Counterexamples in Probabilistic LTL Model Checking for Markov Chains

Matthias Schmalz[1] Daniele Varacca[2] Hagen Völzer[3]

[1]ETH Zurich, Switzerland
[2]PPS - CNRS & Univ. Paris Diderot, France
[3]IBM Zurich Research Laboratory, Switzerland

Abstract. We propose how to present and compute 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 presented counterexample helps the user to identify the underlying error in the system through 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 to identify and repair an error in the system. However a counterexample path is in general infinite, and if we want to show it to the user, we must find a finite representation. In the case of classical LTL model checking, we can exploit the fact that a *periodic* counterexample always exists (see e.g. [18]), 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 $\mathbb{P}[Y] \geq 1-t$ (resp. $\mathbb{P}[Y] > 1-t$). 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 we 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.

June 9, 2009This 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, while 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.

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. We provide references for further reading.

2.1 Words

Let Q be a set of *states*. The set of infinite, finite, nonempty finite words over Q is denoted Q^ω, Q^*, 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) $\Sigma = (Q, S, \to, v)$ consists of a finite set Q of *states*, a nonempty set $S \subseteq Q$ of *initial states*, a *state relation* $\to \subseteq Q \times Q$ and a *valuation function* $v : Q \to 2^{AP}$ 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 $q \to p$. The *size* of Σ is $|\Sigma| := |Q| + |\to|$.

A *path fragment (of Σ)* is a word $q_0 q_1 \ldots \in Q^\omega \cup Q^*$ such that $q_{i-1} \to q_i$, $i > 0$. 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 $(Q, \to)$. A set $K \subseteq Q$ is a *strongly connected component (of Σ)* (scc for short) if it is a strongly connected component of $(Q, \to)$ (see e.g. [7]). A *bottom strongly connected component (of Σ)* (bscc) is an scc K with no outgoing edges, i.e., if $q \in K$ and $q \to p$, then $p \in K$.

A *(labelled discrete-time) Markov chain* (see e.g. [6, 16]) is a system $\Sigma = (Q, S, \to, v)$ equipped with *transition probabilities* given by $P : Q \times Q \to [0,1]$ and *initial probabilities* given by $P_{ini} : Q \to [0,1]$, where $P(q,p) > 0$ iff $q \to p$, $P_{ini}(q) > 0$ 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, 16]) 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 $\Sigma, \mathbb{P}$.

2.3 Temporal Properties

A *(linear-time temporal) property*, denoted Y, Z, is a subset of Q^ω. We mainly consider properties expressible in linear temporal logic (LTL) [17]; 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 $\mathrm{X}, \mathrm{U}, \mathrm{G}, \mathrm{F}$. 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 $\Sigma = (Q, S, \to, v)$ over words $x \in Q^\omega$. For $i \in \mathbb{N}$, $x, i \vDash \Phi$ means that *x satisfies* Φ *at position i* (in the usual sense [11]). Moreover, $x \vDash \Phi$ (*"x satisfies* Φ*"*) abbreviates $x, 0 \vDash \Phi$; $\Sigma \vDash \Phi$ (*"*Σ *satisfies* Φ*"*) means $x \vDash \Phi$ for all $x \in path_\omega(\Sigma)$. We write $Sat(\Sigma, \Phi)$ for the set of all infinite paths of Σ satisfying Φ. For convenience, we often write $Sat(\Phi)$ or Φ instead of $Sat(\Sigma, \Phi)$. In particular, $Sat(true) = path_\omega(\Sigma)$. A formula ϕ without temporal connectives is a *state formula*. For $q \in Q$, $q \vDash \phi$ (*"q satisfies* ϕ*"*) iff $qx \vDash \phi$ 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 $q_0 \wedge \mathrm{X}(q_1 \wedge \dots \mathrm{X}(q_n) \dots)$ and λ instead of *true*. These assumptions do not have an impact on 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 a CTL* formula of the form $A.\Phi$ 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.

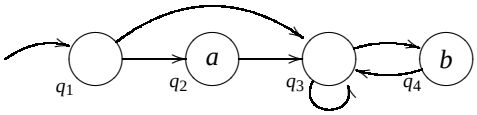
The situation is different for existential quantification. Say we want to understand why a CTL* formula $E.\Phi$ is violated, again Φ being an LTL formula. For example, $E.\Phi$ could express the property that there exists a run of the system such that 'someone wins the jackpot'. If the formula is false, 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. She should have an idea on what the path looks like; in the example: she knows *how* someone could win the jackpot in the system. She can then try to replay that path. Since the desired path does not exist in the system, the user will find in this way a point where the behaviour of the system deviates from her expectations.

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, we consider now a few examples. By default, the examples are based on the system $\Sigma = (Q, S, \to, \nu)$ below. For the qualitative case the particular transition probabilities of the Markov chain are not relevant (see e.g. [20]); so we do not display them.
Each of the examples considers an LTL formula Φ for which $\mathbb{P}[\Phi] < 1$, and in each case we will discuss what a counterexample should look like.



Example 3.1. Let $\Phi = \mathrm{G}\,\neg a$. Since 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 $\alpha = q_1 q_2$. Since, $\mathbb{P}[q_1 q_2 \uparrow] > 0$, 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 $\Phi = \mathrm{F}\,a$. There is a finite path $\alpha := q_1 q_3$ in the system such that each extension of α into $path_\omega(\Sigma)$[1] 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 was executed. Trying to play such a path, which does not exist, she will eventually find the point in the system where actual and expected behaviour deviate.

1. [MS says: changed]

Example 3.3. Let $\Phi = \mathrm{F}\,\mathrm{G}\,\neg b$. We recall that, in any Markov chain, a path eventually enters a bscc with probability one. For each (reachable) bscc K, a path eventually enters K with nonzero probability, and then, with probability 1, it 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) bscc that contains a b-state, and therefore the specification is violated with nonzero probability.

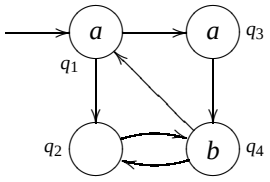
To show that to the user, we propose that the model checker returns a b-state within a bscc, namely q_4. The user then convinces herself that (i) the b-state indeed belongs to a (reachable) bscc and (ii) repeatedly visiting the b-state violates Φ. The latter point (ii) is straightforward in this case. To convince herself of (i), the user plays the following interactive game with the model checker: She tries to find a finite path β 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 $\Phi = \mathrm{G}\,\mathrm{F}\,b \Rightarrow \mathrm{F}\,a$. Repeatedly visiting a b-state without visiting an a-state violates Φ. The specification does not have probability 1, since there is a bscc containing a b-state but no a-state, and that bscc can be reached without passing through an a-state. We propose that the model checker outputs q_4 and $\alpha := q_1 q_3$. The user then convinces herself that (i) q_4 belongs to a bscc, and that α leads to that bscc, 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 to discover the error, she tries to refute (ii) by extending α to $\alpha\beta \in \mathit{path}_{\mathit{fin}}(\Sigma)$ so that β visits an a-state.

In Example 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), $\alpha \in \mathit{path}_{\mathit{fin}}(\Sigma)$, $q \in Q$ such that $\Sigma \models \alpha \wedge \mathrm{G}\,\mathrm{F}\,q \Rightarrow \neg\Phi$. 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 bscc.

There are however still situations, where counterexamples of the form (α, q) cannot be found, and instead of the single state q we need a path fragment:

Example 3.5. Consider $\Phi = \mathrm{F}\,\mathrm{G}(a \Rightarrow a\,\mathrm{U}\,b)$ with the following system:
There are no α, q with $\Sigma \models (\alpha \wedge \mathrm{G}\,\mathrm{F}\,q) \Rightarrow \neg\Phi$. But any path of Σ that visits $\gamma := q_1 q_2$ infinitely often violates Φ.



We therefore consider counterexamples of the form $\alpha \wedge \mathrm{G}\,\mathrm{F}\,\gamma$, $\alpha \in \mathit{path}_{\mathit{fin}}(\Sigma)$, $\gamma \in Q^*$. 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 previous 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 bscc of the system.

Definition 3.6. *A finite path fragment belonging to a bscc 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 the context α *just when:*

1. *if $\gamma \neq \lambda$, then α leads to the bscc of γ, that is, the bscc of γ is the unique bscc reachable after α,*
2. *$\alpha\uparrow \cap \mathit{Sat}(\mathrm{G}\,\mathrm{F}\,\gamma) \cap Y = \varnothing$, i.e., any path starting with α and repeating γ infinitely often violates Y.*

If γ refutes Y in the context α, then the pair (α, γ) represents the set of paths $\alpha\uparrow \cap \mathit{Sat}(\mathrm{G}\,\mathrm{F}\,\gamma)$ 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 $\alpha\uparrow \cap \mathit{Sat}(\mathrm{G}\,\mathrm{F}\,\gamma)$ 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 the context α. Then the user can challenge that claim in the following ways:

1. If $\gamma = \lambda$, then the user tries to construct a path that extends α and satisfies Φ. In failing to do so, she will find a point where actual and expected behaviour deviate.

2.1. She challenges that $\gamma \neq \lambda$ belongs to any bscc at all or that after α only that bscc 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 $\alpha\beta\delta\gamma \in \mathit{path}_{\mathit{fin}}(\Sigma)$.

2.2. She challenges $\alpha\uparrow \cap \mathit{Sat}(\mathrm{G}\,\mathrm{F}\,\gamma) \cap \mathit{Sat}(\Phi) = \varnothing$, where $\gamma \neq \lambda$, by constructing a path $x = \alpha\beta_1\gamma\beta_2\gamma\ldots$, which she believes to satisfy Φ. In failing to construct such a path, she will observe that expected and 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 does so. Otherwise the model checker suggests some extension of the current finite path that allows it to append γ afterwards. This interaction goes on 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., 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 $\Sigma = (Q, S, \to, v), \mathbb{P}$ 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 the 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 $\alpha \in \mathit{path}_{\mathit{fin}}(\Sigma)$ such that $\mathbb{P}[Y \mid \alpha\uparrow] = 0$. Moreover, if $\alpha \in \mathit{path}_{\mathit{fin}}(\Sigma)$ with $\mathbb{P}[Y \mid \alpha\uparrow] = 0$ and after α only one bscc is reachable, there is a recurrent word γ refuting Y in the 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 one hand, it can be shown by the Borel-Cantelli Lemma that the property Y "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 $\mathit{Sat}(\mathrm{G}\,\mathrm{F}\,\gamma) \cap \alpha\uparrow \cap Y$ 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.[2] A similar example shows that the theorem rests on the assumption that the system is finite.

2. [MS says: changed]

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 $\tilde{\alpha}(\tilde{\gamma})^\omega$ has in general probability zero. (The probability is nonzero only if $\tilde{\gamma}$ belongs to a "ring-like" bscc.)[3] Even the set of all periodic paths has in general probability zero, because it is a countable set. In the probabilistic setting, counterexamples must have nonzero probability; therefore periodic paths are unsuitable as counterexamples.

3. [MS says: changed]

4 Quantitative Counterexamples

In this section, we discuss quantitative statements. In the following, let $\Sigma = (Q, S, \to, v)$ be a system, $\mathbb{P}$ a Markov measure, 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 $\mathbb{P}[\Phi] > t$? $(t < 1)$

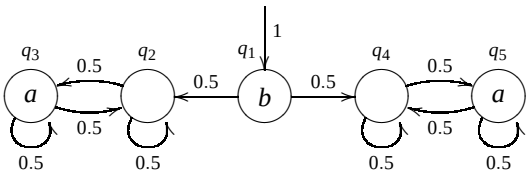
Questions 2 and 4 can be reduced to Questions 1 and 3, respectively, by negating the specification. Usually, quantitative probabilistic model checkers compute the probability of the specification.[4] Hence, we know $\mathbb{P}[\Phi]$ before computing a counterexample, and can therefore reduce Question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

4. [MS says: changed]

4.1 Presenting a Quantitative Counterexample

In some cases, a qualitative counterexample can be used as a quantitative counterexample; that is, however, not always possible:

Example 4.1. Consider the Markov chain $\Sigma, \mathbb{P}$ below together with $\Phi = \mathrm{F}\,\mathrm{G}\,a$. Note that $\mathbb{P}[\Phi] = 0$. There is a recurrent word $\gamma = q_2$ refuting Φ in the context $\alpha = q_1 q_2$. What does it tell us about the probability of Φ? Since $\mathbb{P}[\Phi \mid \alpha\uparrow] = 0$, 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 gives information about one bscc, namely the left one, but a proof for $\mathbb{P}[\Phi] \leq 0$ must involve both bsccs. To overcome this problem, we will consider counterexamples with several recurrent words, so that different bsccs 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 $x \in path_\omega(\Sigma)$ and for each $\gamma \in R$ either (i) $x \vDash \mathrm{G}\,\mathrm{F}\,\gamma$ 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 $Fair_\Sigma(R)$.*

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

Proof. Let $\gamma \in R$. It can be checked that $Fair_\Sigma(\{\gamma\})$ is a fairness property according to [20, 22]. Moreover, $Fair_\Sigma(\{\gamma\})$ 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 $Fair_\Sigma(R) = \bigcap_{\gamma\in R} Fair_\Sigma(\{\gamma\})$. □

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 bscc of the system, but in general it is possible that R contains no recurrent word or several recurrent words for some bscc. Consider for instance the specification $\Phi = \mathrm{G}\,\mathrm{F}\,b$; 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 bscc and visiting q_2 infinitely often; (ii) paths going to the right bscc, where q_2 can no longer be reached. Since all R-fair paths violate Φ and $\mathbb{P}[Fair_\Sigma(R)] = 1$, we have $\mathbb{P}[\Phi] = 0$.

We now formalise this intuition.

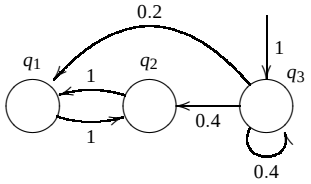
Definition 4.4. *A recurrent set* R refutes Y in the context $\alpha \in path_{fin}(\Sigma)$ *iff* $\alpha\uparrow \cap Fair_\Sigma(R) \cap Y = \varnothing$.

Equivalently, R refutes Y in the 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 the context* α*, then* $\mathbb{P}[Y \mid \alpha\uparrow] = 0$*, and therefore* $\mathbb{P}[Y] \leq 1 - \mathbb{P}[\alpha\uparrow]$.

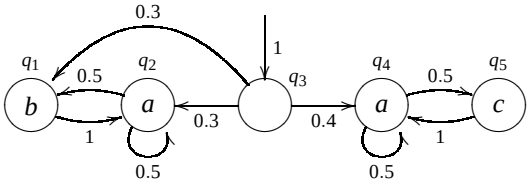
It may also be necessary to consider several contexts:

Example 4.6. Consider the Markov chain $\Sigma, \mathbb{P}$ below and $\Phi = q_3\,\mathrm{U}\,q_2$.
To show that $\mathbb{P}[\Phi] \leq 0.7$, one context word α is not enough. For instance, any recurrent set refutes Φ in the context of q_3q_1, but $\mathbb{P}[q_3q_1\uparrow] = 0.2$. This counterexample only shows that $\mathbb{P}[\Phi] \leq 0.8$.
In order to gather enough weight, we need to use several contexts. For instance let $\alpha_1 = q_3q_1$, $\alpha_2 = q_3q_3q_1$ $\alpha_3 = q_3q_3q_3q_1$. Clearly $\varnothing$ refutes Φ in the context α_i. Then $\mathbb{P}[\Phi] \leq 1 - \mathbb{P}[\cup_i \alpha_i\uparrow]$. Since the three sets are disjoint $\mathbb{P}[\cup_i \alpha_i\uparrow] = 0.2 + 0.08 + 0.032 > 0.3$.



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

Example 4.7. Consider the Markov chain $\Sigma, \mathbb{P}$ below and $\Phi = \mathrm{G}\,\neg c \wedge (\mathrm{G}\,\mathrm{F}\,b \Rightarrow \mathrm{X}\,a)$.
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 the context α_i. First, any R_1-fair path extending q_3q_4 violates Φ, since it visits q_5. Second, any R_2-fair path extending q_3q_1 violates Φ, since 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 bscc it visits; here, satisfaction also depends on the second state of the path. Therefore, in the case of general LTL properties, the bsccs cannot simply be partitioned into "accepting" and "rejecting".

If[5] a recurrent set refutes a property in a context, a larger recurrent set does so, too. We can therefore suppose without loss of generality that all the R_i are the same. In the above example we can choose $R = R_1 \cup R_2$; then R refutes Φ in the context α_i, $i = 1,2$.

Taking[6] only one recurrent set is a design decision simplifying the theory. In practice, it might be desirable to have several recurrent sets.

5. [MS says: changed]

6. [MS says: new]

Definition 4.8. *Let W be a set of finite paths of Σ. A recurrent set R refutes Y in the context W iff $W\uparrow \cap Fair_\Sigma(R) \cap Y = \varnothing$.*

Corollary 4.9. *If there exists a recurrent set refuting Y in the 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 the 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 the 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 $\alpha \in path_{fin}(\Sigma)$ such that there is a recurrent set refuting Y in the 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 the context $I(\Sigma,Y)$. To see that let R_α be the recurrent set refuting Y in the context α, where $\alpha \in I(\Sigma,Y)$. Then $R := \bigcup_{\alpha \in I(\Sigma,Y)} R_\alpha$ refutes Y in the 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,* $\mathbb{P}[I(\Sigma,\Phi)\uparrow \cup Sat(\Phi)] = 1$.
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.

Proposition 4.13.

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

The first statement 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 $\mathbb{P}$.

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* $W \subseteq path_{fin}(\Sigma)$ *such that* $\mathbb{P}[\Phi \mid W\uparrow] = 0$ *and* $\mathbb{P}[W\uparrow] \geq 1 - t$. *Moreover, for any* $W \subseteq path_{fin}(\Sigma)$, $W \neq \varnothing$ *with* $\mathbb{P}[\Phi \mid W\uparrow] = 0$ *there is a recurrent set* R *refuting* Φ *in the 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 the 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:

1. R is a recurrent set: each element $\gamma \in R$ can be checked as in the qualitative case.
2. R refutes Φ in the context α: similar as in the qualitative case, the user interactively tries to construct a path in $\alpha\uparrow \cap Fair_\Sigma(R) \cap \Phi$ 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 is reached, the model checker can also output the $\gamma \in R$ that is 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]. 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 [19] in our presentation. In Section 5.1 we recall the

[1] 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.

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, $\Sigma = (Q, S, \to, v), \mathbb{P}$ 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 whether $\mathbb{P}[\Phi] = 1$ is independent of the underlying Markov measure $\mathbb{P}$ (see e.g. [20]). (It depends only on which transition probabilities are nonzero, which is uniquely determined by $\to$.) We therefore say that a formula Φ is *large (in* Σ*)* iff $\mathbb{P}[\Phi] = 1$.

7. [MS says: changed]

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 be easily 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 $\Theta = \psi \mathrm{U} \xi$ or $\Theta = \mathrm{X}\,\xi$, 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 $Q^L_\Theta, Q^S_\Theta, Q^M_\Theta$. If the initial states of Σ are replaced by a state in Q^L_Θ, then Θ becomes large. If the initial states of Σ are replaced by a state in Q^S_Θ, then $\neg\Theta$ becomes large (Θ becomes "small"). If the initial states of Σ are replaced by a state in Q^M_Θ, then neither Θ nor $\neg\Theta$ becomes large (Θ becomes "medium-sized").

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

$$Q' := Q^L_\Theta \times \{\Theta\} \quad \cup \quad Q^S_\Theta \times \{\neg\Theta\} \quad \cup \quad Q^M_\Theta \times \{\Theta, \neg\Theta\},$$

that is, the states in Q^L_Θ are annotated with Θ, the states in Q^S_Θ with $\neg\Theta$, and the states in Q^M_Θ are split into a copy with Θ and a copy 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 v' is just like v, 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, 19]).

A single transformation step takes time $O(|\Sigma||\Phi|)$. Moreover, the size of Σ' is at most the double of 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 by $Sat(\Sigma_n, \Phi_n)$:

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

The elements of $\pi^n(Sat(\Sigma_n,\Phi_n))\downarrow$ 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 $\pi^n(Sat(\Sigma_n,\Phi_n))\downarrow$. It is also straightforward to compute a deterministic finite automaton (DFA) accepting $path_{fin}(\Sigma)$. By applying standard automata constructions, we obtain a DFA for $I(\Sigma,\Phi)$.

In Theorem 5.2 we give the keypoints of our complexity analysis.

Theorem 5.2.

1. *An NFA accepting* $\pi^n(Sat(\Sigma_n,\Phi_n))\downarrow$ *can be computed in time linear in* $|\Sigma|$ *and exponential in* $|\Phi|$.
2. *A DFA accepting* $\pi^n(Sat(\Sigma_n,\Phi_n))\downarrow$ *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 whole 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 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 the context $I(\Sigma,\Phi)$ and therefore in any context $W \subseteq I(\Sigma,\Phi)$. For each bscc K of Σ, our algorithm calls the function *computeRecurrentWord* to compute a path fragment $\gamma_K \in K^+$ such that $I(\Sigma,\Phi)\uparrow \cap Sat(\mathrm{G\,F}\,\gamma_K) \cap Sat(\Phi) = \varnothing$. The result R is then defined as $R := \{\gamma_K \mid K \text{ bscc of } \Sigma\}$. Note that $Fair_\Sigma(R) = \bigcup_K Sat(\mathrm{G\,F}\,\gamma_K)$. Hence, $I(\Sigma,\Phi)\uparrow \cap Fair_\Sigma(R) \cap Sat(\Phi) = \varnothing$, i.e., R refutes Φ in the 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 post-conditions.*

We now explain how Lines 9-11 can be implemented. Suppose $\Theta = \psi\,\mathrm{U}\,\xi$. Given γ' from Line 8, choose δ' minimal w.r.t. $\sqsubseteq$ such that the following conditions hold:

1. $\gamma'\delta'$ is a finite path fragment of Σ'.
2. $\pi(\gamma'\delta')$ does not end in Q^M_Θ.
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^M_Θ both a state in Q^L_Θ satisfying ξ and a state in Q^S_Θ is reachable. An examination of the state relation of Σ' then yields that a δ'

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

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

satisfying the above conditions exists and can therefore be computed by a breadth-first search.

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

We prove in the appendix that γ satisfies the conditions in Lines 9-11. The running time of *computeRecurrentWord* is as follows:

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

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

The function *computeRecurrentWord* has to be executed once for each bscc of Σ; hence 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 bscc of the recurrent word. If she then wants to find an error in a different bscc, she can compute a recurrent word of that bscc. Hence, 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 adopted to compute a single element α of $I(\Sigma,\Phi)$, whereas the complexity remains the same. The details can be found in the appendix.

Theorem 5.5.
If $I(\Sigma,\Phi) \neq \varnothing$, a single element of $I(\Sigma,\Phi)$ can be computed in $O(|\Sigma||\Phi|2^{|\Phi|})$ steps.

Theorem 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, since it is also the running time of the optimal probabilistic model checking algorithm in [8].

6 Conclusion

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] propose 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.[8] 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.

8. [MS says: inserted]

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, Chair for Software Engineering, 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. E. Hopcroft and J. D. Ullman. *Introduction to Automat Theory, Languages and Computation*. Addison-Wesley, 1979.
16. J. G. Kemeny, J. L. Snell and A. W. Knapp. *Denumerable Markov Chains*. Springer, 1976.
17. A. Pnueli. The temporal logic of programs. In *FOCS 1977*, pages 46–57. IEEE.
18. 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.
19. M. Schmalz. Extensions of an algorithm for generalised fair model checking. Diploma Thesis, Lübeck, Germany, 2007, `www.infsec.ethz.ch/people/mschmalz/dt.pdf`.
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.

A Proofs

We have reordered the sections so that all proofs only rely on statements that have been previously proved.

Proofs in Section 3.3

Theorem 3.7

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

Proof. We prove 1. As a special case of Lemma 4.3, we have $\mathbb{P}[\mathrm{G}\,\mathrm{F}\,\gamma \mid \alpha\uparrow] = 1$. Hence, $\mathbb{P}[Y \mid \alpha\uparrow] = \mathbb{P}[Y \mid \alpha \wedge \mathrm{G}\,\mathrm{F}\,\gamma] = 0$ and therefore $\mathbb{P}[Y] \leq \mathbb{P}[\alpha\uparrow^c] < 1$.

We prove 2. Using the results from [20] it can be shown that, if $\mathbb{P}[Y] < 1$, then there is a recurrent word γ refuting Y in some context α. According to 1, $\mathbb{P}[Y \mid \alpha\uparrow] = 0$. Moreover by [20], if $\mathbb{P}[Y \mid \alpha\uparrow] = 0$ for some $\alpha \in path_{fin}(\Sigma)$ after which only one bscc is reachable, then there is a recurrent word γ refuting Y in the context α. □

A More Detailed Description of Courcoubetis and Yannakakis

Let $\Sigma = (Q, S, \to, v)$ be a system and Φ a formula. In this section we consider the case that Φ is not a state formula. In that case Φ has a subformula of the form $\Theta = \psi\,\mathrm{U}\,\xi$ or $\Theta = \mathrm{X}\,\xi$, 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 $Q^L_\Theta, Q^S_\Theta, Q^M_\Theta$. If the initial states of Σ are replaced by a state in Q^L_Θ, then Θ becomes large. If the initial states of Σ are replaced by a state in Q^S_Θ, then $\neg\Theta$ becomes large (Θ becomes "small"). If the initial states of Σ are replaced by a state in Q^M_Θ, then neither Θ nor $\neg\Theta$ becomes large (Θ becomes "medium-sized").

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

$$Q' \;:=\; Q^L_\Theta \times \{\Theta\} \quad\cup\quad Q^S_\Theta \times \{\neg\Theta\} \quad\cup\quad Q^M_\Theta \times \{\Theta, \neg\Theta\},$$

that is, the states in Q^L_Θ are annotated with Θ, the states in Q^S_Θ with $\neg\Theta$, and the states in Q^M_Θ are split into a copy with Θ and a copy 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 project to an initial state of the original system. The new valuation function v' is just like v, whereas d holds precisely in the states annotated with Θ. Formally, $v'(q, \Theta) := v(q) \cup \{d, a_{(q,\Theta)}\}$, and $v'(q, \neg\Theta) := v(q) \cup \{a_{(q,\neg\Theta)}\}$. The atomic propositions $a_{q'}$, $q' \in Q'$ are fresh and pairwise different (cf. Section 2.3). In the implementation, the atomic propositions $a_{q'}$ can also be dropped.

If $\Theta = \psi\,\mathrm{U}\,\xi$, then $\to'$ is the smallest relation satisfying conditions 1 and 2 below. Let $q', p' \in Q'$ with $\pi(q') = q$, $\pi(p') = p$ and $p' = (p, \Xi)$.

1. If $q \to p$ and $q \notin Q^M_\Theta$, then $q' \to' p'$.
2. If $q \to p$ and $q \in Q^M_\Theta$, then $(q, \Xi) \to' (p, \Xi)$.

If $\Theta = \mathrm{X}\,\xi$, then $\rightarrow'$ is the smallest relation satisfying conditions 1 and 2 below. Let $q \in Q$ and $p' \in Q'$ with $\pi(p') = p$.

1. If $q \rightarrow p$ and $p \models \xi$, then $(q,\Theta) \rightarrow' p'$.
2. If $q \rightarrow p$ and $p \not\models \xi$, then $(q,\neg\Theta) \rightarrow' p'$.

Courcoubetis and Yannakakis have proved that Φ' is large in Σ' iff Φ is large in Σ. For an alternative correctness proof see [19].

A single transformation step takes time $O(|\Sigma||\Phi|)$. Since the size of Σ' is at most the double of the size of Σ, it can be shown that the overall complexity of the algorithm is $O(|\Sigma|2^{|\Phi|})$. (As usual, this complexity analysis is under the uniform cost criterion.)

The following lemma, already observed by Courcoubetis and Yannakakis, illustrates the relation between the path fragments of Σ and Σ'.

Lemma A.1. *For the systems Σ, Σ' as defined above, the following statements hold:*

1. *If z' is a path (fragment) of Σ', then $\pi(z')$ is a path (fragment) of Σ.*
2. *For each path (fragment) z of Σ, there is a path (fragment) z' of Σ' such that $\pi(z') = z$.*
3. *If the last states of the nonempty finite path fragments α'_1, α'_2 of Σ coincide and $\pi(\alpha'_1) = \pi(\alpha'_2)$, then $\alpha'_1 = \alpha'_2$.*

Proof. Statement 1 is an immediate consequence of the definition of $\rightarrow'$ and S'. Statement 2 and 3 of the assertion follow from these facts: If $s \in S$, then there exists a $\Xi_1 \in \{\Theta, \neg\Theta\}$ such that $(s,\Xi_1) \in S'$. For each $q \in Q$ and each $(p,\Xi_2) \in Q'$ with $q \rightarrow p$, there is a unique $\Xi_1 \in \{\Theta, \neg\Theta\}$ such that $(q,\Xi_1) \rightarrow' (p,\Xi_2)$.

Proofs in Section 5.2

Lemma A.2. *Let $\Sigma, \mathbb{P}$ be a Markov chain, Y an ω-regular property and $\alpha \in path_{fin}(\Sigma)$. If $\mathbb{P}[Y \mid \alpha\uparrow] = 0$, then there is a recurrent set R refuting Y in the context α.*

Proof. We cannot apply Theorem 3.7 directly to α, since after α several bsccs might be reachable. But, for any $\beta \in path_{fin}(\Sigma)$ extending α such that after β only one bscc is reachable, let γ_β be a recurrent word refuting Y in the context β. Let R be the set of all such γ_β. Then R refutes Y in the context α. □

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

Proof. Let α be a finite path of Σ. Then

$$
\begin{array}{ll}
 & \alpha \in I(\Sigma,\Phi) \\
\text{iff} & \text{there is a recurrent set } R \text{ with } \alpha\uparrow \cap Fair_\Sigma(R) \cap Sat(\Sigma,\Phi) = \varnothing \\
\text{iff} & \neg\alpha \vee \neg\Phi \text{ is large in } \Sigma \\
\text{iff} & \neg\alpha \vee \neg\Phi_n \text{ is large in } \Sigma_n \\
\text{iff} & \Sigma_n \models \neg\alpha \vee \neg\Phi_n \\
\text{iff} & \forall x' :\ x' \in Sat(\Sigma_n,\Phi_n) \Rightarrow x' \not\models \alpha \\
\text{iff} & \forall x,x' :\ x' \in Sat(\Sigma_n,\Phi_n) \wedge x = \pi^n(x') \Rightarrow \alpha \not\sqsubseteq x \\
\text{iff} & \forall x :\ (\exists x' :\ x' \in Sat(\Sigma_n,\Phi_n) \wedge x = \pi^n(x')) \Rightarrow \alpha \not\sqsubseteq x \\
\text{iff} & \neg\exists x :\ x \in \pi^n(Sat(\Sigma_n,\Phi_n)) \wedge \alpha \sqsubseteq x \\
\text{iff} & \alpha \notin \pi^n(Sat(\Sigma_n,\Phi_n))\downarrow .
\end{array}
$$

The second line follows with the definition of $I(\Sigma,\Phi)$. The third line can be derived from Lemma 4.3 and Lemma A.2. The fourth line holds because the transformations of Courcoubetis and Yannakakis preserve largeness. The fifth line follows because $Sat(\Sigma_n, \neg\alpha \vee \neg\Phi_n)$ is a safety property. The remaining lines are derived by set-theoretical and first order reasoning. □

Finite Automata: before we prove Theorem 5.2, we fix the notation for finite automata. A *nondeterministic finite automaton (NFA) A* (see e.g. [15]) is a tuple (Q,Γ,ρ,q_0,F) consisting of a *set of states* Q, an *input alphabet* Γ, a *transition function* $\rho : Q\times\Gamma \to 2^Q$, an *initial state* q_0 and a set $F \subseteq Q$ of *accepting states*. As usual, *A accepts* a word $w_1 \dots w_n$ over Γ if there is a word of states $q_0 \dots q_n$ such that $q_i \in \rho(q_{i-1}, w_i)$, $1 \leq i \leq n$ and $q_n \in F$. If $|\rho(q,a)| \leq 1$ for each $q \in Q$ and $a \in \Gamma$, the automaton is a *deterministic finite automaton (DFA)*.

Theorem 5.2

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

Proof. We prove 1. Suppose that $\Sigma_n =: (Q_n, S_n, \to_n, v_n)$. We define
$B := (Q_n \cup \{q_0\}, Q, \rho, q_0, Q_n \cup \{q_0\})$ with

- $q_0 \notin Q_n$,
- $\rho(q_0,q) = \{q' \in S_n \mid \pi^n(q') = q \wedge q' \vDash \Phi_n\}, \qquad (q \in Q)$,
- $\rho(p',q) = \{q' \in Q_n \mid \pi^n(q') = q \wedge p' \to_n q'\}, \qquad (p' \in Q_n,\ q \in Q)$.

It is straightforward to check that B can be computed in time linear in $|\Sigma|$ and exponential in $|\Phi|$.

We prove 2. First observe the following properties of B. Any state of B different from the initial state is of the form (q, ann), where q is a state of Σ and ann a sequence of n (possibly negated) subformulas of Φ, which have been introduced by the transformations of Courcoubetis and Yannakakis. (For simplicity, we take the convention that $(a,(b,c)) = ((a,b),c)$).) If B is in the state q' and reads the symbol $p \in Q$, then all the possible successor states are of the form (p, ann), i.e., they have the first component p.

To obtain a DFA B_{det} accepting $\pi^n(Sat(\Sigma_n,\Phi_n))\downarrow$, we apply the well-known subset construction to B. Because of the special structure of B, all reachable states of B_{det} (besides the initial state) are of the form $\{(q,ann_1),(q,ann_2),(q,ann_2),\dots\}$, where q is a state of Σ and ann_i is a sequence of n (possibly negated) subformulas introduced by the algorithm of Courcoubetis and Yannakakis. In other words, the elements of a reachable state of B_{det} (not being the initial state) coincide in their first component. If the first component is q, we call such a state a q-state. The number of reachable states of B is linear in $|Q|$ and doubly exponential in n.

If B_{det} is in a q-state and reads p, then the successor state is nonempty only if Σ has the transition $q \to p$ (cf. Lemma A.1). Hence, the number of transitions of B_{det} is linear in $|\Sigma|$ and doubly exponential in $|\Phi|$. The automaton B_{det} can therefore be computed by a depth-first search in time linear in $|\Sigma|$ and doubly exponential in $|\Phi|$.

We prove 3. We compute a DFA B^c accepting $Q^* \setminus \pi^n(Sat(\Sigma_n, \Phi_n))\downarrow$ from B_{det} as usual by adding an error state and complementing the set of accepting states. It is straightforward to determine an automaton A_Σ accepting $path_{fin}(\Sigma)$. According to Lemma 5.1, the product automaton A of B^c and A_Σ accepts $I(\Sigma,\Phi)$. It can be shown with similar arguments as in the proof of 2 that A can be computed in time linear in $|\Sigma|$ and doubly exponential in $|\Phi|$. □

Proofs in Section 4.2

Proposition 4.12

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

Proof. Let R be a recurrent set refuting Y in the context $I(\Sigma,Y)$. That means, $I(\Sigma,Y)\uparrow \cap Fair_\Sigma(R) \cap Y = \varnothing$. Assertion 1 follows with Lemma 4.3.

For a proof of 2, suppose $\mathbb{P}[I(\Sigma,\Phi)\uparrow \cup Sat(\Phi)] < 1$. Note that $I(\Sigma,\Phi)\uparrow \cup Sat(\Phi)$ is ω-regular (cf. Theorem 5.2). With Theorem 3.7 choose a recurrent word γ refuting $I(\Sigma,\Phi)\uparrow \cup Sat(\Phi)$ in some context α. Since after α only the bscc of γ is reachable,

$$\alpha\uparrow \cap Fair_\Sigma(\{\gamma\}) \cap (I(\Sigma,\Phi)\uparrow \cup Sat(\Phi)) = \varnothing.$$

Hence, $\alpha \in I(\Sigma,\Phi)$. Then $\varnothing = \alpha\uparrow \cap Fair_\Sigma(\{\gamma\}) \cap I(\Sigma,\Phi)\uparrow = \alpha\uparrow \cap Fair_\Sigma(\{\gamma\})$ – a contradiction.

Assertion 3 can be derived from 1 and 2:

$$\begin{aligned}\mathbb{P}[I(\Sigma,\Phi)\uparrow] &= \mathbb{P}[I(\Sigma,\Phi)\uparrow \cap Sat(\Phi)] + \\ &\quad \mathbb{P}[I(\Sigma,\Phi)\uparrow \cap Sat(\neg\Phi)] + \\ &\quad \mathbb{P}[I(\Sigma,\Phi)\uparrow^c \cap Sat(\neg\Phi)] \\ &= \mathbb{P}[Sat(\neg\Phi)]\,.\end{aligned}$$

□

Proposition 4.13

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

Proof. We prove 1. We have already seen that there is a recurrent set refuting Y in the context $I(\Sigma,Y)$. Suppose some recurrent set R_W refutes Y in some context $W \subseteq path_{fin}(\Sigma)$. For any $\alpha \in W$, R_W refutes Y in the context α, and therefore $\alpha \in I(\Sigma,Y)$. We conclude that $W \subseteq I(\Sigma,Y)$.

We prove 2. Because of Corollary 4.5 and Lemma A.2, $\mathbb{P}[\Phi \mid \alpha\uparrow] = 0$ if and only if there is a recurrent set refuting Φ in the context α. Therefore $I(\Sigma,\Phi)$ is the set of all $\alpha \in path_{fin}(\Sigma)$ such that $\mathbb{P}[\Phi \mid \alpha\uparrow] = 0$. □

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 $W \subseteq path_{fin}(\Sigma)$ such that $\mathbb{P}[\Phi \mid W\uparrow] = 0$ and $\mathbb{P}[W\uparrow] \geq 1-t$. Moreover, for any $W \subseteq path_{fin}(\Sigma)$, $W \neq \varnothing$ with $\mathbb{P}[\Phi \mid W\uparrow] = 0$ there is a recurrent set R refuting Φ in the context W.*

Proof. First take $W := I(\Sigma, \Phi)$. Because of Proposition 4.12, $\mathbb{P}[\Phi \mid W\uparrow] = 0$. Moreover, since $\mathbb{P}[\neg\Phi] = \mathbb{P}[W\uparrow]$, $\mathbb{P}[W\uparrow] = 1 - \mathbb{P}[\Phi] \geq 1-t$.

Second suppose $W \subseteq path_{fin}(\Sigma)$, $W \neq \varnothing$ with $\mathbb{P}[\Phi \mid W\uparrow] = 0$. Because of Assertion 2 in Proposition 4.13, $W \subseteq I(\Sigma, \Phi)$. Therefore the recurrent set R refuting Φ in the context $I(\Sigma, \Phi)$ also refutes Φ in the context W. □

Proofs in Section 5.3

Lemma A.3. *Suppose $x \in path_\omega(\Sigma)$ and $x' \in path_\omega(\Sigma')$ with $\pi(x') = x$. If $x \in I(\Sigma, \Phi)\uparrow$, then $x' \in I(\Sigma', \Phi')\uparrow$.*

Proof. Let $x \in path_\omega(\Sigma)$, $x' \in path_\omega(\Sigma')$ with $\pi(x') = x$ and $x \in I(\Sigma, \Phi)\uparrow$. Choose $\alpha \in I(\Sigma, \Phi)$ so that $\alpha \sqsubseteq x$. Let α' be the prefix of x' of the same length as α. We will show that $\alpha' \in I(\Sigma', \Phi')$.

Note that $\alpha \Rightarrow \neg\Phi$ is large in Σ. Hence, $\alpha \Rightarrow \neg\Phi'$ is large in Σ'. As $\Sigma' \vDash \alpha' \Rightarrow \alpha$, $\alpha' \Rightarrow \neg\Phi'$ is large in Σ'. Therefore, $\alpha' \in I(\Sigma', \Phi')$. □

Lemma 5.3 *The function computeRecurrentWord terminates and establishes its post-conditions.*

Proof. First note that *computeRecurrentWord* terminates; a termination argument is the number of temporal connectives in Φ.

We prove the postconditions by induction over Φ. Suppose Φ is a state formula. Postcondition (1) obviously holds. We know there is some recurrent set R such that $I(\Sigma, \Phi)\uparrow \cap Fair_\Sigma(R) \cap Sat(\Phi) = \varnothing$. Since Φ is a state formula, $I(\Sigma, \Phi)\uparrow \cap Sat(\Phi) = \varnothing$. Hence, $I(\Sigma, \Phi)\uparrow \cap Sat(\mathrm{G\,F}\,\gamma) \cap Sat(\Phi) = \varnothing$.

Now suppose Φ is not a state formula, and the recursive call in Line 8 establishes its postconditions. We prove that γ satisfies the postconditions. Postcondition (1) obviously holds. For Postcondition (2), let $x \in I(\Sigma, \Phi)\uparrow \cap Sat(\mathrm{G\,F}\,\gamma)$. With Lemma A.1 choose $x' \in path_\omega(\Sigma')$ such that $\pi(x') = x$. By Line 10, $x' \vDash \mathrm{G\,F}\,\gamma'$, and, by Lemma A.3, $x' \in I(\Sigma', \Phi')$. By the induction hypothesis, $x' \nvDash \Phi'$. By Line 11, $x' \vDash \mathrm{G}(\Theta \Leftrightarrow d)$, and therefore $x' \nvDash \Phi$. Since d does not appear in Φ, $x \nvDash \Phi$. □

Lemma A.4. *Suppose $\Theta = \psi\,\mathrm{U}\,\xi$. Let γ' be a nonempty finite path fragment of Σ'. Choose δ' minimal w.r.t. $\sqsubseteq$ such that the following conditions hold:*

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

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

Then the conditions in Lines 10 and 11 of computeRecurrentWord applied to Σ, Φ, q hold.

Proof. We first address the condition in Line 10. Choose a path fragment $\tilde{\gamma}'$ of Σ' such that $\pi(\tilde{\gamma}') = \gamma$. We have to show that $\gamma' \sqsubseteq \tilde{\gamma}'$. Since γ does not end in Q_Θ^M, the last states of $\tilde{\gamma}'$ and $\gamma'\delta'$ coincide. By Lemma A.1, $\gamma'\delta' = \tilde{\gamma}'$, and therefore $\gamma' \sqsubseteq \tilde{\gamma}'$.

For the condition in Line 11, it can be observed that any path $x' \in path_\omega(\Sigma')$ violating $\mathrm{G}(\Theta \Leftrightarrow d)$ satisfies $\mathrm{F}\,\mathrm{G}(d \wedge \neg\xi)$, i.e., from some position on x' satisfies d and does not satisfy ξ. Now suppose that $x' \vDash \mathrm{G}\,\mathrm{F}\,\gamma$. Because of Condition 3 in the definition of γ, x' satisfies $\mathrm{G}(\Theta \Leftrightarrow d)$. □

Lemma A.5. *Suppose $\Theta = \mathrm{X}\,\xi$. Let γ' be a finite path fragment of Σ'. If $\pi(\gamma')$ does not end in Q_Θ^M, we set $\gamma := \pi(\gamma')$. Otherwise, we extend γ' by one state q' to $\gamma'q' \in path_{fin}(\Sigma')$ and set $\gamma := \pi(\gamma'q')$.*

Then the conditions in Lines 10 and 11 of computeRecurrentWord applied to Σ, Φ, q hold.

Proof. We first consider the condition in Line 10. Suppose $\pi(\gamma')$ does not end in Q_Θ^M. Take $\tilde{\gamma}'$ with $\pi(\tilde{\gamma}') = \gamma$. Then $\pi(\gamma') = \pi(\tilde{\gamma}')$ and γ', $\tilde{\gamma}'$ end in the same state. Hence, by Lemma A.1, $\gamma' = \tilde{\gamma}'$. Now suppose $\pi(\gamma')$ ends in Q_Θ^M. Let q' be a state of Σ' such that $\gamma = \pi(\gamma'q')$ and $\gamma'q'$ is a path fragment of Σ'. Take $\tilde{\gamma}'\tilde{q}'$ with $\pi(\tilde{\gamma}'\tilde{q}') = \gamma$. We have to show that $\gamma' \sqsubseteq \tilde{\gamma}'\tilde{q}'$. The states q' and $\tilde{q}'$ are not necessarily the same, but there projections are. By the construction of Σ', the last states of γ' and $\tilde{\gamma}'$ coincide. Therefore, by Lemma A.1, $\gamma' = \tilde{\gamma}' \sqsubseteq \tilde{\gamma}'\tilde{q}'$.

The condition in Line 11 holds, because any $x' \in path_\omega(\Sigma')$ satisfies $\mathrm{G}(\Theta \Leftrightarrow d)$. □

The function *computeContext*, an adopted version of *computeRecurrentWord* that computes a single element of $I(\Sigma, \Phi)$, is outlined in Figure 2. The statement in Line 9 can be implemented as in *computeRecurrentWord*.

Fig. 2. Function: $\alpha = computeContext(\Sigma, \Phi)$

```
 1 Precondition: Σ is a system, Φ a formula.
 2 Postcondition: α ∈ I(Σ, Φ).
 3 begin
 4     if Φ is a state formula then
 5         choose α as an initial state of Σ violating Φ;
 6     else
 7         α' := computeContext(Σ', Φ');
 8         choose a finite path α of Σ such that
 9         for each x' ∈ path_ω(Σ'), if α ⊑ π(x'), then α' ⊑ x';
10     end
11 end
```

Lemma A.6. *The function computeContext terminates and establishes its postcondition.*

Proof. Note that *computeContext* terminates because the number of temporal connectives in the formula decreases with each recursive call.

If Φ is a state formula, the postcondition is obviously established. Suppose that Φ is no state formula and $\alpha' \in I(\Sigma', \Phi')$. Let $\alpha \in path_{fin}(\Sigma)$ such that Line 9 holds. We show that $\alpha \in I(\Sigma, \Phi)$.

Let R be the recurrent set computed by help of *computeRecurrentWord* refuting Φ in the context $I(\Sigma, \Phi)$. Let x be an R-fair path of Σ extending α. Since for each bscc R contains a recurrent word refuting Φ belonging to that bscc, there is a recurrent word γ refuting Φ such that $x \vDash \mathrm{G}\,\mathrm{F}\,\gamma$. Choose $x' \in path_\omega(\Sigma')$ with $\pi(x') = x$. Because of Line 9, $\alpha' \sqsubseteq x'$ and therefore $x' \in I(\Sigma', \Phi')\uparrow$. Let γ' be the recurrent word of Σ' from which γ has been computed; in particular, $I(\Sigma', \Phi')\uparrow \cap Sat(\mathrm{G}\,\mathrm{F}\,\gamma') \cap Sat(\Phi') = \varnothing$. Because of Line 10 of *computeRecurrentWord*, $x' \vDash \mathrm{G}\,\mathrm{F}\,\gamma'$. Hence, $x' \nvDash \Phi'$. Because of Line 11 of *computeRecurrentWord*, $x' \vDash \mathrm{G}(\Theta \Leftrightarrow d)$ and therefore $x' \nvDash \Phi$. Since d does not appear in Φ, $x \nvDash \Phi$. As x is an arbitrary R-fair path of Σ extending α, we conclude that $\alpha\uparrow \cap Fair_\Sigma(R) \cap Sat(\Phi) = \varnothing$. Hence, $\alpha \in I(\Sigma, \Phi)$. □