \alpha\uparrow] = 0$. Moreover, if $\alpha \in \text{path}_{\text{fin}}(\Sigma)$ with $\mathbb{P}[Y \mid \alpha\uparrow] = 0$ and after α only one bsc is reachable, there is a recurrent word γ refuting Y in context α .*

The assumption in 2 that Y is ω -regular cannot be dropped. Take the Markov chain with two states q, p , both being initial states. From any state, the next state is q with probability $1/3$ and p with probability $2/3$. On the one hand, it can be shown by the Borel-Cantelli Lemma that the property Y , i.e., that “at infinitely many positions, the number of previous p ’s equals the number of previous q ’s”, has probability zero. On the other hand, there is no recurrent word γ refuting Y in some context α : a path in $\alpha\uparrow \cap \text{Sat}(GF\gamma) \cap Y$ can be constructed by extending α , visiting γ infinitely often and, between the γ ’s, making the number of previous p ’s equal the number of previous q ’s. A similar example shows that the theorem rests on the assumption that Q is finite.

We conclude this section by comparing our notion of recurrent word γ in a context α with the periodic paths $\tilde{\alpha}(\tilde{\gamma})^\omega$ used as counterexamples in classical model checking. The

pair (α, γ) describes the set of all infinite paths extending α and executing γ infinitely often, which has nonzero probability. The periodic path $\tilde{\alpha}(\tilde{\gamma})^\omega$ in general has probability zero. (The probability is nonzero only if $\tilde{\gamma}$ belongs to a “ring-like” bsc.) Even the set of all periodic paths has probability zero in general, because it is a countable set. In the probabilistic setting, counterexamples must have nonzero probability; therefore periodic paths are unsuitable as counterexamples.

4 Quantitative Counterexamples

In this section, we discuss quantitative statements. In the following, let $\Sigma = (Q, S, \rightarrow, \nu)$, \mathbb{P} be a Markov chain, Φ an LTL formula and Y a property. The corresponding question for a counterexample (or a witness) can take one of the following four shapes:

1. Why is $\mathbb{P}[\Phi] \leq t?$ ($t < 1$)
2. Why is $\mathbb{P}[\Phi] \geq t?$ ($t > 0$)
3. Why is $\mathbb{P}[\Phi] < t?$ ($t > 0$)
4. Why is $\mathbb{P}[\Phi] > t?$ ($t < 1$)

Questions 2 and 4 can be reduced to Questions 1 and 3, respectively, by negating the specification. Usually, quantitative probabilistic model checkers compute the probability of the specification. Hence, we know $\mathbb{P}[\Phi]$ before computing a counterexample, and can therefore reduce Question 3 to Question 1 by considering a bound between t and $\mathbb{P}[\Phi]$. We therefore restrict our attention to Question 1.

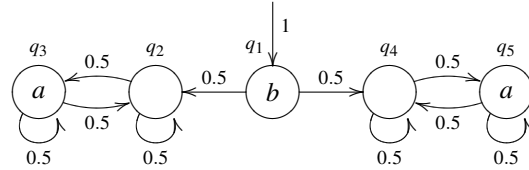
4.1 Presenting a Quantitative Counterexample

In some cases, a qualitative counterexample can be used as a quantitative counterexample. However, this is not always possible:

Example 4.1. Consider the Markov chain Σ, \mathbb{P} below together with $\Phi = FG a$.

Note that $\mathbb{P}[\Phi] = 0$. There is a recurrent word $\gamma = q_2$ refuting Φ in context $\alpha = q_1 q_2$. What does it tell us about the probability of Φ ? As $\mathbb{P}[\Phi | \alpha \uparrow] = 0$, we have $\mathbb{P}[\Phi] \leq 1 - \mathbb{P}[\alpha \uparrow] = 0.5$. However, this does not answer the question of why is $\mathbb{P}[\Phi] \leq 0$.

The problem is that the pair (α, γ) only provides information about one bsc, namely the left one, but a proof for $\mathbb{P}[\Phi] \leq 0$ must involve both bscs. To overcome this problem, we will consider counterexamples with several recurrent words, so that different bscs can be taken into account.



Definition 4.2. A recurrent set (of Σ) is a set of recurrent words of Σ . Given a recurrent set R , a word $x \in Q^\omega$ is R -fair (for Σ) iff $x \in \text{path}_\omega(\Sigma)$ and for each $\gamma \in R$ either (i) $x \models GF\gamma$ or (ii) some prefix of x cannot be extended to a finite path of Σ with suffix γ . The set of R -fair paths is denoted as $\text{Fair}_\Sigma(R)$.

Lemma 4.3. Let R be a recurrent set. Then $\mathbb{P}[\text{Fair}_\Sigma(R)] = 1$.

Proof. Let $\gamma \in R$. It can be checked that $\text{Fair}_\Sigma(\{\gamma\})$ is a fairness property according to [20, 22]. Moreover, $\text{Fair}_\Sigma(\{\gamma\})$ is ω -regular. Varacca and Völzer [20] have shown that

any ω -regular fairness property has probability one. The assertion then follows from the facts that R is countable and $Fair_\Sigma(R) = \bigcap_{\gamma \in R} Fair_\Sigma(\{\gamma\})$. \square

In the above example, consider $\alpha = q_1$ and the recurrent set $R = \{q_2, q_4\}$. Note that every R -fair run x violates the specification. Because of Lemma 4.3, $\mathbb{P}[\alpha \uparrow \cap Fair_\Sigma(R)] = \mathbb{P}[\alpha \uparrow] = 1$, and thus we have $\mathbb{P}[\Phi] \leq 1 - \mathbb{P}[\alpha \uparrow] = 0$. Together, α and R describe a set of paths violating Φ having probability 1. The prefix α describes how the violations begin. The recurrent set R describes what happens infinitely often in a violating path.

In the preceding example, R contains exactly one recurrent word for each bsc of the system, but in general it is possible that R contains no recurrent word or several recurrent words for some bsc. Consider, for instance, the specification $\Phi = GFb$; again $\mathbb{P}[\Phi] = 0$. A counterexample would be $\alpha = \lambda$ and $R = \{q_2\}$. In this case there are two kinds of R -fair paths: (i) paths going to the left bsc and visiting q_2 infinitely often; (ii) paths going to the right bsc, where q_2 can no longer be reached. As all R -fair paths violate Φ and $\mathbb{P}[Fair_\Sigma(R)] = 1$, we have $\mathbb{P}[\Phi] = 0$.

We now formalise this intuition.

Definition 4.4.

A recurrent set R refutes Y in context $\alpha \in path_{fm}(\Sigma)$ iff $\alpha \uparrow \cap Fair_\Sigma(R) \cap Y = \emptyset$.

Equivalently, R refutes Y in context α if every path of the system that extends α and is R -fair violates Y . In that case, Y is violated with at least the probability of $\alpha \uparrow$.

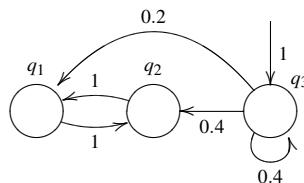
Corollary 4.5. *If there exists a recurrent set refuting Y in context α , then $\mathbb{P}[Y \mid \alpha \uparrow] = 0$, and therefore $\mathbb{P}[Y] \leq 1 - \mathbb{P}[\alpha \uparrow]$.*

It may also be necessary to consider several contexts:

Example 4.6. Consider the Markov chain Σ, \mathbb{P} below and $\Phi = q_3 U q_2$.

To show that $\mathbb{P}[\Phi] \leq 0.7$, one context word α is not enough. For instance, any recurrent set refutes Φ in context q_3q_1 , but $\mathbb{P}[q_3q_1 \uparrow] = 0.2$. This counterexample only shows that $\mathbb{P}[\Phi] \leq 0.8$.

To gather enough weight, we need to use several contexts. For instance let $\alpha_1 = q_3q_1$, $\alpha_2 = q_3q_3q_1$ and $\alpha_3 = q_3q_3q_3q_1$. Clearly \emptyset refutes Φ in context α_i . Then $\mathbb{P}[\Phi] \leq 1 - \mathbb{P}[\bigcup_i \alpha_i \uparrow]$. As the three sets are disjoint, $\mathbb{P}[\bigcup_i \alpha_i \uparrow] = 0.2 + 0.08 + 0.032 > 0.3$.

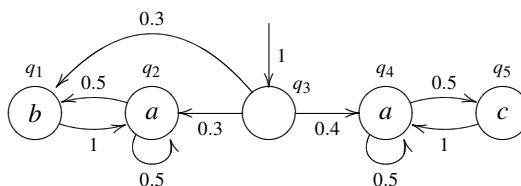


In this simple example, the recurrent sets do not matter. Different contexts in principle require different recurrent sets:

Example 4.7. Consider the Markov chain Σ, \mathbb{P} below and $\Phi = G \neg c \wedge (GFb \Rightarrow Xa)$.

Let $\alpha_1 = q_3q_4$, $\alpha_2 = q_3q_1$ and $R_1 = \{q_5\}$, $R_2 = \{q_1\}$. Note that R_i refutes Φ in context α_i . First, any R_1 -fair path extending q_3q_4 violates Φ , as it visits q_5 . Second, any R_2 -fair path extending q_3q_1 violates Φ , as it visits q_1 infinitely often,

but its second state does not satisfy a . Hence, $\mathbb{P}[\Phi] \leq 1 - \mathbb{P}[\alpha_1 \uparrow \cup \alpha_2 \uparrow] = 0.3$.



This example also shows that whether a path almost certainly satisfies Φ depends not only on which bscs it visits; here, satisfaction also depends on the second state of the path. Therefore, in the case of general LTL properties, the bscs cannot simply be partitioned into “accepting” and “rejecting”.

If a recurrent set refutes a property in a context, a larger recurrent set will also do so. We can therefore suppose without loss of generality that all R_i are the same. In the above example, we can choose $R = R_1 \cup R_2$; then R refutes Φ in context α_i , $i = 1, 2$. Taking only one recurrent set is a design decision simplifying the theory. In practice, it might be desirable to have several recurrent sets.

Definition 4.8. *Let W be a set of finite paths of Σ . A recurrent set R refutes Y in context W iff $W \uparrow \cap \text{Fair}_\Sigma(R) \cap Y = \emptyset$.*

Corollary 4.9. *If there exists a recurrent set refuting Y in context W , then $\mathbb{P}[Y \mid W \uparrow] = 0$, and therefore $\mathbb{P}[Y] \leq 1 - \mathbb{P}[W \uparrow]$.*

Thus, we present a quantitative counterexample explaining why $\mathbb{P}[\Phi] \leq t$ by the sets W and R such that R is a recurrent set refuting Φ in context W and $\mathbb{P}[W \uparrow] \geq 1 - t$.

4.2 Completeness

Corollary 4.9 is a soundness result: if there is a recurrent set refuting Y in context W , then the property is violated with probability at least $\mathbb{P}[W \uparrow]$. It turns out that Definition 4.8 also gives us a complete representation of a counterexample: if a property is violated with some probability, there is a pair (W, R) witnessing it. In fact, there is a canonical set that can always be used as the context W .

Definition 4.10. *Let $I(\Sigma, Y)$ be the set of all $\alpha \in \text{path}_{\text{fm}}(\Sigma)$ such that there is a recurrent set refuting Y in context α . We call $I(\Sigma, Y)$ the initial language (of Σ w.r.t. Y).*

Note that $I(\Sigma, Y)$ by itself is a context, i.e., there is a recurrent set refuting Y in context $I(\Sigma, Y)$. To verify that, let R_α be the recurrent set refuting Y in context α , where $\alpha \in I(\Sigma, Y)$. Then $R := \bigcup_{\alpha \in I(\Sigma, Y)} R_\alpha$ refutes Y in context $I(\Sigma, Y)$.

Theorem 4.11. *For any LTL formula Φ , $I(\Sigma, \Phi)$ is regular.*

In Section 5.2, we will explain how to compute a finite automaton accepting $I(\Sigma, \Phi)$.

The next proposition states important properties of the initial language. Firstly, almost all elements of $I(\Sigma, Y) \uparrow$ are violations of Y . Moreover, if the property is given by an LTL formula Φ , almost all violations of Φ belong to $I(\Sigma, \Phi) \uparrow$. Hence, the probabilities of $\neg\Phi$ and $I(\Sigma, \Phi) \uparrow$ coincide.

Proposition 4.12.

1. $\mathbb{P}[I(\Sigma, Y) \uparrow \cap Y] = 0$.
2. For any LTL formula Φ , $\mathbb{P}[I(\Sigma, \Phi) \uparrow \cup \text{Sat}(\Phi)] = 1$.
3. For any LTL formula Φ , $\mathbb{P}[I(\Sigma, \Phi) \uparrow] = \mathbb{P}[\neg\Phi]$.

We can now give some equivalent characterisations of the initial language. The first statement of Proposition 4.13 asserts that the initial language is the largest context in which a recurrent set refuting Y exists. The second statement provides an alternative definition of the initial language in terms of \mathbb{P} .

Proposition 4.13.

1. The initial language $I(\Sigma, Y)$ is the largest set $W \subseteq \text{path}_{\text{fin}}(\Sigma)$ such that there is a recurrent set refuting Y in context W .
2. For any LTL formula Φ , $I(\Sigma, \Phi)$ is the set of all $\alpha \in \text{path}_{\text{fin}}(\Sigma)$ so that $\mathbb{P}[\Phi \mid \alpha \uparrow] = 0$.

Finally we prove completeness.

Theorem 4.14. *Let Φ be an LTL formula, $0 \leq t < 1$ and $\mathbb{P}[\Phi] \leq t$. Then there is a nonempty set $W \subseteq \text{path}_{\text{fin}}(\Sigma)$ such that $\mathbb{P}[\Phi \mid W \uparrow] = 0$ and $\mathbb{P}[W \uparrow] \geq 1 - t$. Moreover, for any $W \subseteq \text{path}_{\text{fin}}(\Sigma)$, $W \neq \emptyset$ with $\mathbb{P}[\Phi \mid W \uparrow] = 0$ there is a recurrent set R refuting Φ in context W .*

We will see in Section 5.3 that the set R can be chosen to contain exactly one recurrent word per bscc. If the bound t is tight, i.e., $t = \mathbb{P}[\Phi]$, the context W is in general infinite. If $t > \mathbb{P}[\Phi]$, one can show – using standard results of measure theory – that it is always possible to choose W as a finite subset of $I(\Sigma, \Phi)$.

4.3 Interaction with the Model Checker

In this section, we discuss the interaction between user and model checker for quantitative counterexamples. The model checker computes $\mathbb{P}[\Phi]$ and presents $W \subseteq I(\Sigma, \Phi)$ such that $\mathbb{P}[W \uparrow] \geq t$, where t is given by the user. The user then inspects W , and may identify some $\alpha \in W$ for which she does not believe that $\mathbb{P}[\Phi \mid \alpha \uparrow] = 0$. To convince the user, the model checker computes a recurrent set R refuting Φ in context W , which contains at most one element for each bscc of the system (see Section 5.3). The interaction between user and model checker that follows is similar to the qualitative case. The user can challenge the following:

1. R is a recurrent set: each element $\gamma \in R$ can be checked as in the qualitative case.
2. R refutes Φ in context α : similar as in the qualitative case, the user interactively tries to construct a path in $\alpha \uparrow \cap \text{Fair}_{\Sigma}(R) \cap \Phi$ and fails. Note that the model checker can assure fairness while the user can concentrate on constructing a path that ultimately satisfies Φ . Once a bscc has been reached, the model checker can also output the $\gamma \in R$ associated with that bscc.

The set W may be too large or even infinite so that inspecting each element individually is not feasible (see [5, 9]). This raises the question of how the user can understand what words are contained in W . Also, the reader may want evidence that indeed $\mathbb{P}[W \uparrow] \geq t$. Similar questions arise in the study of counterexamples for probabilistic CTL ([14]) model checking, and we refer to the literature for possible approaches [4, 5, 9, 23]. We also discuss these issues further in Section 6.

5 Computing Counterexamples

In this section, we explain how the counterexamples defined above can be computed. Our algorithm is based on, but substantially complements an algorithm of Courcoubetis and Yannakakis [8]¹. We follow [18] in our presentation. In Section 5.1 we briefly

¹ Note that we refer to the optimal algorithm in Section 3.1 of [8], and not to the automata-based algorithm in Section 4.1, which is non-optimal for LTL.

recall the underlying model checking algorithm. In Sections 5.2 and 5.3 we address the computation of an automaton accepting the initial language and the computation of a recurrent set, respectively.

Throughout the entire section, $\Sigma = (Q, S, \rightarrow, \nu), \mathbb{P}$ is a Markov chain and Φ an LTL formula. Without loss of generality, we assume that Φ only contains the temporal connectives X and U.

It is well known that the assertion $\mathbb{P}[\Phi] = 1$ is independent from the underlying Markov measure \mathbb{P} (see e.g. [20]). (It depends only on which transition probabilities are nonzero, which is uniquely determined by \rightarrow .) We therefore simply say that a formula Φ is *large (in Σ)* iff $\mathbb{P}[\Phi] = 1$.

5.1 Recalling Courcoubetis and Yannakakis

The algorithm presented in [8] works in steps. At each step, it eliminates one temporal operator from the specification and at the same time refines the system so that the largeness of the specification is preserved. After eliminating all operators, the specification becomes a state formula ϕ , for which largeness can easily be checked: ϕ is large iff all initial states satisfy ϕ . We now briefly recall how the transformation takes place.

If Φ is not a state formula, then it has a subformula of the form $\Theta = \psi U \xi$ or $\Theta = X \xi$, where ψ, ξ are state formulas. The algorithm chooses such a formula Θ and replaces it by a fresh atomic proposition d . We call the resulting formula Φ' .

The algorithm then partitions the set of states Q into three blocks Q_Θ^L , Q_Θ^S and Q_Θ^M . If the initial states of Σ are replaced by a state in Q_Θ^L , Θ becomes large. If the initial states of Σ are replaced by a state in Q_Θ^S , $\neg\Theta$ becomes large (Θ becomes “small”). If the initial states of Σ are replaced by a state in Q_Θ^M , neither Θ nor $\neg\Theta$ becomes large (Θ becomes “medium-sized”).

The new system $\Sigma' = (Q', S', \rightarrow', \nu')$ has the set of states

$$Q' := Q_\Theta^L \times \{\Theta\} \cup Q_\Theta^S \times \{\neg\Theta\} \cup Q_\Theta^M \times \{\Theta, \neg\Theta\},$$

that is, the states in Q_Θ^L are annotated with Θ , the states in Q_Θ^S with $\neg\Theta$, and the states in Q_Θ^M are split into a copy with Θ and one with $\neg\Theta$. We denote the first projection as π so that, for instance, $\pi(q, \Theta) = q$. We extend π to words in the natural way. The initial states of the new system are the states that are projected to an initial state of the original system. The new valuation function ν' is just like ν , whereas d holds in the states annotated with Θ and only there. Finally, the transition relation of Σ' is defined so that Φ' is large in Σ' iff Φ is large in Σ (see [8, 18]).

A single transformation step takes time $O(|\Sigma||\Phi|)$. Moreover, the size of Σ' is at most twice the size of Σ ; hence, it can be shown that the overall complexity of the algorithm is $O(|\Sigma|2^{|\Phi|})$.

5.2 Computing the Initial Language

In this section, we explain how to compute a deterministic finite automaton (DFA) accepting $I(\Sigma, \Phi)$. The algorithm from 5.1 terminates after n transformation steps on Σ and Φ , resulting in the system Σ_n and state formula Φ_n . The n -fold projection on states and paths of Σ_n is denoted π^n , that is, π^n maps a state (path) of Σ_n to the corresponding state (path) of Σ . The following lemma shows how $I(\Sigma, \Phi)$ can be expressed in terms of $Sat(\Sigma_n, \Phi_n)$:

Lemma 5.1. *We have $I(\Sigma, \Phi) = \text{path}_{\text{fin}}(\Sigma) \setminus \pi^n(\text{Sat}(\Sigma_n, \Phi_n)) \downarrow$.*

The elements of $\pi^n(\text{Sat}(\Sigma_n, \Phi_n)) \downarrow$ are (modulo π^n) the finite paths of Σ_n starting in a state satisfying Φ_n . It is therefore straightforward to compute a non-deterministic finite automaton (NFA) accepting $\pi^n(\text{Sat}(\Sigma_n, \Phi_n)) \downarrow$. It is also straightforward to compute a deterministic finite automaton (DFA) accepting $\text{path}_{\text{fin}}(\Sigma)$. By applying standard automata constructions, we obtain a DFA for $I(\Sigma, \Phi)$.

In Theorem 5.2, we provide the key points of our complexity analysis.

Theorem 5.2.

1. *An NFA accepting $\pi^n(\text{Sat}(\Sigma_n, \Phi_n)) \downarrow$ can be computed in time linear in $|\Sigma|$ and exponential in $|\Phi|$.*
2. *A DFA accepting $\pi^n(\text{Sat}(\Sigma_n, \Phi_n)) \downarrow$ can be computed in time linear in $|\Sigma|$ and doubly exponential in $|\Phi|$.*
3. *A DFA accepting $I(\Sigma, \Phi)$ can be computed in time linear in $|\Sigma|$ and doubly exponential in $|\Phi|$.*

The overall running time is linear in $|\Sigma|$ and doubly exponential in $|\Phi|$, and we do not know whether an exponential algorithm can be found. In Section 5.3 we explain how to compute a single element of $I(\Sigma, \Phi)$ without computing the entire DFA; the running time of the latter approach is linear in $|\Sigma|$ and exponential in $|\Phi|$.

5.3 Computing a Recurrent Set

In this subsection, Σ' and Φ' denote the system and formula after one transformation step has been applied to Σ and Φ . Moreover, Θ is the subformula of Φ that has been replaced by the new atomic proposition d during the transformation.

We explain how to compute a recurrent set R refuting Φ in context $I(\Sigma, \Phi)$ and therefore in any context $W \subseteq I(\Sigma, \Phi)$. For each bsc K of Σ , our algorithm calls a function *computeRecurrentWord* to compute a path fragment $\gamma_K \in K^+$ such that $I(\Sigma, \Phi) \uparrow \cap \text{Sat}(GF\gamma_K) \cap \text{Sat}(\Phi) = \emptyset$. The result R is then defined as $R := \{\gamma_K \mid K \text{ bsc of } \Sigma\}$. Note that $\text{Fair}_\Sigma(R) = \bigcup_K \text{Sat}(GF\gamma_K)$. Hence, $I(\Sigma, \Phi) \uparrow \cap \text{Fair}_\Sigma(R) \cap \text{Sat}(\Phi) = \emptyset$, i.e., R refutes Φ in context $I(\Sigma, \Phi)$.

The function *computeRecurrentWord* is outlined in Figure 1. Correctness can be shown by induction over Φ .

Lemma 5.3.

*The function *computeRecurrentWord* terminates and establishes its postconditions.*

We now explain how Lines 9-11 of Figure 1 can be implemented. Suppose $\Theta = \psi \cup \xi$. Given γ' from Line 8, choose δ' minimal w.r.t. \sqsubseteq such that the following holds:

1. $\gamma'\delta'$ is a finite path fragment of Σ' .
2. $\pi(\gamma'\delta')$ does not end in Q_Θ^M .
3. If $\pi(\gamma'\delta')$ visits a state satisfying d , then $\pi(\gamma'\delta')$ visits a state satisfying ξ .

Set $\gamma := \pi(\gamma'\delta')$.

It can be shown that from each state in Q_Θ^M both a state in Q_Θ^L satisfying ξ and a state in Q_Θ^S is reachable. An examination of the state relation of Σ' then yields that a δ' satisfying the above conditions exists and can therefore be computed by a breadth-first search.

Fig. 1. Function: $\gamma = \text{computeRecurrentWord}(\Sigma, \Phi, q)$

```

1 Precondition:  $\Sigma$  is a system,  $\Phi$  a formula,  $q$  a state of  $\Sigma$ .
2 Postcondition:
  (1)  $\gamma$  is a finite path fragment of  $\Sigma$  with first state  $q$ .
      (In particular, if  $q$  belongs to the bsc  $K$ , then  $\gamma \in K^+$ .)
  (2)  $I(\Sigma, \Phi) \uparrow \cap \text{Sat}(\text{GF}\gamma) \cap \text{Sat}(\Phi) = \emptyset$ .
3 begin
4   if  $\Phi$  is a state formula then
5      $\gamma := q$ ;
6   else
7     choose a state  $q'$  of  $\Sigma'$  with  $\pi(q') = q$ ;
8      $\gamma' := \text{computeRecurrentWord}(\Sigma', \Phi', q')$ ;
9     choose a finite path fragment  $\gamma$  of  $\Sigma$  such that
10    (1) for each path fragment  $\tilde{\gamma}'$  of  $\Sigma'$ , if  $\gamma = \pi(\tilde{\gamma}')$ , then  $\gamma' \sqsubseteq \tilde{\gamma}'$ ,
11    (2) for each  $x' \in \text{path}_{\omega}(\Sigma')$ , if  $\pi(x') \models \text{GF}\gamma$ , then  $x' \models \text{G}(\Theta \Leftrightarrow d)$ .
12  end
13 end

```

Now suppose $\Theta = X\xi$. Given γ' from Line 8, we construct γ as follows. If $\pi(\gamma')$ does not end in Q_{Θ}^M , we set $\gamma := \pi(\gamma')$. Otherwise, we extend γ' by one state q' to $\gamma'q' \in \text{path}_{\text{fin}}(\Sigma')$ and set $\gamma := \pi(\gamma'q')$.

A proof that γ satisfies the conditions in Lines 9-11 can be found in [19]. The running time of *computeRecurrentWord* is as follows:

Theorem 5.4. *Executing $\text{computeRecurrentWord}(\Sigma, \Phi, q)$ takes time $O(|\Sigma||\Phi|2^{|\Phi|})$.*

Proof. Let n be the number of transformation steps and Σ_i , $1 \leq i \leq n$, the system after the i th transformation step. The length of $\gamma = \text{computeRecurrentWord}(\Sigma, \Phi, q)$ is bounded by $O(\sum_{i=1}^n |\Sigma_i|)$, because the i th incarnation of *computeRecurrentWord* increases γ by at most $|\Sigma_i|$, $1 \leq i \leq n$. As $|\Sigma_i| \leq |\Sigma|2^i$, the length of γ is in $O(|\Sigma|2^{|\Phi|})$.

Computing γ from γ' includes reading γ' and computing some extension; both can be accomplished in time $O(|\Sigma|2^{|\Phi|})$. This has to be repeated n times; hence the overall running time is $O(|\Sigma||\Phi|2^{|\Phi|})$. \square

The function *computeRecurrentWord* has to be executed once for each bsc of Σ ; accordingly the overall running time is linear in the number of bsccs and $|\Sigma|$ and exponential in $|\Phi|$.

Note that the user does not need to compute the entire recurrent set at once. Instead, after computing one recurrent word, she can already inspect the bsc of the recurrent word. If she then wants to find an error in a different bsc, she can compute a recurrent word of that bsc. Thus, although the worst-case running time is quadratic in the size of the system, the user already obtains the first diagnostic feedback after $O(|\Sigma||\Phi|2^{|\Phi|})$ steps.

The function *computeRecurrentWord* can be adapted to compute a single element α of $I(\Sigma, \Phi)$, whereas the complexity remains the same. The details can be found in [19].

Theorem 5.5.

If $I(\Sigma, \Phi) \neq \emptyset$, a single element of $I(\Sigma, \Phi)$ can be computed in $O(|\Sigma||\Phi|2^{|\Phi|})$ steps.

Theorems 5.5 and 5.4 mean that a representation of a qualitative counterexample can be computed in time linear in the system and exponential in the specification. This running time is optimal, because it is also the running time of the optimal probabilistic model checking algorithm in [8].

6 Conclusions and Related Work

We have proposed a way of presenting and computing counterexamples in probabilistic LTL model checking for Markov chains. Our notion is sound and complete, which means that a counterexample in our sense can be computed if and only if the specification is not met with the desired probability. We have also pointed out how such a counterexample can be utilised to find an error in the system.

Aljazzar and Leue [2] propose solutions for counterexamples in probabilistic model checking with respect to timed probabilistic reachability properties in Markov chains. Han and Katoen [12] and Wimmer *et al.* [23] present algorithms computing counterexamples for model checking PCTL (probabilistic CTL [14]) formulas in Markov chains. There are also suggestions of how to present such counterexamples to the user [4, 9]. In [1, 13], the problem has been tackled for continuous time Markov chains. In [3] Aljazzar and Leue generalise their proposal in [2] for (unnested, upwards-bounded) PCTL formulas and Markov decision processes.

Recently, Andrés *et al.* [5] proposed an approach for LTL formulas on Markov chains (and also Markov decision processes). They refer to the fact that probabilistic model checking of an LTL formula in a Markov chain M_1 can be reduced to probabilistic model checking of an upwards-bounded reachability property in a generated Markov chain M_2 , which is doubly exponentially larger than M_1 in the size of the LTL formula [10]. Then they develop a counterexample representation in the style of Han and Katoen [12], which can be mapped to a subset of the initial language in M_1 . The authors propose an interesting way of convincing the user that the upwards-bounded reachability property is indeed violated in the generated Markov chain M_2 . However, in contrast to our approach, they do not address how to convince the user of the probability of the original LTL formula in the original system M_1 .

The above approaches [2, 4, 5, 9, 12, 23] have in common that a counterexample is *finitary*, i.e., a set of finite paths W so that any path of the system extending W violates the specification. In our terminology, W is a subset of the initial language. We have pointed out in Section 3.1 that sets of finite paths are not sufficient to refute general LTL properties – in particular liveness properties. Even so, the techniques of presenting finitary counterexamples to the user can be applied to what we have called a context W in our counterexample presentation. In future work, it would be interesting to combine these techniques with our approach. Another important direction is to carry out some case studies to evaluate the interaction between user and model checker.

Acknowledgement: We thank Husain Aljazzar, Christian Dax, Barbara Jobstmann and Felix Klaedtke for useful discussions and helpful comments. We also thank the reviewers for suggesting improvements to the paper.

References

1. H. Aljazzar, H. Hermanns and S. Leue. Counterexamples for timed probabilistic reachability. In *FORMATS 2005*, volume 3829 of *LNCS*, pages 177–195. Springer.
2. H. Aljazzar and S. Leue. Extended directed search for probabilistic timed reachability. In *FORMATS 2006*, volume 4202 of *LNCS*, pages 33–51. Springer.
3. H. Aljazzar and S. Leue. Counterexamples for model checking of Markov decision processes. Tech. Report soft-08-01, University of Konstanz, Germany, 2007.
4. H. Aljazzar and S. Leue. Debugging of dependability models using interactive visualization of counterexamples. In *QEST 2008*, pages 189–198. IEEE.
5. M. Andrés, P. D’Argenio and P. van Rossum. Significant diagnostic counterexamples in probabilistic model checking. In *Haifa Verification Conference 2008*, *LNCS*, to appear.
6. L. Breiman. *Probability*. Addison Wesley, 1968.
7. T. H. Cormen, C. E. Leiserson and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 2001.
8. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, 1995.
9. B. Damman, T. Han and J.-P. Katoen. Regular expressions for PCTL counterexamples. In *QEST 2008*, pages 179–188. IEEE.
10. L. de Alfaro. Temporal logics for the specification of performance and reliability. In *STACS 1997*, volume 1200 of *LNCS*, pages 165–176. Springer.
11. E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, volume B, chapter 16, pages 995–1072. Elsevier Science, 1990.
12. T. Han and J.-P. Katoen. Counterexamples in probabilistic model checking. In *TACAS 2007*, volume 4424 of *LNCS*, pages 72–86. Springer.
13. T. Han and J.-P. Katoen. Providing evidence of likely being on time: Counterexample generation for CTMC model checking. In *ATVA 2007*, volume 4762 of *LNCS*, pages 331–346.
14. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.*, 6(5):512–535, 1994.
15. J. G. Kemeny, J. L. Snell and A. W. Knapp. *Denumerable Markov Chains*. Springer, 1976.
16. A. Pnueli. The temporal logic of programs. In *FOCS 1977*, pages 46–57. IEEE.
17. K. Ravi, R. Bloem and F. Somenzi. A comparative study of symbolic algorithms for the computation of fair cycles. In *FMCAD 2000*, volume 1954 of *LNCS*, pages 143–160. Springer.
18. M. Schmalz. Extensions of an algorithm for generalised fair model checking. Diploma Thesis, Lübeck, Germany, 2007, www.infsec.ethz.ch/people/mschmalz/dt.pdf.
19. M. Schmalz, H. Völzer, and D. Varacca. Counterexamples in probabilistic LTL model checking for Markov chains. Technical Report 627, ETH Zürich, Switzerland, www.inf.ethz.ch/research/disstechreps/techreports, 2009.
20. D. Varacca and H. Völzer. Temporal logics and model checking for fairly correct systems. In *LICS 2006*, pages 389–398. IEEE.
21. M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *FOCS 1985*, pages 327–338. IEEE.
22. H. Völzer, D. Varacca and E. Kindler. Defining fairness. In *CONCUR 2005*, volume 3653 of *LNCS*, pages 458–472. Springer.
23. R. Wimmer, B. Braitling and B. Becker. Counterexample generation for discrete-time Markov chains using bounded model checking. In *VMCAI 2009*, volume 5403 of *LNCS*, pages 366–380. Springer.