Significant Diagnostic Counterexamples in Probabilistic Model Checking

Miguel E. Andrés[1] *, Pedro D'Argenio[2] ** , Peter van Rossum[1]

[1]Institute for Computing and Information Sciences, The Netherlands.
{mandres,petervr}@cs.ru.nl
[2]FaMAF, Universidad Nacional de Córdoba, CONICET, Argentina.
dargenio@famaf.unc.edu.ar

Abstract. This paper presents a novel technique for counterexample generation in probabilistic model checking of Markov chains and Markov Decision Processes. (Finite) paths in counterexamples are grouped together in witnesses that are likely to provide similar debugging information to the user. We list five properties that witnesses should satisfy in order to be useful as debugging aid: similarity, accuracy, originality, significance, and finiteness. Our witnesses contain paths that behave similarly outside strongly connected components.
Then, we show how to compute these witnesses by reducing the problem of generating counterexamples for general properties over Markov Decision Processes, in several steps, to the easy problem of generating counterexamples for reachability properties over acyclic Markov chains.

1 Introduction

Model checking is an automated technique that, given a finite-state model of a system and a property stated in an appropriate logical formalism, systematically checks the validity of this property. Model checking is a general approach and is applied in areas like hardware verification and software engineering.

Nowadays, the interaction geometry of distributed systems and network protocols calls for probabilistic, or more generally, quantitative estimates of, e.g., performance and cost measures. Randomized algorithms are increasingly utilized to achieve high performance at the cost of obtaining correct answers only with high probability. For all this, there is a wide range of models and applications in computer science requiring quantitative analysis. Probabilistic model checking allows to check whether or not a probabilistic property is satisfied in a given model, e.g., "Is every message sent successfully received with probability greater or equal than 0.99?".

A major strength of model checking is the possibility of generating diagnostic information in case the property is violated. This diagnostic information is provided through a *counterexample* showing an execution of the model that invalidates the property under verification. Besides the immediate feedback in model checking, counterexamples are also used in abstraction-refinement techniques [CGJ+00], and provide the foundations for schedule derivation (see, e.g., [BLR05]).

Although counterexample generation was studied from the very beginning in most model checking techniques, this has not been the case for probabilistic model checking. Only recently [AHL05,AD06,AL06,HK07a,HK07b,AL07] attention was drawn to this subject,fifteen years after the first studies on probabilistic model checking. Contrarily to other model checking techniques, counterexamples in this setting are *not* given by a single execution path. Instead, they are *sets of executions* of the system satisfying a certain undesired property whose probability mass is higher than a given bound.

* Supported by NWO project 612.000.526
** Supported by the ANPCyT project PICT 26135 and CONICET project PIP 6391

Since counterexamples are used as a diagnostic tool, previous works on counterexamples have presented them as sets of *finite* paths with probability large enough. We refer to these sets as *representative counterexamples*. Elements of representative counterexamples with high probability have been considered the most informative since they contribute mostly to the property refutation.

A challenge in counterexample generation for probabilistic model checking is that (1) representative counterexamples are very large (often infinite), (2) many of its elements have very low probability (which implies that are vert distant from the counterexample), and (3) that elements can be extremely similar to each other (consequently providing similar diagnostic information). Even worse, (4) sometimes the finite paths with highest probability do not indicate the most likely violation of the property under consideration.

For example, look at the Markov chain $\mathcal{D}$ in Figure 1. The property $\mathcal{D} \models_{\leq 0.5} \Diamond\psi$ stating that execution reaches a state satisfying ψ (i.e., reaches s_3 or s_4) with probability lower or equal than 0.5 is violated (since the probability of reaching ψ is 1). The left hand side of table in Figure 2 lists finite paths reaching ψ ranked according to their probability. Note that finite paths with highest probability take the left branch in the system, whereas the right branch in itself has higher probability, illustrating Problem 4. To adjust the model so that it does satisfy the property (bug fixing), it is not sufficient to modify the left hand side of the system alone; no matter how one changes the left hand side, the probability of reaching ψ remains at least 0.6. Furthermore, the first six finite paths provide similar diagnostic information: they just make extra loops in s_1. This is an example of Problem 3. Additionally, the probability of every single finite path is far below the bound 0.5, making it unclear if a particular path is important; see Problem 2 above. Finally, the (unique) counterexample for the property $\mathcal{D} \models_{<1} \Diamond\psi$ consists of infinitely many finite paths (namely all finite paths of $\mathcal{D}$); see Problem 1. To overcome these problems, we partition a representative counterexample into sets of

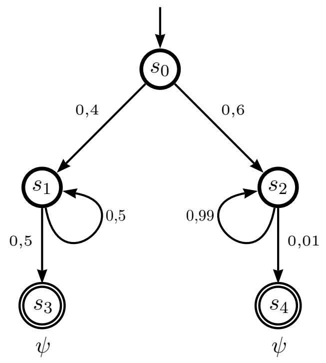


Fig. 1: Markov chain

	Single paths		**Witnesses**	
Rank	F. Path	Prob	Witness	Mass
1	$s_0(s_1)^1 s_3$	0.2	$[s_0 s_2 s_4]$	0.6
2	$s_0(s_1)^2 s_3$	0.1	$[s_0 s_1 s_3]$	0.4
3	$s_0(s_1)^3 s_3$	0.05		
4	$s_0(s_1)^4 s_3$	0.025		
5	$s_0(s_1)^5 s_3$	0.0125		
6	$s_0(s_1)^6 s_3$	0.00625		
7	$s_0(s_2)^1 s_4$	0.006		
8	$s_0(s_2)^2 s_4$	0.0059		
9	$s_0(s_2)^3 s_4$	0.0058		
⋮	⋮	⋮		

Fig. 2: Comparison Table

finite paths that follow a similar pattern. We call these sets *witnesses*. To ensure that witnesses provide valuable diagnostic information, we desire that the set of witnesses that form a counterexample satisfies several properties: two different witnesses should provide different diagnostic information (solving Problem 3) and elements of a single witness should provide similar diagnostic information, as a consequence witnesses have a high probability mass (solving Problems 2 and 4), and the number of witnesses of a representative counterexample should be finite (solving Problem 1).

In our setting, witnesses consist of paths that behave the same outside strongly connected components. In the example of Figure 1, there are two witnesses: the set of all finite paths going right, represented by $[s_0 s_2 s_4]$ whose probability (mass) is 0.6, and the set of all finite paths going left, represented by $[s_0 s_1 s_3]$ with probability (mass) 0.4.

In this paper, we show how to obtain such sets of witnesses for bounded probabilistic LTL properties on Markov Decision Processes (MDP). In fact, we first show how to reduce this problem to finding witnesses for upper bounded probabilistic reachability properties on discrete time Markov chains (MCs). The major technical matters lie on this last problem to which most of the paper is devoted.

In a nutshell, the process to find witnesses for the violation of $\mathcal{D} \models_{\leq p} \Diamond\psi$, with $\mathcal{D}$ being an MC, is as follows. We first eliminate from the original MC all the "uninteresting" parts. This proceeds as the first steps of the model checking process: make absorbing all states satisfying ψ, and all states that cannot reach ψ, obtaining a new MC $\mathcal{D}_\psi$. Next reduce this last MC to an acyclic MC $\mathrm{Ac}(\mathcal{D}_\psi)$ in which all strongly connected components have been conveniently abstracted with a single probabilistic transition. The original and the acyclic MCs are related by a mapping that, to each finite path in $\mathrm{Ac}(\mathcal{D}_\psi)$ (that we call *rail*), assigns a set of finite paths behaving similarly in $\mathcal{D}$ (that we call *torrent*). This map preserves the probability of reaching ψ and hence relates counterexamples in $\mathrm{Ac}(\mathcal{D}_\psi)$ to counterexamples in $\mathcal{D}$. Finally, counterexamples in $\mathrm{Ac}(\mathcal{D}_\psi)$ are computed by reducing the problem to a k shortest path problem, as in [HK07a]. Because $\mathrm{Ac}(\mathcal{D}_\psi)$ is acyclic, the complexity is lower than the corresponding problem in [HK07a].

It is worth mentioning that our technique can also be applied to pCTL formulas without nested path quantifiers.

Organization of the paper. Section 2 presents the necessary background on Markov chains (MC), Markov Decision Processes (MDP), and Linear Temporal Logic (LTL). Section 3 presents the definition of counterexamples and discusses the reduction from general LTL formulas to upper bounded probabilistic reachability properties, and the extraction of the maximizing MC in an MDP. Section 4 discusses desired properties of counterexamples. In Sections 5 and 6 we introduce the fundamentals on rails and torrents, the reduction of the original MC to the acyclic one, and our notion of significant diagnostic counterexamples. Section 7 then presents the techniques to actually compute counterexamples. In Section 8 we discuss related work and give final conclusions.

2 Preliminaries

2.1 Markov Decision Processes

Markov Decision Processes (MDPs) constitute a formalism that combines nondeterministic and probabilistic choices. They are an important model in corporate finance, supply chain optimization, system verification and optimization. There are many slightly different variants of this formalism such as action-labeled MDPs [Bel57,FV97], probabilistic automata [SL95,SdV04]; we work with the state-labeled MDPs from [BdA95].

Definition 2.1. Let S be a finite set. A *probability distribution* on S is a function $p\colon S \to [0,1]$ such that $\sum_{s\in S} p(s) = 1$. We denote the set of all probability distributions on S by $\mathrm{Distr}(S)$. Additionally, we define the *Dirac distribution* on an element $s \in S$ as 1_s, i.e., $1_s(s) = 1$ and $1_s(t) = 0$ for all $t \in S \setminus \{s\}$.

Definition 2.2. A *Markov Decision Process* (MDP) is a quadruple $\mathcal{M} = (S, s_0, L, \tau)$, where

- S is the finite state space;
- $s_0 \in S$ is the initial state;
- L is a labeling function that associates to each state $s \in S$ a set $L(s)$ of propositional variables that are *valid* in s;
- $\tau\colon S \to \wp(\mathrm{Distr}(S))$ is a function that associates to each $s \in S$ a non-empty and finite subset of $\mathrm{Distr}(S)$ of probability distributions.

Definition 2.3. Let $\mathcal{M} = (S, s_0, \tau, L)$ be an MDP. We define a *successor* relation $\delta \subseteq S \times S$ by $\delta \triangleq \{(s,t) | \exists\, \pi \in \tau(s) \,.\, \pi(t) > 0\}$ and for each state $s \in S$ we define the sets

$$\mathrm{Paths}(\mathcal{M}, s) \triangleq \{t_0 t_1 t_2 \ldots \in S^\omega | t_0 = s \wedge \forall n \in \mathbb{N} \,.\, \delta(t_n, t_{n+1})\} \text{ and}$$

$$\mathrm{Paths}^\star(\mathcal{M}, s) \triangleq \{t_0 t_1 \ldots t_n \in S^\star | t_0 = s \wedge \forall\, 0 \leq i < n \,.\, \delta(t_n, t_{n+1})\}$$

of paths of $\mathcal{D}$ and finite paths of $\mathcal{D}$ respectively beginning at s. We usually omit $\mathcal{M}$ from the notation; we also abbreviate $\text{Paths}(\mathcal{M}, s_0)$ as $\text{Paths}(\mathcal{M})$ and $\text{Paths}^\star(\mathcal{M}, s_0)$ as $\text{Paths}^\star(\mathcal{M})$. For $\omega \in \text{Paths}(s)$, we write the $(n{+}1)$-st state of ω as ω_n. As usual, we let $\mathcal{B}_s \subseteq \wp(\text{Paths}(s))$ be the Borel σ-algebra on the cones $\langle t_0 \dots t_n \rangle \triangleq \{\omega \in \text{Paths}(s) | \omega_0 = t_0 \wedge \dots \wedge \omega_n = t_n\}$. Additionally, for a set of finite paths $\Lambda \subseteq \text{Paths}^\star(s)$, we define $\langle \Lambda \rangle \triangleq \bigcup_{\sigma \in \Lambda} \langle \sigma \rangle$.

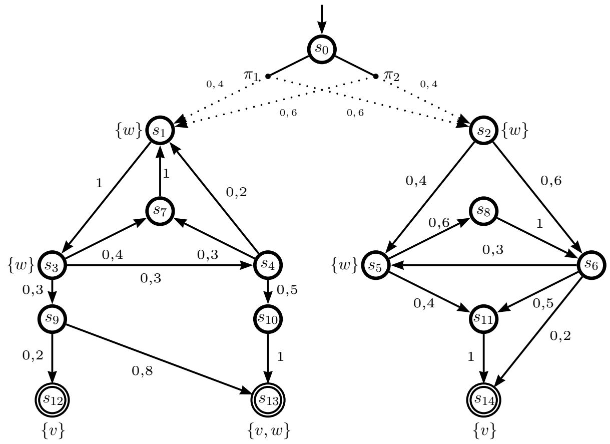


Fig. 3: Markov Decision Process

Figure 3 shows an MDP. Absorbing states (i.e., states s with $\tau(s) = \{1_s\}$) are represented by double lines. This MDP features a single nondeterministic decision, to be made in state s_0, namely π_1 and π_2.

Definition 2.4. Let $\mathcal{M} = (S, s_0, \tau, L)$ be an MDP, $s \in S$ and $\mathcal{A} \subseteq S$. We define the sets of paths and finite paths reaching $\mathcal{A}$ from s as

$$\text{Reach}(\mathcal{M}, s, \mathcal{A}) \triangleq \{\omega \in \text{Paths}(\mathcal{M}, s) \mid \exists_{i \geq 0}. \omega_i \in \mathcal{A}\} \text{ and}$$
$$\text{Reach}^\star(\mathcal{M}, s, \mathcal{A}) \triangleq \{\sigma \in \text{Paths}^\star(\mathcal{M}, s) \mid \text{last}(\sigma) \in \mathcal{A} \wedge \forall_{i \leq |\sigma|-1}. \sigma_i \notin \mathcal{A}\}$$

respectively. Note that $\text{Reach}^\star(\mathcal{M}, s, \mathcal{A})$ consists of those finite paths σ starting on s reaching $\mathcal{A}$ exactly once, at the end of the execution. It is easy to check that these sets are *prefix free*, i.e. contain finite paths such that none of them is a prefix of another one.

2.2 Schedulers

Schedulers (also called strategies, adversaries, or policies) resolve the nondeterministic choices in an MDP [PZ93,Var85,BdA95].

Definition 2.5. Let $\mathcal{M} = (S, s_0, \tau, L)$ be an MDP. A *scheduler* η on $\mathcal{M}$ is a function from $\text{Paths}^\star(\mathcal{M})$ to $\text{Distr}(\wp(\text{Distr}(S)))$ such that for all $\sigma \in \text{Paths}^\star(\mathcal{M})$ we have $\eta(\sigma) \in \text{Distr}(\tau(\text{last}(\sigma)))$. We denote the set of all schedulers on $\mathcal{M}$ by $\text{Sch}(\mathcal{M})$.

Note that our schedulers are randomized, i.e., in a finite path σ a scheduler chooses an element of $\tau(\text{last}(\sigma))$ probabilistically. Under a scheduler η, the probability that the next state reached after the path σ is t, equals $\sum_{\pi \in \tau(\text{last}(\sigma))} \eta(\sigma)(\pi) \cdot \pi(t)$. In this way, a scheduler induces a probability measure on $\mathcal{B}_s$ as usual.

Definition 2.6. Let $\mathcal{M} = (S, s_0, \tau, L)$ be an MDP and η a scheduler on $\mathcal{M}$. We define the probability measure $\mathbf{Pr}_\eta$ as the unique measure on $\mathcal{B}_{s_0}$ such that for all $s_0 s_1 \dots s_n \in \text{Paths}^\star(\mathcal{M})$

$$\mathbf{Pr}_\eta(\langle s_0 s_1 \dots s_n \rangle) = \prod_{i=0}^{n-1} \sum_{\pi \in \tau(s_i)} \eta(s_0 s_1 \dots s_i)(\pi) \cdot \pi(s_{i+1}).$$

We now recall the notions of deterministic and memoryless schedulers.

Definition 2.7. Let $\mathcal{M}$ be an MDP and η a scheduler on $\mathcal{M}$. We say that η is *deterministic* if $\eta(\sigma)(\pi_i)$ is either 0 or 1 for all $\pi_i \in \tau(\mathrm{last}(\sigma))$ and all $\sigma \in \mathrm{Paths}^\star(\mathcal{M})$. We say that a scheduler is *memoryless* if for all finite paths σ_1, σ_2 of $\mathcal{M}$ with $\mathrm{last}(\sigma_1) = \mathrm{last}(\sigma_2)$ we have $\eta(\sigma_1) = \eta(\sigma_2)$

Definition 2.8. Let $\mathcal{M}$ be an MDP and $\Delta \in \mathcal{B}_{s_0}$. Then the *maximal probability* $\mathbf{Pr}^+$ and *minimal probability* $\mathbf{Pr}^-$ of Δ are defined by

$$\mathbf{Pr}^+(\Delta) \triangleq \sup_{\eta \in \mathrm{Sch}(\mathcal{M})} \mathbf{Pr}_\eta(\Delta) \quad \text{and} \quad \mathbf{Pr}^-(\Delta) \triangleq \inf_{\eta \in \mathrm{Sch}(\mathcal{M})} \mathbf{Pr}_\eta(\Delta).$$

A scheduler that attains $\mathbf{Pr}^+(\Delta)$ or $\mathbf{Pr}^-(\Delta)$ is called a *maximizing* or *minimizing* scheduler respectively.

2.3 Markov chains

A (discrete time) *Markov chain* is an MDP associating exactly one probability distribution to each state. In this way nondeterministic choices are not longer allowed.

Definition 2.9 (Markov chain). Let $\mathcal{M} = (S, s_0, \tau, L)$ be an MDP. If $|\tau(s)| = 1$ for all $s \in S$, then we say that $\mathcal{M}$ is a *Markov chain* (MC).

In order to simplify notation we represent probabilistic transitions on MCs by means of a probabilistic matrix $\mathcal{P}$ instead of τ. Additionally, we denote by $\mathbf{Pr}_{\mathcal{D},s}$ the probability measure induced by a MC $\mathcal{D}$ with initial state s and we abbreviate $\mathbf{Pr}_{\mathcal{D},s_0}$ as $\mathbf{Pr}_{\mathcal{D}}$.

2.4 Linear Temporal Logic

Linear temporal logic (LTL) [MP91] is a modal temporal logic with modalities referring to time. In LTL is possible to encode formulas about the future of paths: a condition will eventually be true, a condition will be true until another fact becomes true, etc.

Definition 2.10. LTL is built up from the set of propositional variables $\mathcal{V}$, the logical connectives $\neg$, $\wedge$, and a temporal modal operator by the following grammar:

$$\phi ::= \mathcal{V} \mid \neg\phi \mid \phi \wedge \phi \mid \phi \mathcal{U} \phi.$$

Using these operators we define $\vee, \rightarrow, \Diamond$, and $\Box$ in the standard way.

Definition 2.11. Let $\mathcal{M} = (S, s_0, \tau, L)$ be an MDP. We define satisfiability for paths ω in $\mathcal{M}$, propositional variables $v \in \mathcal{V}$, and LTL formulas ϕ, γ inductively by

$$\begin{array}{ll} \omega \models_{\mathcal{M}} v \quad \Leftrightarrow v \in L(\omega_0) & \omega \models_{\mathcal{M}} \phi \wedge \gamma \Leftrightarrow \omega \models_{\mathcal{M}} \phi \text{ and } \omega \models_{\mathcal{M}} \gamma \\ \omega \models_{\mathcal{M}} \neg\phi \Leftrightarrow \mathrm{not}(\omega \models_{\mathcal{M}} \phi) & \omega \models_{\mathcal{M}} \phi \mathcal{U} \gamma \ \Leftrightarrow \exists_{i \geq 0}.\omega_{\downarrow i} \models_{\mathcal{M}} \gamma \text{ and } \forall_{0 \leq j < i}.\omega_{\downarrow j} \models_{\mathcal{M}} \phi \end{array}$$

where $\omega_{\downarrow i}$ is the i-th suffix of ω. When confusion is unlikely, we omit the subscript $\mathcal{M}$ on the satisfiability relation.

Definition 2.12. Let $\mathcal{M}$ be an MDP. We define the language $\mathrm{Sat}_{\mathcal{M}}(\phi)$ associated to an LTL formula ϕ as the set of paths satisfying ϕ, i.e. $\mathrm{Sat}_{\mathcal{M}}(\phi) \triangleq \{\omega \in \mathrm{Paths}(\mathcal{M}) \mid \omega \models \phi\}$. Here we also generally omit the subscript $\mathcal{M}$.

We now define satisfiability of an LTL formula ϕ on an MDP $\mathcal{M}$. We say that $\mathcal{M}$ satisfies ϕ with probability at most p ($\mathcal{M} \models_{\leq p} \phi$) if the probability of getting an execution satisfying ϕ is at most p.

Definition 2.13. Let $\mathcal{M}$ be an MDP, ϕ an LTL formula and $p \in [0, 1]$. We define $\models_{\leq p}$ and $\models_{\geq p}$ by

$$\mathcal{M} \models_{\leq p} \phi \Leftrightarrow \mathbf{Pr}^+(\mathrm{Sat}(\phi)) \leq p,$$
$$\mathcal{M} \models_{\geq p} \phi \Leftrightarrow \mathbf{Pr}^-(\mathrm{Sat}(\phi)) \geq p.$$

We define $\mathcal{M} \models_{<p} \phi$ and $\mathcal{M} \models_{>p} \phi$ in a similar way. In case the MDP is fully probabilistic, i.e., an MC, the satisfiability problem is reduced to $\mathcal{M} \models_{\bowtie p} \phi \Leftrightarrow \mathbf{Pr}_{\mathcal{M}}(\mathrm{Sat}(\phi)) \bowtie p$, where $\bowtie \in \{<, \leq, >, \geq\}$.

3 Counterexamples

In this section, we define what counterexamples are and how the problem of finding counterexamples to a general LTL property over Markov Decision Processes reduces to finding counterexamples to reachability problems over Markov chains.

Definition 3.1 (Counterexamples). Let $\mathcal{M}$ be an MDP and ϕ an LTL formula. A *counterexample* to $\mathcal{M} \models_{\leq p} \phi$ is a measurable set $\mathcal{C} \subseteq \mathrm{Sat}(\phi)$ such that $\mathbf{Pr}^+(\mathcal{C}) > p$. Counterexamples to $\mathcal{M} \models_{<p} \phi$ are defined similarly.

Counterexamples to $\mathcal{M} \models_{>p} \phi$ and $\mathcal{M} \models_{\geq p} \phi$ cannot be defined straightforwardly as it is always possible to find a set $\mathcal{C} \subseteq \mathrm{Sat}(\phi)$ such that $\mathbf{Pr}^-(\mathcal{C}) \leq p$ or $\mathbf{Pr}^-(\mathcal{C}) < p$, note that the empty set trivially satisfies it. Therefore, the best way to find counterexamples to lower bounded probabilities is to find counterexamples to the dual properties $\mathcal{M} \models_{<1-p} \neg\phi$ and $\mathcal{M} \models_{\leq 1-p} \neg\phi$. That is, while for upper bounded probabilities, a counterexample is a set of paths satisfying the property with mass probability beyond the bound, for lower bounded probabilities the counterexample is a set of paths that *does not* satisfy the property with sufficient probability.

Example 1. Consider the MDP $\mathcal{M}$ of Figure 4 and the LTL formula $\Diamond v$. It is easy to check that $\mathcal{M} \not\models_{<1} \Diamond v$. The set $\mathcal{C} = \mathrm{Sat}(\Diamond v) = \{\rho \in \mathrm{Paths}(s_0) | \exists_{i \geq 0}.\rho = s_0(s_1)^i(s_4)^\omega\} \cup \{\rho \in \mathrm{Paths}(s_0) | \exists_{i \geq 0}.\rho = s_0(s_3)^i(s_5)^\omega\}$ is a counterexample. Note that $\mathbf{Pr}_\eta(\mathcal{C}) = 1$ where η is any deterministic scheduler on $\mathcal{M}$ satisfying $\eta(s_0) = \pi_1$.

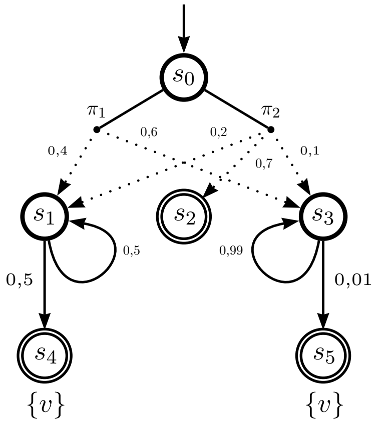


Fig. 4:

LTL formulas are actually checked by reducing the model checking problem to a reachability problem [dAKM97]. For checking upper bounded probabilities, the LTL formula is translated into an equivalent deterministic Rabin automaton and composed with the MDP under verification. On the obtained MDP, the set of states forming accepting end components (SCC that traps accepting conditions with probability 1) are identified. The maximum probability of the LTL property on the original MDP is the same as the maximum probability of reaching a state of an accepting end component in the final MDP. Hence, from now on we will focus on counterexamples to properties of the form $\mathcal{M} \models_{\leq p} \Diamond\psi$ or $\mathcal{M} \models_{<p} \Diamond\psi$, where ψ is a propositional formula, i.e., a formula without temporal operators.

In the following, it will be useful to identify the set of states in which a propositional property is valid.

Definition 3.2. Let $\mathcal{M}$ be an MDP. We define the state language $\mathrm{Sat}_\mathcal{M}(\psi)$ associated to a propositional formula ψ as the set of states satisfying ψ, i.e., $\mathrm{Sat}_\mathcal{M}(\psi) \triangleq \{s \in S \mid s \models \psi\}$, where $\models$ has the obvious satisfaction meaning for states. As usual, we generally omit the subscript $\mathcal{M}$.

We will show now that, in order to find a counterexample to a property in an MDP with respect to an upper bound, it suffices to find a counterexample for the MC *induced* by the maximizing scheduler. The maximizing scheduler turns out to be deterministic and memoryless [BdA95]; consequently the induced Markov chain can be easily extracted from the MDP as follows.

Definition 3.3. Let $\mathcal{M} = (S, s_0, \tau, L)$ be an MDP and η a deterministic memoryless scheduler. Then we define the MC induced by η as $\mathcal{M}_\eta = (S, s_0, \mathcal{P}_\eta, L)$ where $\mathcal{P}_\eta(s,t) = (\eta(s))(t)$ for all $s, t \in S$.

Now we state that finding counterexamples to upper bounded probabilistic reachability LTL properties on MDPs can be reduced to finding counterexamples to upper bounded probabilistic reachability LTL properties on MCs.

Theorem 3.4. *Let $\mathcal{M}$ be an* MDP*, ψ a propositional formula and $p \in [0,1]$. Then, there is a maximizing (deterministic memoryless) scheduler η such that $\mathcal{M} \models_{\leq p} \Diamond\psi \Leftrightarrow \mathcal{M}_\eta \models_{\leq p} \Diamond\psi$. Moreover, if $\mathcal{C}$ is a counterexample to $\mathcal{M}_\eta \models_{\leq p} \Diamond\psi$ then $\mathcal{C}$ is also a counterexample to $\mathcal{M} \models_{\leq p} \Diamond\psi$.*

Note that η can be computed by solving a linear minimization problem [BdA95]. See Section 7.1.

4 Representative Counterexamples, Partitions and Witnesses

The notion of counterexample from Definition 3.1 is very broad: just an arbitrary (measurable) set of paths with high enough mass probability. To be useful as a debugging tool (and in fact to be able to present the counterexample to a user), we need counterexamples with specific properties. We will partition counterexamples (or rather, representative counterexamples) in witnesses and list five informal properties that we consider valuable in order to increase the quality of witnesses as a debugging tool.

We first note that for reachability properties it is sufficient to consider counterexamples that consist of finite paths.

Definition 4.1 (Representative counterexamples). Let $\mathcal{M}$ be an MDP, ψ a propositional formula and $p \in [0,1]$. A *representative counterexample* to $\mathcal{M} \models_{\leq p} \Diamond\psi$ is a set $\mathcal{C} \subseteq \mathrm{Reach}^\star(\mathcal{M}, \mathrm{Sat}(\psi))$ such that $\mathbf{Pr}^+(\langle\mathcal{C}\rangle) > p$. We denote the set of all representative counterexamples to $\mathcal{M} \models_{\leq p} \Diamond\psi$ by $\mathcal{R}(\mathcal{M}, p, \psi)$.

Theorem 4.2. *Let $\mathcal{M}$ be an* MDP*, ψ a propositional formula and $p \in [0,1]$. If $\mathcal{C}$ is a representative counterexample to $\mathcal{M} \models_{\leq p} \Diamond\psi$, then $\langle\mathcal{C}\rangle$ is a counterexample to $\mathcal{M} \models_{\leq p} \Diamond\psi$. Furthermore, there exists a counterexample to $\mathcal{M} \models_{\leq p} \Diamond\psi$ if and only if there exists a representative counterexample to $\mathcal{M} \models_{\leq p} \Diamond\psi$.*

Following [HK07a], we present the notions of *minimum counterexample*, *strongest evidence* and *most indicative counterexamples*.

Definition 4.3 (Minimum counterexample). Let $\mathcal{D}$ be an MC, ψ a propositional formula and $p \in [0,1]$. We say that $\mathcal{C} \in \mathcal{R}(\mathcal{D}, p, \psi)$ is a *minimum counterexample* if $|\mathcal{C}| \leq |\mathcal{C}'|$, for all $\mathcal{C}' \in \mathcal{R}(\mathcal{D}, p, \psi)$.

Definition 4.4 (Strongest evidence). Let $\mathcal{D}$ be an MC, ψ a propositional formula and $p \in [0,1]$. A *strongest evidence* to $\mathcal{D} \not\models_{\leq p} \Diamond\psi$ is a finite path $\sigma \in \mathrm{Reach}^\star(\mathcal{D}, \mathrm{Sat}(\psi))$ such that $\mathbf{Pr}_{\mathcal{D}}(\langle\sigma\rangle) \geq \mathbf{Pr}_{\mathcal{D}}(\langle\rho\rangle)$, for all $\rho \in \mathrm{Reach}^\star(\mathcal{D}, \mathrm{Sat}(\psi))$.

Definition 4.5 (Most indicative counterexample). Let $\mathcal{D}$ be an MC, ψ a propositional formula and $p \in [0,1]$. We call $\mathcal{C} \in \mathcal{R}(\mathcal{D}, p, \psi)$ a *most indicative counterexample* if it is minimum and $\mathbf{Pr}_{\mathcal{D}}(\langle\mathcal{C}\rangle) \geq \mathbf{Pr}_{\mathcal{D}}(\langle\mathcal{C}'\rangle)$, for all minimum counterexamples $\mathcal{C}' \in \mathcal{R}(\mathcal{D}, p, \psi)$.

Unfortunately, very often most indicative counterexamples are very large (even infinite), many of its elements have insignificant measure and elements can be extremely similar to each other (consequently providing the same diagnostic information). Even worse, sometimes the finite paths with highest probability do not exhibit the way in which the system accumulates higher probability to reach the undesired property (and consequently where an error occurs with higher probability). For these reasons, we are of the opinion that representative counterexamples are still too general in order to be useful as feedback information. We approach this problem by refining a representative counterexample into sets of finite paths following a "similarity" criteria (introduced in Section 5). These sets are called *witnesses* of the counterexample.

Recall that a set Y of nonempty sets is a partition of X if the elements of Y cover X and are pairwise disjoint. We define counterexample partitions in the following way.

Definition 4.6 (Counterexample partitions and witnesses). Let $\mathcal{M}$ be an MDP, ψ a propositional formula, $p \in [0,1]$, and $\mathcal{C}$ a representative counterexample to $\mathcal{M} \models_{\leq p} \Diamond\psi$. A *counterexample partition* $W_{\mathcal{C}}$ is a partition of $\mathcal{C}$. We call the elements of $W_{\mathcal{C}}$ *witnesses*.

Since not every partition generates useful witnesses (from the debugging perspective), we now state five informal properties that we consider valuable in order to improve the diagnostic information provided by witnesses. In Section 7 we show how to partition the representative counterexample in order to obtain witnesses satisfying most of these properties.

Similarity: Elements of a witness should provide similar debugging information.
Accuracy: Witnesses with higher probability should exhibit evolutions of the system with higher probability of containing errors.
Originality: Different witnesses should provide different debugging information.
Significance: Witnesses should be as closed to the counterexample as possible (their mass probability should be as closed as possible to the bound p).
Finiteness: The number of witnesses of a counterexample partition should be finite.

5 Rails and Torrents

As argued before we consider that representative counterexamples are excessively general to be useful as feedback information. Therefore, we group finite paths of a representative counterexample in witnesses if they are "similar enough". We will consider finite paths that behave the same outside SCCs of the system as providing similar feedback information.

In order to formalize this idea, we first reduce the original MC $\mathcal{D}$ to an acyclic MC preserving reachability probabilities. We do so by removing all SCCs K of $\mathcal{D}$ keeping just *input states* of K. In this way, we get a new acyclic MC denoted by $\mathrm{Ac}(\mathcal{D})$. The probability matrix of the Markov chain relates input states of each SCC to its *output states* with the reachability probability between these states in $\mathcal{D}$. Secondly, we establish a map between finite paths σ in $\mathrm{Ac}(\mathcal{D})$ (*rails*) and sets of paths W_σ in $\mathcal{D}$ (*torrents*). Each torrent contains finite paths that are similar, i.e., behave the same outside SCCs. We conclude the section showing that the probability of σ is equal to the mass probability of W_σ.

Reduction to Acyclic Markov chains

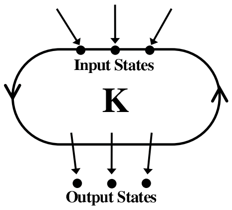
Consider an MC $\mathcal{D} = (S, s_0, \mathcal{P}, L)$. Recall that a subset $\mathrm{K} \subseteq S$ is called *strongly connected* if for every $s, t \in \mathrm{K}$ there is a finite path from s to t. Additionally K is called a *strongly connected component* (SCC) if it is a maximally (with respect to $\subseteq$) strongly connected subset of S.

Note that every state is a member of exactly one SCC of $\mathcal{D}$; even those states that are not involved in cycles, since the trivial finite path s connects s to itself. We call *trivial strongly connected components* to the SCCs containing absorbing states or states not involved in cycles (note that trivial SCCs are composed by one single state). From now on we let $\mathrm{SCC}^\star$ be the set of non trivial strongly connected components of an MC.

A Markov chain is called *acyclic* if it contains only trivial SCCs. Note that an acyclic Markov chain still has absorbing states.

Definition 5.1 (Input and Output states). Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be an MC. Then, for each $\mathrm{SCC}^\star$ K of $\mathcal{D}$, we define the sets $\mathrm{Inp}_{\mathrm{K}} \subseteq S$ of all states in K that have an incoming transition from a state outside of K and $\mathrm{Out}_{\mathrm{K}} \subseteq S$ of all states outside of K that have an incoming transition from a state of K in the following way

$$\mathrm{Inp}_{\mathrm{K}} \triangleq \{t \in \mathrm{K} \mid \exists\, s \in S \setminus \mathrm{K} \,.\, \mathcal{P}(s,t) > 0\},$$
$$\mathrm{Out}_{\mathrm{K}} \triangleq \{s \in S \setminus \mathrm{K} \mid \exists\, t \in \mathrm{K} \,.\, \mathcal{P}(t,s) > 0\}.$$



We also define for each SCC* K an MC related to K as $\mathcal{D}_{\mathrm{K}} \triangleq (\mathrm{K} \cup \mathrm{Out}_{\mathrm{K}}, s_{\mathrm{K}}, \mathcal{P}_{\mathrm{K}}, L_{\mathrm{K}})$ where s_{K} is any state in $\mathrm{Inp}_{\mathrm{K}}$, $L_{\mathrm{K}}(s) \triangleq L(s)$, and $\mathcal{P}_{\mathrm{K}}(s,t)$ is equal to $\mathcal{P}(s,t)$ if $s \in \mathrm{K}$ and equal to 1_s otherwise. Additionally, for every state s involved in non trivial SCCs we define SCC_s^+ as $\mathcal{D}_{\mathrm{K}}$, where K is the SCC* of $\mathcal{D}$ such that $s \in \mathrm{K}$.

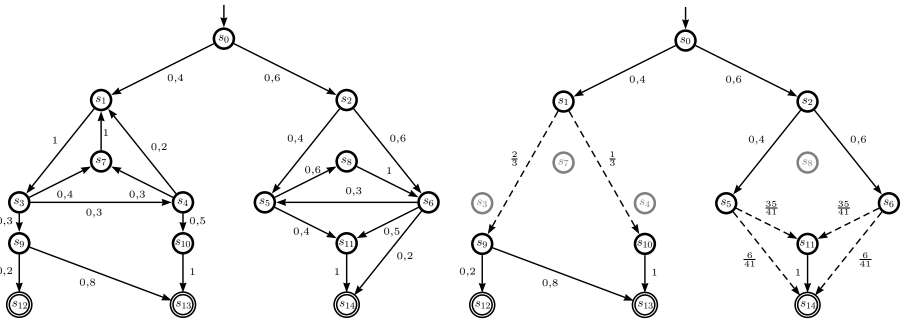
Now we are able to define an acyclic MC $\mathrm{Ac}(\mathcal{D})$ related to $\mathcal{D}$.

Definition 5.2. Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be a MC. We define $\mathrm{Ac}(\mathcal{D}) \triangleq (S', s_0, \mathcal{P}', L')$ where

- $S' \triangleq \overbrace{S \setminus \bigcup_{\mathrm{K} \in \mathrm{SCC}^\star} \mathrm{K}}^{S_{\mathrm{com}}} \cup \overbrace{\bigcup_{\mathrm{K} \in \mathrm{SCC}^\star} \mathrm{Inp}_{\mathrm{K}}}^{S_{\mathrm{inp}}}$,
- $L' \triangleq L_{|_{S'}}$,
- $\mathcal{P}'(s,t) \triangleq \begin{cases} \mathcal{P}(s,t) & \text{if } s \in S_{com}, \\ \mathbf{Pr}_{\mathcal{D},s}(\mathrm{Reach}(\mathrm{SCC}_s^+, s, \{t\})) & \text{if } s \in S_{inp} \wedge t \in \mathrm{Out}_{\mathrm{SCC}_s^+}, \\ 1_s & \text{if } s \in S_{inp} \wedge \mathrm{Out}_{\mathrm{SCC}_s^+} = \emptyset, \\ 0 & \text{otherwise.} \end{cases}$

Note that $\mathrm{Ac}(\mathcal{D})$ is indeed acyclic.

Example 2. Consider the MC $\mathcal{D}$ of Figure 5(a). The strongly connected components of $\mathcal{D}$ are $\mathrm{K}_1 \triangleq \{s_1, s_3, s_4, s_7\}$, $\mathrm{K}_2 \triangleq \{s_5, s_6, s_8\}$ and the singletons $\{s_0\}$, $\{s_2\}$, $\{s_9\}$, $\{s_{10}\}$, $\{s_{11}\}$, $\{s_{12}\}$, $\{s_{13}\}$, and $\{s_{14}\}$. The input states of K_1 are $\mathrm{Inp}_{\mathrm{K}_1} = \{s_1\}$ and its output states are $\mathrm{Out}_{\mathrm{K}_1} = \{s_9, s_{10}\}$. For K_2, $\mathrm{Inp}_{\mathrm{K}_2} = \{s_5, s_6\}$ and $\mathrm{Out}_{\mathrm{K}_2} = \{s_{11}, s_{14}\}$. The reduced acyclic MC of $\mathcal{D}$ is shown in Figure 5(b).



(a) Original MC

(b) Derived Acyclic MC

Fig. 5:

Rails and Torrents

We now relate (finite) paths in $\mathrm{Ac}(\mathcal{D})$ (rails) to sets of paths in $\mathcal{D}$ (torrents).

Definition 5.3 (Rails). Let $\mathcal{D}$ be an MC. A finite path $\sigma \in \mathrm{Paths}^\star(\mathrm{Ac}(\mathcal{D}))$ will be called a *rail of $\mathcal{D}$*.

Consider a rail σ, i.e., a finite path of $\mathrm{Ac}(\mathcal{D})$. We will use σ to represent those paths ω of $\mathcal{D}$ that behave "similar to" σ outside SCCs of $\mathcal{D}$. Naively, this means that σ is a subsequence of ω. There are two technical subtleties to deal with: every input state in σ must be the first state in its SCC in ω (freshness) and every SCC visited by ω must be also visited by σ (inertia) (see Definition 5.5). We need these extra conditions to make sure that no path ω behaves "similar to" two distinct rails (see Lemma 5.7).

Recall that given a finite sequence σ and a (possible infinite) sequence ω, we say that σ is a *subsequence* of ω, denoted by $\sigma \sqsubseteq \omega$, if and only if there exists a strictly increasing function $f : \{0, 1, \ldots, |\sigma| - 1\} \to \{0, 1, \ldots, |\omega| - 1\}$ such that $\forall_{0 \le i < |\sigma|}.\sigma_i = \omega_{f(i)}$. If ω is an infinite sequence, we interpret the codomain of f as $\mathbb{N}$. In case f is such a function we write $\sigma \sqsubseteq_f \omega$.

Definition 5.4. Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be an MC. On S we consider the equivalence relation $\underset{\mathcal{D}}{\sim}$ satisfying $s \underset{\mathcal{D}}{\sim} t$ if and only if s and t are in the same strongly connected component. Again, we usually omit the subscript $\mathcal{D}$ from the notation.

The following definition refines the notion of subsequence, taking care of the two technical subtleties noted above.

Definition 5.5. Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be an MC, ω a (finite) path of $\mathcal{D}$, and $\sigma \in \mathrm{Paths}^\star(\mathrm{Ac}(\mathcal{D}))$ a finite path of $\mathrm{Ac}(\mathcal{D})$. Then we write $\sigma \preceq \omega$ if there exists $f : \{0, 1, \ldots, |\sigma| - 1\} \to \mathbb{N}$ such that $\sigma \sqsubseteq_f \omega$ and

$$\forall_{0 \le j < f(i)} : \omega_{f(i)} \not\sim \omega_j;\ \text{for all } i = 0, 1, \ldots |\sigma| - 1, \qquad [\textit{Freshness property}]$$
$$\forall_{f(i) < j < f(i+1)} : \omega_{f(i)} \sim \omega_j;\ \text{for all } i = 0, 1, \ldots |\sigma| - 2. \qquad [\textit{Inertia property}]$$

In case f is such a function we write $\sigma \preceq_f \omega$.

Example 3. Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be the MC of Figure 5(a) and take $\sigma = s_0 s_2 s_6 s_{14}$. Then for all $i \in \mathbb{N}$ we have $\sigma \preceq_{f_i} \omega_i$ where $\omega_i = s_0 s_2 s_6 (s_5 s_8 s_6)^i s_{14}$ and $f_i(0) \triangleq 0$, $f_i(1) \triangleq 1$, $f_i(2) \triangleq 2$, and $f_i(3) \triangleq 3 + 3i$. Additionally, $\sigma \not\preceq s_0 s_2 s_5 s_8 s_6 s_{14}$ since for all f satisfying $\sigma \sqsubseteq_f s_0 s_2 s_5 s_8 s_6 s_{14}$ we must have $f(2) = 5$; this implies that f does not satisfy the freshness property. Finally, note that $\sigma \not\preceq s_0 s_2 s_6 s_{11} s_{14}$ since for all f satisfying $\sigma \sqsubseteq_f s_0 s_2 s_6 s_{11} s_{14}$ we must have $f(2) = 2$; this implies that f does not satisfy the inertia property.

We now give the formal definition of torrents.

Definition 5.6 (Torrents). Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be an MC and σ a sequence of states in S. We define the function Torr by

$$\mathrm{Torr}(\mathcal{D}, \sigma) \triangleq \{\omega \in \mathrm{Paths}(\mathcal{D}) \mid \sigma \preceq \omega\}.$$

We call $\mathrm{Torr}(\mathcal{D}, \sigma)$ the *torrent* associated to σ.

We now show that torrents are disjoint (Lemma 5.7) and that the probability of a rail is equal to the probability of its associated torrent (Theorem 5.10). For this last result, we first show that torrents can be represented as the disjoint union of cones of finite paths. We call these finite paths *generators* of the torrent (Definition 5.8).

Lemma 5.7. *Let $\mathcal{D}$ be an* MC*. For every $\sigma, \rho \in \mathrm{Paths}^\star(\mathrm{Ac}(\mathcal{D}))$ we have*

$$\sigma \neq \rho \Rightarrow \mathrm{Torr}(\mathcal{D}, \sigma) \cap \mathrm{Torr}(\mathcal{D}, \rho) = \emptyset$$

Definition 5.8 (Torrent Generators). Let $\mathcal{D}$ be an MC. Then we define for every rail $\sigma \in \mathrm{Paths}^\star(\mathrm{Ac}(\mathcal{D}))$ the set

$$\mathrm{TorrGen}(\mathcal{D}, \sigma) \triangleq \{\rho \in \mathrm{Paths}^\star(\mathcal{D}) \mid \exists f : \sigma \preceq_f \rho \wedge f(|\sigma| - 1) = |\rho| - 1\}.$$

In the example from the Introduction (see Figure 1), $s_0 s_1 s_3$ and $s_0 s_2 s_4$ are rails. Their associated torrents are, respectively, $\{s_0 s_1^n s_3^\omega \mid n \in \mathbb{N}^*\}$ and $\{s_0 s_2^n s_4^\omega \mid n \in \mathbb{N}^*\}$ (note that s_3 and s_4 are absorbing states), i.e. the paths going left and the paths going right. The generators of the first torrent are $\{s_0 s_1^n s_3 \mid n \in \mathbb{N}^*\}$ and similarly for the second torrent.

Lemma 5.9. *Let $\mathcal{D}$ be an* MC *and $\sigma \in \mathrm{Paths}^\star(\mathrm{Ac}(\mathcal{D}))$ a rail of $\mathcal{D}$. Then we have*

$$\mathrm{Torr}(\mathcal{D}, \sigma) = \biguplus_{\rho \in \mathrm{TorrGen}(\mathcal{D}, \sigma)} \langle \rho \rangle.$$

Theorem 5.10. *Let $\mathcal{D}$ be an* MC*. Then for every rail $\sigma \in \mathrm{Paths}^\star(\mathrm{Ac}(\mathcal{D}))$ we have*

$$\mathbf{Pr}_{\mathrm{Ac}(\mathcal{D})}(\langle \sigma \rangle) = \mathbf{Pr}_{\mathcal{D}}(\mathrm{Torr}(\mathcal{D}, \sigma)).$$

6 Significant Diagnostic Counterexamples

So far we have formalized the notion of paths behaving similarly (i.e., behaving the same outside SCCs) in an MC $\mathcal{D}$ by removing all SCC of $\mathcal{D}$, obtaining $\mathrm{Ac}(\mathcal{D})$. A representative counterexample to $\mathrm{Ac}(\mathcal{D}) \models_{\leq p} \Diamond\psi$ gives rise to a representative counterexample to $\mathcal{D} \models_{\leq p} \Diamond\psi$ in the following way: for every finite path σ in the representative counterexample to $\mathrm{Ac}(\mathcal{D}) \models_{\leq p} \Diamond\psi$ the set $\mathrm{TorrGen}(\mathcal{D}, \sigma)$ is a witness, then we obtain the desired representative counterexample to $\mathcal{D} \models_{\leq p} \Diamond\psi$ by taking the union of these witnesses.

Before giving a formal definition, there is still one technical issue to resolve: we need to be sure that by removing SCCs we are not discarding useful information. Because torrents are built from rails, we need to make sure that when we discard SCCs, we do not discard rails that reach ψ.

We achieve this by first making states satisfying ψ absorbing. Additionally, we make absorbing states from which it is not possible to reach ψ. Note that this does not affect counterexamples.

Definition 6.1. Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be an MC and ψ a propositional formula. We define the MC $\mathcal{D}_\psi \triangleq (S, s_0, \mathcal{P}_\psi, L)$, with

$$\mathcal{P}_\psi(s,t) \triangleq \begin{cases} 1 & \text{if } s \notin \mathrm{Sat}_\Diamond(\psi) \wedge s = t, \\ 1 & \text{if } s \in \mathrm{Sat}(\psi) \wedge s = t, \\ \mathcal{P}(s,t) & \text{if } s \in \mathrm{Sat}_\Diamond(\psi) - \mathrm{Sat}(\psi), \\ 0 & \text{otherwise,} \end{cases}$$

where $\mathrm{Sat}_\Diamond(\psi) \triangleq \{s \in S \mid \mathbf{Pr}_{\mathcal{D},s}(\mathrm{Reach}(\mathcal{D}, s, \mathrm{Sat}(\psi))) > 0\}$ is the set of states reaching ψ in $\mathcal{D}$.

The following theorem shows the relation between paths, finite paths, and probabilities of $\mathcal{D}$, $\mathcal{D}_\psi$, and $\mathrm{Ac}(\mathcal{D}_\psi)$. Most importantly, the probability of a rail σ (in $\mathrm{Ac}(\mathcal{D}_\psi)$) is equal to the probability of its associated torrent (in $\mathcal{D}$) (item 5 below) and the probability of $\Diamond\psi$ is not affected by reducing $\mathcal{D}$ to $\mathrm{Ac}(\mathcal{D}_\psi)$ (item 6 below).

Note that a rail σ is always a finite path in $\mathrm{Ac}(\mathcal{D}_\psi)$, but that we can talk about its associated torrent $\mathrm{Torr}(\mathcal{D}_\psi, \sigma)$ in $\mathcal{D}_\psi$ and about its associated torrent $\mathrm{Torr}(\mathcal{D}, \sigma)$ in $\mathcal{D}$. The former exists for technical convenience; it is the latter that we are ultimately interested in. The following theorem also shows that for our purposes, viz. the definition of the generators of the torrent and the probability of the torrent, there is no difference (items 3 and 4 below).

Theorem 6.2. *Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be an* MC *and ψ a propositional formula. Then for every $\sigma \in \mathrm{Paths}^\star(\mathcal{D}_\psi)$*

1. $\mathrm{Reach}^\star(\mathcal{D}_\psi, s_0, \mathrm{Sat}(\psi)) = \mathrm{Reach}^\star(\mathcal{D}, s_0, \mathrm{Sat}(\psi))$,
2. $\mathbf{Pr}_{\mathcal{D}_\psi}(\langle\sigma\rangle) = \mathbf{Pr}_{\mathcal{D}}(\langle\sigma\rangle)$,
3. $\mathrm{TorrGen}(\mathcal{D}_\psi, \sigma) = \mathrm{TorrGen}(\mathcal{D}, \sigma)$,
4. $\mathbf{Pr}_{\mathcal{D}_\psi}(\mathrm{Torr}(\mathcal{D}_\psi, \sigma)) = \mathbf{Pr}_{\mathcal{D}}(\mathrm{Torr}(\mathcal{D}, \sigma))$,
5. $\mathbf{Pr}_{\mathrm{Ac}(\mathcal{D}_\psi)}(\langle\sigma\rangle) = \mathbf{Pr}_{\mathcal{D}}(\mathrm{Torr}(\mathcal{D}, \sigma))$,
6. $\mathrm{Ac}(\mathcal{D}_\psi) \models_{\leq p} \Diamond\psi$ *if and only if* $\mathcal{D} \models_{\leq p} \Diamond\psi$, *for any* $p \in [0, 1]$.

Proof. Straightforward □

Definition 6.3 (Torrent-Counterexamples). Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be an MC, ψ a propositional formula, and $p \in [0,1]$ such that $\mathcal{D} \not\models_{\leq p} \Diamond\psi$. Let $\mathcal{C}$ be a representative counterexample to $\mathrm{Ac}(\mathcal{D}_\psi) \models_{\leq p} \Diamond\psi$. We define the set

$$\mathrm{TorRepCount}(\mathcal{C}) \triangleq \{\mathrm{TorrGen}(\mathcal{D}, \sigma) \mid \sigma \in \mathcal{C}\}.$$

We call the set $\mathrm{TorRepCount}(\mathcal{C})$ a *torrent-counterexample* of $\mathcal{C}$. Note that this set is a partition of a representative counterexample to $\mathcal{D} \models_{\leq p} \Diamond\psi$. Additionally, we denote by $\mathcal{R}_t(\mathcal{D}, p, \psi)$ to the set of all torrent-counterexamples to $\mathcal{D} \models_{\leq p} \Diamond\psi$, i.e., $\{\mathrm{TorRepCount}(\mathcal{C}) \mid \mathcal{C} \in \mathcal{R}(\mathrm{Ac}(\mathcal{D}), p, \psi)\}$.

Theorem 6.4. *Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be an* MC*, ψ a propositional formula, and $p \in [0,1]$ such that $\mathcal{D} \not\models_{\leq p} \Diamond\psi$. Take $\mathcal{C}$ a representative counterexample to $\mathrm{Ac}(\mathcal{D}_\psi) \models_{\leq p} \Diamond\psi$. Then the set of finite paths $\biguplus_{W \in \mathrm{TorRepCount}(\mathcal{C})} W$ is a representative counterexample to $\mathcal{D} \models_{\leq p} \Diamond\psi$.*

Note that for each $\sigma \in \mathcal{C}$ we get a witness $\mathrm{TorrGen}(\mathcal{D}, \sigma)$. Also note that the number of rails is finite, so there are also only finitely many witnesses.

Following [HK07a], we extend the notions of *minimum counterexamples* and *strongest evidence*.

Definition 6.5 (Minimum torrent-counterexample). Let $\mathcal{D}$ be an MC, ψ a propositional formula and $p \in [0,1]$. We say that $\mathcal{C}_t \in \mathcal{R}_t(\mathcal{D}, p, \psi)$ is a *minimum torrent-counterexample* if $|\mathcal{C}_t| \leq |\mathcal{C}'_t|$, for all $\mathcal{C}'_t \in \mathcal{R}_t(\mathcal{D}, p, \psi)$.

Definition 6.6 (Strongest torrent-evidence). Let $\mathcal{D}$ be an MC, ψ a propositional formula and $p \in [0,1]$. A *strongest torrent-evidence* to $\mathcal{D} \not\models_{\leq p} \Diamond\psi$ is a torrent $\mathrm{Torr}(\mathcal{D}, \sigma)$ such that $\sigma \in \mathrm{Paths}^\star(\mathrm{Ac}(\mathcal{D}_\psi))$ and $\mathbf{Pr}_\mathcal{D}(\mathrm{Torr}(\mathcal{D}, \sigma)) \geq \mathbf{Pr}_\mathcal{D}(\mathrm{Torr}(\mathcal{D}, \rho))$ for all $\rho \in \mathrm{Paths}^\star(\mathrm{Ac}(\mathcal{D}_\psi))$.

Now we define our notion of significant diagnostic counterexamples. It is the generalization of most indicative counterexample from [HK07a] to our setting.

Definition 6.7 (Most indicative torrent-counterexample). Let $\mathcal{D}$ be an MC, ψ a propositional formula and $p \in [0,1]$. We call $\mathcal{C}_t \in \mathcal{R}_t(\mathcal{D}, p, \psi)$ a *most indicative torrent-counterexample* if it is a minimum torrent-counterexample and $\mathbf{Pr}(\bigcup_{T \in \mathcal{C}_t} \langle T \rangle) \geq \mathbf{Pr}(\bigcup_{T \in \mathcal{C}'_t} \langle T \rangle)$ for all minimum torrent-counterexamples $\mathcal{C}'_t \in \mathcal{R}_t(\mathcal{D}, p, \psi)$.

Note that in our setting, as in [HK07a], a minimal torrent-counterexample $\mathcal{C}$ consists of the $|\mathcal{C}|$ strongest torrent-evidences.

By Theorem 6.4 it is possible to obtain strongest torrent-evidence and most indicative torrent-counterexamples of an MC $\mathcal{D}$ by obtaining strongest evidence and most indicative counterexamples of $\mathrm{Ac}(\mathcal{D}_\psi)$ respectively.

7 Computing Counterexamples

In this section we show how to compute most indicative torrent-counterexamples. We also discuss what information to present to the user: how to present witnesses and how to deal with overly large strongly connected components.

7.1 Maximizing Schedulers

The calculation of the maximal probability on a reachability problem can be performed by solving a linear minimization problem [BdA95,dA97]. This minimization problem is defined on a system of inequalities that has a variable x_i for each different state s_i and an inequality $\sum_j \pi(s_j) \cdot x_j \leq x_i$ for each distribution $\pi \in \tau(s_i)$. The maximizing (deterministic memoryless) scheduler η can be easily extracted out of such system of inequalities after obtaining the solution. If $p_0, \ldots, p_n$ are the values that minimize $\sum_i x_i$ in the previous system, then η is such that, for all s_i, $\eta(s_i) = \pi$ whenever $\sum_j \pi(s_j) \cdot p_j = p_i$. In the following we denote $\mathbf{P}_{s_i}[\Diamond\psi] \triangleq x_i$.

7.2 Computing most indicative torrent-counterexamples

We divide the computation of most indicative torrent-counterexamples to $\mathcal{M} \models_{\leq p} \Diamond\psi$ in three stages: *pre-processing*, SCC *analysis*, and *searching*.

Pre-processing stage. We first modify the original MC $\mathcal{D}$ by making all states in $\mathrm{Sat}(\psi) \cup S \setminus \mathrm{Sat}_\Diamond(\psi)$ absorbing. In this way we obtain the MC $\mathcal{D}_\psi$ from Definition 6.1. Note that we do not have to spend additional computational resources to compute this set, since $\mathrm{Sat}_\Diamond(\psi) = \{s \in S \mid \mathbf{P}_s[\Diamond\psi] > 0\}$ and hence all required data is already available from the LTL model checking phase.

SCC **analysis stage.** We remove all SCCs K of $\mathcal{D}_\psi$ keeping just *input states* of K, getting the acyclic MC $\mathrm{Ac}(\mathcal{D}_\psi)$ according to Definition 5.2.

To compute this, we first need to find the SCCs of $\mathcal{D}_\psi$. There exists several well known algorithms to achieve this: Kosaraju's, Tarjan's, Gabow's algorithms (among others). We also have to compute the reachability probability from input states to output states of every SCC. This can be done by using steady state analysis techniques [Cas93].

Searching stage. To find most indicative torrent-counterexamples in $\mathcal{D}$, we find most indicative counterexamples in $\mathrm{Ac}(\mathcal{D}_\psi)$. For this we use the same approach as [HK07a], turning the MC into a weighted digraph to replace the problem of finding the finite path with highest probability by a shortest path problem. The nodes of the digraph are the states of the MC and there is an edge between s and t if $\mathcal{P}(s,t) > 0$. The weight of such an edge is $-\log(\mathcal{P}(s,t))$.

Finding the most indicative counterexample in $\mathrm{Ac}(\mathcal{D}_\psi)$ is now reduced to finding k shortest paths. As explained in [HK07a], our algorithm has to compute k on the fly. Eppstein's algorithm [Epp98] produces the k shortest paths in general in $O(m + n \log n + k)$, where m is the number of nodes and n the number of edges. In our case, since $\mathrm{Ac}(\mathcal{D}_\psi)$ is acyclic, the complexity decreases to $O(m + k)$.

7.3 Debugging issues

Representative finite paths. What we have computed so far is a most indicative counterexample to $\mathrm{Ac}(\mathcal{D}_\psi) \models_{\leq p} \Diamond\psi$. This is a finite set of rails, i.e., a finite set of paths in $\mathrm{Ac}(\mathcal{D}_\psi)$. Each of these paths σ represents a witness $\mathrm{TorrGen}(\mathcal{D}, \sigma)$. Note that this witness itself has usually infinitely many elements.

In practice, one has to display a witness to the user. The obvious way would be to show the user the rail σ. This, however, may be confusing to the user as σ is not a finite path of the original Markov Decision Process. Instead of presenting the user with σ, we therefore show the user the finite path of $\mathrm{TorrGen}(\mathcal{D}, \sigma)$ with highest probability.

Definition 7.1. Let $\mathcal{D}$ be an MC, and $\sigma \in \mathrm{Paths}^\star(\mathrm{Ac}(\mathcal{D}_\psi))$ a rail of $\mathcal{D}$. We define the *representant of* $\mathrm{Torr}(\mathcal{D}, \sigma)$ as

$$\mathrm{repTorr}\,(\mathcal{D}, \sigma) = \mathrm{repTorr}\left(\biguplus_{\rho \in \mathrm{TorrGen}(\mathcal{D},\sigma)} \langle\rho\rangle \right) \triangleq \arg \max_{\rho \in \mathrm{TorrGen}(\mathcal{D},\sigma)} \mathbf{Pr}(\langle\rho\rangle)$$

Note that given $\mathrm{repTorr}\,(\mathcal{D}, \sigma)$ one can easily recover σ. Therefore, no information is lost by presenting torrents as one of its generators instead of as a rail.

Expanding SCC**.** Note that in the Preprocessing stage, we reduced the size of many SCCs of the system (and likely even completely removed some) by making states in $\mathrm{Sat}(\psi) \cup S \setminus \mathrm{Sat}_\Diamond(\psi)$ absorbing. However, It is possible that the system still contains some very large strongly connected components. In that case, a single witness could have a very large probability mass and one could argue that the information presented to the user is not detailed enough. For instance, consider the Markov chain of Figure 6 in which there is a single large SCC with input state t and output state u.

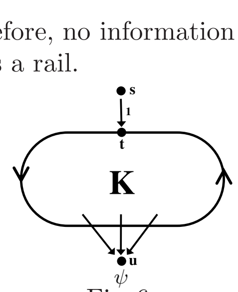


Fig. 6:

The most indicative torrent-counterexample to the property $\mathcal{D} \models_{\leq 0.9} \Diamond\psi$ is simply $\{\mathrm{TorrGen}(stu)\}$, i.e., a single witness with probability mass 1 associated to the rail stu. Although this may seem uninformative, we argue that it is more informative than listing several paths of the form $st \cdots u$ with probability summing up to, say, 0.91. Our single witness counterexample suggests that the outgoing transition to a state not reaching ψ was simply forgotten in the design; the listing of paths still allows the possibility that one of the probabilities in the whole system is simply wrong.

Nevertheless, if the user needs more information to tackle bugs inside SCCs, note that there is more information available at this point. In particular, for every strongly connected component K, every input state s of K (even for every state in K), and every output state t of K, the probability of reaching t from s is already available from the computation of $\mathrm{Ac}(\mathcal{D}_\psi)$ during the SCC analysis stage of Section 7.2.

8 Final Discussion

We have presented a novel technique for representing and computing counterexamples for nondeterministic and probabilistic systems. We partition a counterexample in witnesses and state five properties that we consider valuable in order to increase the utility of witnesses as a debugging tool: (similarity) elements of a witness should provide similar debugging information; (originality) different witnesses should provide different debugging information; (accuracy) witnesses with higher probability should indicate system behavior more likely to contain errors; (significance) probability of a witness should be relatively high; (finiteness) there should be finitely many witnesses. We achieve this by grouping finite paths in a counterexample together in a witness if they behave the same outside the strongly connected components.

Presently, some work has been done on counterexample generation techniques for different variants of probabilistic models (Discrete Markov chains and Continues Markov chains) [AHL05,AL06,HK07a,HK07b]. In our terminology, these works consider witnesses consisting of a *single* finite path. We have already discussed in the Introduction that the single path approach does not meet the properties of accuracy, originality, significance, and finiteness.

Instead, our witness/torrent approach provides a high level of abstraction of a counterexample. By grouping together finite paths that behave the same outside strongly connected components in a single witness, we can achieve these properties to a higher extent. Behaving the same outside strongly connected components is a reasonable way of formalizing the concept of providing *similar* debugging information. This grouping also makes witnesses significantly different from each other: each witness comes from a different rail and each rail provides a different way to reach the undesired property. Then each witness provides *original* information. Of course, our witnesses are more *significant* than single finite paths, because they are sets of finite paths. This also gives us more *accuracy* than the approach with single finite paths, as a collection of finite paths behaving the same and reaching an undesired condition with high probability is more likely to show how the system reaches this condition than just a single path. Finally, because there is a finite number of rails, there is also a *finite* number of witnesses.

Another key difference of our work with previous ones is that our technique allows to generate counterexamples for probabilistic systems *with* nondeterminism. However, a recent report [AL07] also considers counterexample generation for MDPs. Their approach only extends to upper bounded pCTL formulas without nested temporal operators. We would like to remark that our technique to approach counterexample generation for MDPs completely differs from theirs.

Finally, we are not aware of any other work in the literature considering counterexamples for probabilistic LTL model checking.

The authors would like to stress the important result of [HK07a], which provides a systematic characterization of counterexample generation in terms of shortest paths problems. We use this result to generate counterexamples for the acyclic Markov chains.

In the future we intend to implement a tool to generate our significant diagnostic counterexamples; a very preliminary version has already been implemented. There is still work to be done on improving the visualization of the witnesses, in particular, when a witness captures a large strongly connected component. Another direction is to investigate how this work can be extended to timed systems, either modeled with continuous time Markov chains or with probabilistic timed automata.

Acknowledgement. The authors thank David Jansen for helpful comments on an earlier version of this paper.

References

[AD06] Miguel E. Andrés and Pedro D'Argenio. Derivation of counterexamples for quantitative model checking. Master's thesis, Universidad Nacional de Córdoba, 2006.

[AHL05] Husain Aljazzar, Holger Hermanns, and Stefan Leue. Counterexamples for timed probabilistic reachability. In *Formal Modeling and Analysis of Timed Systems (FORMATS '05)*, volume 3829, pages 177–195, 2005.

[AL06] Husain Aljazzar and Stefan Leue. Extended directed search for probabilistic timed reachability. In *Formal Modeling and Analysis of Timed Systems (FORMATS '06)*, pages 33–51, 2006.

[AL07] Husain Aljazzar and Stefan Leue. Counterexamples for model checking of markov decision processes. Computer Science Technical Report soft-08-01, University of Konstanz, December 2007.

[BdA95] Andrea Bianco and Luca de Alfaro. Model checking of probabilistic and nondeterministic systems. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Foundations of Software Technology and Theoretical Computer Science (FSTTCS '95)*, volume 1026, pages 499–513, 1995.

[Bel57] Richard E. Bellman. A Markovian decision process. *J. Math. Mech.*, 6:679–684, 1957.

[BLR05] Gerd Behrmann, Kim G. Larsen, and Jacob I. Rasmussen. Optimal scheduling using priced timed automata. *SIGMETRICS Perform. Eval. Rev.*, 32(4):34–40, 2005.

[Cas93] Christos G. Cassandras. *Discrete Event Systems: Modeling and Performance Analysis*. Richard D. Irwin, Inc., and Aksen Associates, Inc., 1993.

[CGJ+00] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In *Computer Aided Verification*, pages 154–169, 2000.

[dA97] Luca de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.

[dAKM97] Luca de Alfaro, Arjun Kapur, and Zohar Manna. Hybrid diagrams: A deductive-algorithmic approach to hybrid system verification. In *Symposium on Theoretical Aspects of Computer Science*, pages 153–164, 1997.

[Epp98] David Eppstein. Finding the k shortest paths. In *SIAM Journal of Computing*, pages 652–673, 1998.

[FV97] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. 1997.

[HK07a] Tingting Han and Joost-Pieter Katoen. Counterexamples in probabilistic model checking. In *Tools and Algorithms for the Construction and Analysis of Systems: 13th International Conference (TACAS '07)*, volume 4424, pages 60–75, 2007.

[HK07b] Tingting Han and Joost-Pieter Katoen. Providing evidence of likely being on time counterexample generation for ctmc model checking. In *International Symposium on Automated Technology for Verification and Analysis (ATVA '07)*, volume 4762, pages 331–346, 2007.

[MP91] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer, 1991.

[PZ93] Amir Pnueli and Lenore D. Zuck. Probabilistic verification. *Information and Computation*, 103(1):1–29, 1993.

[SdV04] Ana Sokolova and Erik P. de Vink. Probabilistic automata: System types, parallel composition and comparison. In Christel Baier, Boudewijn R. Haverkort, Holger Hermans, Joost-Pieter Katoen, and Markus Siegle, editors, *Validation of Stochastic Systems: A Guide to Current Research*, volume 2925, pages 1–43. 2004.

[SL95] Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

[Var85] M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state systems. In *Proc. 26th IEEE Symp. Found. Comp. Sci.*, pages 327–338, 1985.