Open access • Journal Article • DOI:10.1109/TSE.2009.5

Counterexample Generation in Probabilistic Model Checking — Source link

Tingting Han, Joost-Pieter Katoen, D. Berteun

Institutions: RWTH Aachen University

Published on: 01 Mar 2009 - IEEE Transactions on Software Engineering (IEEE)

Topics: Counterexample, Probabilistic CTL, Markov process, Markov chain and Markov decision process

Counterexample Generation in Probabilistic Model Checking

Tingting Han, Joost-Pieter Katoen, *Member*, *IEEE Computer Society*, and Berteun Damman

Abstract—Providing evidence for the refutation of a property is an essential, if not the most important, feature of model checking. This paper considers algorithms for counterexample generation for probabilistic CTL formulas in discrete-time Markov chains. Finding the strongest evidence (i.e., the most probable path) violating a (bounded) until-formula is shown to be reducible to a single-source (hop-constrained) shortest path problem. Counterexamples of smallest size that deviate most from the required probability bound can be obtained by applying (small amendments to) k-shortest (hop-constrained) paths algorithms. These results can be extended to Markov chains with rewards, to LTL model checking, and are useful for Markov decision processes. Experimental results show that, typically, the size of a counterexample is excessive. To obtain much more compact representations, we present a simple algorithm to generate (minimal) regular expressions that can act as counterexamples. The feasibility of our approach is illustrated by means of two communication protocols: leader election in an anonymous ring network and the Crowds protocol.

Index Terms—Diagnostic feedback, Markov chain, model checking, regular expression, shortest path.

1 Introduction

A major strength of model checking is the possibility to generate counterexamples in case a property is violated. They are of utmost importance in model checking: First, and for all, they provide diagnostic feedback even in cases where only a fragment of the entire model can be searched. They also constitute the key to successful abstraction-refinement techniques [15] and are at the core of obtaining feasible schedules in, e.g., timed model checking [11]. As a result, advanced counterexample generation and analysis techniques have intensively been investigated, see, e.g., [39], [10], [21].

The shape of a counterexample depends on the checked formula and the temporal logic. For logics such as LTL, typically finite or infinite paths through the model are required. The violation of linear-time safety properties is indicated by finite paths that end in a "bad" state. Liveness properties instead require infinite paths ending in a cyclic behavior indicating that something "good" will never happen. LTL model checkers usually incorporate breadth-first search algorithms to generate *shortest* counterexamples, i.e., paths of minimal length. For branching-time logics such as CTL, paths may act as counterexamples for a subclass of universally quantified formulas, i.e., those in ACTL∩LTL. To cover a broader spectrum of formulas, though, more advanced structures such as trees of paths [16], proof-like counterexamples [29] (for ACTL\LTL), or annotated paths [58] (for ECTL) are used.

• *The authors are with the Department of Computer Science, RWTH Aachen University, Ahornstraße 55, D-52074 Aachen, Germany, and with the Department of Computer Science, University of Twente, PO Box 217, NL-7500 AE Enschede, The Netherlands.*
E-mail: {tingting.han, katoen}@cs.rwth-aachen.de, berteun@dds.nl.

Manuscript received 27 Jan. 2008; revised 8 Sept. 2008, accepted 23 Sept. 2008; published online 21 Jan. 2009.
Recommended for acceptance by J. Hillston, M. Kwiatkowska, and M. Telek. For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSESI-2008-01-0042. Digital Object Identifier no. 10.1109/TSE.2009.5.

This paper considers the generation of counterexamples in probabilistic model checking. Probabilistic model checking is a technique to verify system models in which transitions are equipped with random information. Popular models are discrete and continuous-time Markov chains (DTMCs and CTMCs, respectively), and variants thereof which exhibit nondeterminism. Efficient model-checking algorithms for these models have been developed, have been implemented in a variety of software tools, and have been applied to case studies from various application areas ranging from randomized distributed algorithms, computer systems, and security protocols to biological systems and quantum computing. The crux of probabilistic model checking is to appropriately combine techniques from numerical mathematics and operations research with standard reachability analysis. In this way, properties such as "the (maximal) probability to reach a set of goal states by avoiding certain states is at most 0.6" can be automatically checked up to a user-defined precision. Markovian models comprising millions of states can be checked rather fast by dedicated tools such as PRISM [46] and MRMC [42], as well as extensions to existing tools such as GreatSPN, SPIN, PEPA Workbench, and Statemate.

In probabilistic model checking, however, counterexample generation is almost not developed; a notable exception is the recent heuristic search algorithm for CTMCs and DTMCs [3], [4] that works under the assumption that the model is unknown. Instead, we consider a setting in which it has already been established that a certain state refutes a given property. This paper considers algorithms, complexity results, and experimental results for the generation of counterexamples in probabilistic model checking. The considered setting is probabilistic CTL [33] for DTMCs, a model in which all transitions are equipped with a probability. In this setting, typically there is no single path, but rather a *set* of paths that indicates why a given property is refuted. We first concentrate on

properties of the form $\mathcal{P}_{\leq p}(\Phi\ \mathrm{U}^{\leq h}\Psi)$, where Φ and Ψ characterize sets of states, p is a probability, and h a (possibly infinite) bound on the maximal allowed number of steps before reaching a goal (i.e., a Ψ) state. In case state s refutes this formula, the probability of all paths in s satisfying $\Phi\ \mathrm{U}^{\leq h}\Psi$ exceeds p. We consider two problems that are aimed to provide useful diagnostic feedback for this violation: generating strongest evidences and smallest counterexamples.

Strongest evidences are the most probable paths that satisfy $\Phi \mathrm{U}^{\leq h}\Psi$. They "contribute" mostly to the property refutation and are thus expected to be informative. For unbounded until (i.e., $h = \infty$), determining strongest evidences is shown to be equivalent to a standard single-source shortest path (SP) problem; in case h is bounded, we obtain a special case of the (resource) constrained shortest path (CSP) problem [2] that can be solved in $\mathcal{O}(hm)$, where m is the number of transitions in the DTMC. Alternatively, the Viterbi algorithm [61], [40] can be used for bounded h yielding the same time complexity.

Evidently, strongest evidences often do not suffice as true counterexamples as their probability mass lies (far) below p. As a next step, therefore, we consider the problem of determining most probable subtrees (rooted at s). Similar to the notion of shortest counterexample in LTL model checking, we consider trees of *smallest size* that exceed the probability bound p. Additionally, such trees, of size k, say, are required to *maximally* exceed the lower bound, i.e., no subtrees should exist of size at most k that exceed p. The problem of generating such *smallest counterexamples* can be cast as a k shortest paths problem. For unbounded-until formulas (i.e., $h = \infty$), the generation of such smallest counterexamples can be carried out in pseudopolynomial time by adopting k shortest paths algorithms [26], [22] that compute k on the fly. For bounded until-formulas, we propose an adaptation of the recursive enumeration algorithm (REA) of Jiménez and Marzal [38]. The time complexity of this adapted algorithm is $\mathcal{O}(hm+hk\log(\frac{m}{n}))$, where n is the number of states in the DTMC.

This approach is applicable to probability thresholds with lower bounds, i.e., formulas of the form $\mathcal{P}_{\geq p}(\Phi \mathrm{U}^{\leq h}\Psi)$, as well as to the logic LTL. It is applicable to various other models such as Markov reward models and Markov decision processes (MDPs) once a scheduler for an MDP violating an until-formula is obtained. It also provides the basis for counterexample generation techniques for time-bounded reachability in CTMCs [31], CEGAR techniques for MDPs [35], and counterexamples for the logic cpCTL [8]. Heuristic search algorithms for CTMC counterexamples are provided in [3], [4]. Counterexamples for refinement of probabilistic programs have recently been considered in [53].

Once we have established the theoretical underpinnings, we report on experiments that apply our counterexample generation algorithms to example DTMCs. Using the synchronous leader election protocol [36], we show that the size of counterexamples may be double exponential in terms of the input parameters of the protocol (like number of processes and rounds). In order to obtain insight into this phenomenon, we provide a short mathematical analysis of the number of evidences in counterexamples in this protocol. The resulting closed-form expression confirms the double exponential growth. To achieve a more succinct representation we propose to use regular expressions. The advantage of regular expressions is that they are commonly known, are easy to understand, and may be very compact. The idea is to represent a DTMC by a deterministic finite-state automaton (DFA, for short) and obtain regular expressions by applying successive state elimination where the order of state elimination is determined heuristically [32]. The computation of the probability of a regular expression is performed using the approach advocated by Daws [20] for parametric model checking of DTMCs. This boils down to a recursive evaluation which is guaranteed to be exact (i.e., no rounding errors), provided the transition probabilities are rational. We provide the details of this approach and show its result when applied to the leader election protocol. We briefly argue that model reduction such as bisimulation and SCC elimination [50] can be used to obtain even more compact counterexamples. Finally, we show the generation of counterexamples on the Crowds protocol [56], a protocol for anonymous Web browsing that has been adopted, among others, to Bluetooth [59] and wireless Internet [5].

The paper is organized as follows: Section 2 introduces DTMCs and PCTL logic. Section 3 considers the notion of evidences and counterexamples. Section 4 shows the adaptation of a DTMC to a weighted digraph. Sections 5 and 6 consider the algorithms for generating strongest evidences and smallest counterexamples, respectively. Sections 7 and 8 extend the approach to lower-bound probability operators, the qualitative fragment of PCTL and rewards, respectively. Section 9 discusses the implementation details as well as the leader election case study. Section 10 presents the algorithm for the regular expressions. Section 11 considers the Crowds protocol, and Section 12 concludes.

This paper is an extension of [30] and [19].

2 Preliminaries

2.1 Markov Chains

Let AP be a fixed, finite set of atomic propositions ranged over by $a, b, c, \ldots$.

Definition 1 (DTMCs). *A (labeled)* discrete-time Markov chain (DTMC) *is a triple $\mathcal{D} = (S, \mathbf{P}, L)$, where:*

- *S is a finite set of states;*
- *$\mathbf{P} : S \times S \to [0, 1]$ is a stochastic matrix;*
- *$L : S \to 2^{\mathrm{AP}}$ is a labeling function which assigns to each state $s \in S$ the set $L(s)$ of atomic propositions that are valid in s.*

Intuitively, a DTMC is a Kripke structure in which all transitions are equipped with discrete probabilities such that the sum of outgoing transitions of each state equals 1. A state s in $\mathcal{D}$ is called *absorbing* if $\mathbf{P}(s, s) = 1$. Without loss of generality, we assume a DTMC to have a unique initial state.

Definition 2 (Paths). *Let $\mathcal{D} = (S, \mathbf{P}, L)$ be a DTMC.*

- *An* infinite path *ρ in $\mathcal{D}$ is an infinite sequence $s_0{\cdot}s_1{\cdot}s_2{\cdot}s$ of states such that $\forall i \geq 0.\ \mathbf{P}(s_i, s_{i+1}) > 0$.*
- *A finite path σ is a finite prefix of an infinite path.*

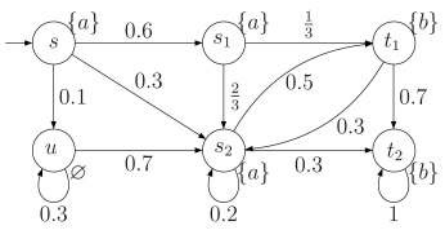

Fig. 1. An example DTMC.

Let $Paths^{\omega}_{\mathcal{D}}(s)$ denote the set of all infinite paths in $\mathcal{D}$ that start in state s and $Paths^{\star}_{\mathcal{D}}(s)$ denote the set of all finite paths of s. The subscript $\mathcal{D}$ is omitted when it is clear from the context. For state s and finite path $\sigma = s_0 \cdots s_n$ with $\mathbf{P}(s_n, s) > 0$, let $\sigma{\cdot}s$ denote the path obtained by extending σ by s.

Let τ denote either a finite or an infinite path. Let $|\tau|$ denote the length of τ, i.e., $|s_0{\cdot}s_1 \cdots s_n| = n$, $|s_0| = 0$, and $|\tau| = \infty$ for infinite τ. For $0 \le i \le |\tau|$, $\tau[i] = s_i$ denotes the $(i+1)$th state in τ. We use $\tau{\downarrow_i}$ to denote the prefix of τ truncated at length i (thus ending in s_i), formally, $\tau{\downarrow_i} = \tau[0]{\cdot}\tau[1]{\cdots}\tau[i]$. We use $Pref(\tau)$ to denote the set of prefixes of τ, i.e., $Pref(\tau) = \{\tau{\downarrow_i} \mid 0 \le i \le |\tau|\}$.

A DTMC $\mathcal{D}$ induces a probability space. The underlying σ algebra is defined over the basic cylinder set induced by the finite paths starting in the initial state s_0. The probability measure $\Pr^{\mathcal{D}}_{s_0}$ (briefly $\Pr$) induced by $(\mathcal{D}, s_0)$ is the unique measure on this σ algebra where:

$$\Pr\{\underbrace{\rho \in Paths^{\omega}_{\mathcal{D}}(s_0) \mid \rho{\downarrow_n} = s_0 \cdots s_n}_{Cyl(s_0 \cdots s_n)}\} = \prod_{0 \le i < n} \mathbf{P}(s_i, s_{i+1}).$$

The probability of finite path $\sigma = s_0 \cdots s_n$ is defined as $\mathbb{P}(\sigma) = \prod_{0 \le i < n} \mathbf{P}(s_i, s_{i+1})$. Note that, although $\Pr(Cyl(\sigma)) = \mathbb{P}(\sigma)$, they have different meanings: $\Pr$ is a measure on infinite paths whereas $\mathbb{P}$ refers to finite ones. For a set C of finite paths which is *prefix containment free*, i.e., for any $\sigma, \sigma' \in C$ with $\sigma \neq \sigma'$, $\sigma \notin Pref(\sigma')$, the probability of C is $\mathbb{P}(C) = \sum_{\sigma \in C} \mathbb{P}(\sigma)$. Paths in C induce disjoint cylinder sets.

Example 1. Fig. 1 illustrates a DTMC with initial state s. $\mathrm{AP} = \{a, b\}$ and L is given as $L(s) = L(s_i) = \{a\}$, for $i = 1, 2$; $L(t_1) = L(t_2) = \{b\}$ and $L(u) = \emptyset$. t_2 is an absorbing state. $\sigma_1 = s{\cdot}u{\cdot}s_2{\cdot}t_1{\cdot}t_2$ is a finite path with $\mathbb{P}(\sigma_1) = 0.1 \times 0.7 \times 0.5 \times 0.7$ and $|\sigma_1| = 4$, $\sigma_1[3] = t_1$. $\rho_1 = s{\cdot}(s_2{\cdot}t_1)^{\omega}$ is an infinite path.

2.2 Logic

Probabilistic computation tree logic (PCTL) [33] is an extension of CTL in which state formulas are interpreted over states of a DTMC and path formulas are interpreted over infinite paths in a DTMC. The syntax of PCTL is:

$$\Phi ::= \mathrm{tt} \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{P}_{\trianglelefteq p}(\phi),$$

where $p \in [0, 1]$ is a probability, $\trianglelefteq \in \{<, \le, >, \ge\}$, and ϕ is a path formula defined according to the following grammar:

$$\phi ::= \Phi \mathrm{U}^{\le h} \Phi \mid \Phi \mathrm{W}^{\le h} \Phi,$$

where $h \in \mathbb{N}_{\ge 0} \cup \{\infty\}$. The path formula $\Phi \mathrm{U}^{\le h} \Psi$ asserts that Ψ is satisfied within h transitions and that all preceding states satisfy Φ. For $h = \infty$ such path formulas are standard (unbounded) until-formulas, whereas, in other cases, these are bounded until-formulas. $\mathrm{W}^{\le h}$ is the weak counterpart of $\mathrm{U}^{\le h}$ which does not require Ψ to eventually become true. For the sake of simplicity, we do not consider the next-operator. The temporal operators $\Diamond^{\le h}$ and $\Box^{\le h}$ are obtained as follows:

$$\mathcal{P}_{\trianglelefteq p}(\Diamond^{\le h}\Phi) = \mathcal{P}_{\trianglelefteq p}(\mathrm{tt}\mathrm{U}^{\le h}\Phi),$$
$$\mathcal{P}_{\trianglelefteq p}(\Box^{\le h}\Phi) = \mathcal{P}_{\trianglelefteq p}(\Phi\mathrm{W}^{\le h}\mathrm{ff}).$$

Note that $\mathrm{ff} = \neg\mathrm{tt}$. The example formula $\mathcal{P}_{\le 0.5}(a\mathrm{U}b)$ asserts that the probability of reaching a b-state via an a-path is at most 0.5 and $\mathcal{P}_{>0.001}(\Diamond^{\le 50} error)$ states that the probability for a system error within 50 steps exceeds 0.001. Dually, $\mathcal{P}_{<0.999}(\Box^{\le 50}\neg error)$ states that the probability for no error in the next 50 steps is less than 0.999.

Semantics. Let DTMC $\mathcal{D} = (S, \mathbf{P}, L)$. The semantics of PCTL is defined by a satisfaction relation, denoted by $\models$, which is characterized as the least relation over the states in S (infinite paths in $\mathcal{D}$, respectively) and the state formulas (path formulas) satisfying:

$$\begin{array}{lll} s \models \mathrm{tt} & & \\ s \models a & \text{iff} & a \in L(s), \\ s \models \neg\Phi & \text{iff} & \text{not } (s \models \Phi), \\ s \models \Phi \wedge \Psi & \text{iff} & s \models \Phi \text{ and } s \models \Psi, \\ s \models \mathcal{P}_{\trianglelefteq p}(\phi) & \text{iff} & Prob(s, \phi) \trianglelefteq p. \end{array}$$

Let $Paths^{\omega}(s, \phi)$ denote the set of infinite paths that start in state s and satisfy ϕ. To put it in a more formal way, $Paths^{\omega}(s, \phi) = \{\rho \in Paths^{\omega}(s) \mid \rho \models \phi\}$. Then, $Prob(s, \phi) = \Pr\{\rho \mid \rho \in Paths^{\omega}(s, \phi)\}$. Let ρ be an infinite path in $\mathcal{D}$. The semantics of PCTL path formulas is defined as:

$$\begin{array}{lll} \rho \models \Phi\mathrm{U}^{\le h}\,\Psi & \text{iff} & \exists\, i{\le}h.\big(\rho[i] \models \Psi \wedge \forall\, 0{\le}j{<}i.\ \rho[j] \models \Phi\big), \\ \rho \models \Phi\mathrm{W}^{\le h}\,\Psi & \text{iff} & \text{either } \rho \models \Phi\mathrm{U}^{\le h}\,\Psi \text{ or } \forall\, i{\le}h.\ \rho[i] \models \Phi. \end{array}$$

For finite path σ, the semantics of path formulas is defined in a similar way by changing the range of variable i to $i \le \min\{h, |\sigma|\}$. There is a close relationship between until and weak until. More precisely, for any state s and PCTL formulas Φ and Ψ:

$$\mathcal{P}_{\ge p}(\Phi\mathrm{W}^{\le h}\Psi) \equiv \mathcal{P}_{\le 1-p}\big((\Phi \wedge \neg\Psi)\mathrm{U}^{\le h}(\neg\Phi \wedge \neg\Psi),$$
$$\mathcal{P}_{\ge p}(\Phi\mathrm{U}^{\le h}\Psi) \equiv \mathcal{P}_{\le 1-p}\big((\Phi \wedge \neg\Psi)\ \mathrm{W}^{\le h}\ (\neg\Phi \wedge \neg\Psi)\big).$$

This relationship is used later on to show that counterexamples for formulas with probability lower bounds can be obtained using algorithms for formulas with upper bounds.

Let ϕ be an until-formula, i.e., $\phi = \Phi\mathrm{U}^{\le h}\Psi$. Let $Paths^{\star}(s, \phi)$ denote the set of finite paths starting in s that fulfil ϕ. For finite path σ, the relation $\models_{\min}$ denotes the minimal satisfaction of a PCTL path formula. Formally, $\sigma \models_{\min} \phi$ iff $\sigma \models \phi$ and $\sigma' \not\models \phi$ for any $\sigma' \in Pref(\sigma)\backslash\{\sigma\}$.

Example 2. For the PCTL state formula $\mathcal{P}_{\le 0.95}(a\mathrm{U}b)$ and the DTMC $\mathcal{D}$ in Fig. 1, let path $\sigma = s{\cdot}s_2{\cdot}t_1{\cdot}t_2$. $\sigma \models a\mathrm{U}b$ but $\sigma \not\models_{\min} a\mathrm{U}b$. $s \models \mathcal{P}_{\le 0.95}(a\mathrm{U}b)$ since $Prob(s, a\mathrm{U}b) = 0.9$.

Let $Paths^{\star}_{\min}(s, \phi) = \{\sigma \in Paths^{\star}(s) \mid \sigma \models_{\min} \phi\}$. It easily follows that $Paths^{\star}_{\min}(s, \phi)$ is prefix containment-free and that, for any state s:

$$Prob(s, \Phi \mathrm{U}^{\leq h}\ \Psi) = \mathbb{P}\big(Paths^{\star}_{\min}(s, \Phi \mathrm{U}^{\leq h}\ \Psi)\big).$$

In the rest of this paper, we explore counterexamples for formulas of the form $\mathcal{P}_{\leq p}(\Phi \mathrm{U}^{\leq h} \Psi)$ with $p \neq 0, 1$, i.e., with probability upper bounds. In Section 7, we extend our results to formulas with probability lower bounds and deal with qualitative bounds (i.e., $p = 0, 1$). Section 8 shows how our results can be used to show the NP-completeness of shortest counterexamples for DTMCs with rewards.

3 Evidences and Counterexamples

Let us first consider what a counterexample in our setting actually is. To that end, consider the PCTL formula $\mathcal{P}_{\leq p}(\phi)$, where $p \in (0, 1)$, and let $\phi = \Phi \mathrm{U}^{\leq h} \Psi$ for the rest of the paper. It follows that:

$$\begin{array}{ll} & s \not\models \mathcal{P}_{\leq p}(\phi) \\ \text{iff} & \text{not } (Prob(s, \phi) \leq p) \\ \text{iff} & Prob(s, \phi) > p \\ \text{iff} & \mathbb{P}\big(Paths^{\star}_{\min}(s, \phi)\big) > p. \end{array}$$

So, $\mathcal{P}_{\leq p}(\phi)$ is refuted by state s whenever the total probability mass of all ϕ-paths that start in s exceeds p. Even for unbounded until-formulas, the validity can be shown by finite paths as only paths that end in a Ψ-state contribute to $Paths^{\star}_{\min}(s, \phi)$. This indicates that a counterexample for $s \not\models \mathcal{P}_{\leq p}(\phi)$ is a *set of finite paths* starting in s and minimally satisfying ϕ. Any finite path that contributes to the violation is called an *evidence*.

Definition 3 (Evidence). *An* evidence *for violating $\mathcal{P}_{\leq p}(\phi)$ in state s is a finite path $\sigma \in Paths^{\star}_{\min}(s, \phi)$.*

The contribution of each evidence is characterized by its probability. Thus, an evidence with the largest contribution is defined.

Definition 4 (Strongest evidence). *For a* strongest *evidence σ and any evidence σ', it holds that: $\mathbb{P}(\sigma) \geq \mathbb{P}(\sigma')$.*

Dually, a strongest evidence for violating $\mathcal{P}_{\leq p}(\phi)$ is a strongest witness for fulfilling $\mathcal{P}_{>p}(\phi)$. Evidently, a strongest evidence is not necessarily a counterexample as its probability mass may be (far) below p. We thus define a counterexample as follows:

Definition 5 (Counterexample). *A* counterexample *for $\mathcal{P}_{\leq p}(\phi)$ in state s is a set C of evidences such that $C \subseteq Paths^{\star}_{\min}(s, \phi)$ and $\mathbb{P}(C) > p$.*

A counterexample for state s is thus a set of evidences that all start in s. We will, at the moment, not dwell further upon how to represent this set and assume an abstract representation as a set suffices; a compact representation will be proposed in Section 10. Note that the measurability of counterexamples is ensured by the fact that $C \subseteq Paths^{\star}_{\min}(s, \phi)$ is prefix containment-free; hence, $\mathbb{P}(C)$ is well-defined. Let $CX_p(s, \phi)$ denote the set of all counterexamples for $\mathcal{P}_{\leq p}(\phi)$ in state s. For $C \in CX_p(s, \phi)$ and Cs superset C': $C \subseteq C' \subseteq Paths^{\star}_{\min}(s, \phi)$, it follows that $C' \in CX_p(s, \phi)$ since $\mathbb{P}(C') \geq \mathbb{P}(C) > p$. That is to say, any extension of a counterexample C with paths in $Paths^{\star}_{\min}(s, \phi)$ is a counterexample. This motivates the notion of minimality.

Definition 6 (Minimal counterexample). *$C \in CX_p(s, \phi)$ is a* minimal counterexample *if $|C| \leq |C'|$, for any $C' \in CX_p(s, \phi)$.*

As in conventional model checking, we are not interested in generating arbitrary counterexamples, but those that are easy to comprehend and provide clear evidence of the refutation of the formula. So, akin to shortest counterexamples for linear-time logics, we consider the notion of a smallest counterexample. Such counterexamples are required to be succinct, i.e., minimal, allowing easier analysis of the cause of refutation, and most distinctive, i.e., their probability should exceed p more than all other minimal counterexamples. This motivates the following definition:

Definition 7 (Smallest counterexample). *$C \in CX_p(s, \phi)$ is a* smallest counterexample *if it is minimal and $\mathbb{P}(C) \geq \mathbb{P}(C')$ for any minimal counterexample $C' \in CX_p(s, \phi)$.*

The intuition is that a smallest counterexample is the one that deviates most from the required probability bound given that it has the smallest number of paths. Thus, there does not exist an equally sized counterexample that deviates more from p. Strongest evidences, minimal counterexamples, or smallest counterexamples may not be unique, as different paths may have equal probability. As a result, not every strongest evidence is contained in a minimal (or smallest) counterexample. Whereas minimal counterexamples may not contain any strongest evidence, any smallest counterexample contains at least one strongest evidence. Using standard mathematical results we obtain:

Lemma 1. *A finite counterexample for $s \not\models \mathcal{P}_{\leq p}(\phi)$ exists.*

Proof. By contradiction. Assume there are only infinite counterexamples for $s \not\models \mathcal{P}_{\leq p}(\phi)$. Let $C = \{\sigma_1, \sigma_2, \ldots\}$ be one such counterexample, i.e.,

$$\underbrace{\sum_{i=1}^{\infty} \mathbb{P}(\sigma_i)}_{=L} \;=\; \lim_{j \to \infty} \underbrace{\sum_{i=1}^{j} \mathbb{P}(\sigma_i)}_{a_j} \;>\; p.$$

Note that, since all $\mathbb{P}(\sigma_i)$ are positive, the order of summation is irrelevant for the limit. By definition of limit, this means that

$$\forall \epsilon > 0.\ \exists N_\epsilon \in \mathbb{N}.\ \forall n \geq N_\epsilon.\ |a_n - L| \;<\; \epsilon. \qquad (1)$$

Take ϵ such that $0 < \epsilon < L - p$. By (1), for some $n \geq N_\epsilon$, $|a_n - L| < L - p$, i.e., $a_n > p$. But then, the finite set $C' = \{\sigma_1, \ldots, \sigma_n\}$ is also a counterexample as $\mathbb{P}(C') > p$. Contradiction. □

From this lemma, it directly follows that a smallest counterexample for $s \not\models \mathcal{P}_{\leq p}(\phi)$ is finite.

Remark 1 (Finiteness). For until-formulas with *strict* upper bounds, i.e., $\mathcal{P}_{<p}(\phi)$, a finite counterexample may not exist. This occurs when, e.g., the only counterexample is an infinite set C of finite paths with $\mathbb{P}(C) = p$. The limit of the sum of the path probabilities (obeying a geometric distribution) equals p, but infinitely many paths are

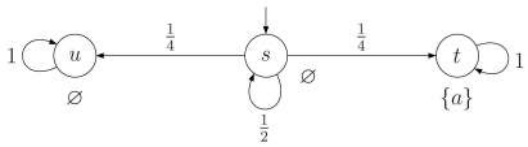

Fig. 2. A DTMC with infinite counterexample for $P_{<\frac{1}{2}}(\Diamond a)$.

needed to reach p. For instance, consider the DTMC in Fig. 2. The violation of $\mathcal{P}_{<\frac{1}{2}}(\Diamond\, a)$ in state s can only be shown by an infinite set of paths, viz. all paths that traverse the self-loop at state s arbitrarily often reach state t.

Example 3. Consider the DTMC in Fig. 1, for which s violates $\mathcal{P}_{\leq\frac{1}{2}}(a\mathrm{U}b)$. Evidences are, among others, $\sigma_1 = s{\cdot}s_1{\cdot}t_1$, $\sigma_2 = s{\cdot}s_1{\cdot}s_2{\cdot}t_1$, $\sigma_3 = s{\cdot}s_2{\cdot}t_1$, $\sigma_4 = s{\cdot}s_1{\cdot}s_2{\cdot}t_2$, and $\sigma_5 = s{\cdot}s_2{\cdot}t_2$. Their respective probabilities are 0.2, 0.2, 0.15, 0.12 and 0.09. $\sigma = s{\cdot}s_1{\cdot}t_1{\cdot}t_2$ is not an evidence as it contains a proper prefix, $s{\cdot}s_1{\cdot}t_1$, that satisfies $a\mathrm{U}b$.

Paths σ_1 and σ_2 are strongest evidences. The set $C_1 = \{\sigma_1, \ldots, \sigma_5\}$ with $\mathbb{P}(C_1) = 0.76$ is a counterexample, but not a minimal one, as the removal of either σ_1 or σ_2 also yields a counterexample. $C_2 = \{\sigma_1, \sigma_2, \sigma_4\}$ is a minimal but not a smallest counterexample, as $C_3 = \{\sigma_1, \sigma_2, \sigma_3\}$ is minimal too with $\mathbb{P}(C_3) = 0.56 > 0.52 = \mathbb{P}(C_2)$. C_3 is a smallest counterexample.

In the remainder of the paper, we consider the strongest evidence problem (SE) that, for a given state s with $s \not\models \mathcal{P}_{\leq p}(\phi)$, determines the strongest evidence for this violation. Subsequently, we consider the corresponding smallest counterexample problem (SC).

4 Reduction to Graph Theory

Prior to finding strongest evidences or smallest counterexamples, we modify the DTMC and turn it into a weighted digraph. This enables us, as we will show, to exploit well-known efficient graph algorithms to the SE and SC problem. Let $Sat(\Phi) = \{s \in S \mid s \models \Phi\}$ for any Φ. Due to the bottom-up traversal of the model-checking algorithm over the formula $\phi = \Phi\mathrm{U}^{\leq h}\Psi$, we may assume that $Sat(\Phi)$ and $Sat(\Psi)$ are known.

4.1 Step 1: Adapting the DTMC

First, we make all states in the DTMC $\mathcal{D} = (S, \mathbf{P}, L)$ that neither satisfy Φ nor Ψ absorbing. Then we add an extra state t so that all outgoing transitions from a Ψ-state are replaced by a transition to t with probability 1. State t can thus only be reached via a Ψ-state. The obtained DTMC $\mathcal{D}' = (S', \mathbf{P}', L')$ has state space $S \cup \{t\}$ for $t \notin S$. The stochastic matrix $\mathbf{P}'$ is defined as follows:

$$\mathbf{P}'(s,t) = \begin{cases} 1 & \text{if } s \in \mathrm{Sat}(\Psi) \text{ or } s = t \\ 0 & \text{o.w.} \end{cases}$$

and, for $s, s' \neq t$,

$$\mathbf{P}'(s,s') = \begin{cases} 1 & \text{if } s \in \mathrm{Sat}(\neg\Phi \wedge \neg\Psi) \text{ and } s = s' \\ \mathbf{P}(s,s') & \text{if } s \in \mathrm{Sat}(\Phi \wedge \neg\Psi) \\ 0 & \text{o.w.} \end{cases}$$

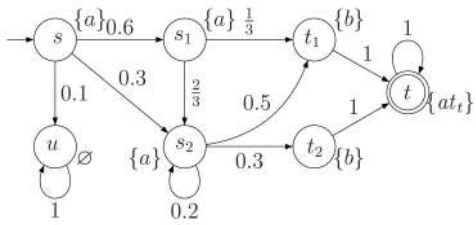

Fig. 3. Transformation from a DTMC to a weighted digraph: Step 1.

$L'(s) = L(s)$ for $s \in S$ and $L'(t) = \{at_t\}$, where $at_t \notin L(s')$ for any $s' \in S$, i.e., at_t uniquely identifies being at state t. Remark that all the $(\neg\Phi \wedge \neg\Psi)$ states could be collapsed into a single state, but this is not further explored here. The time complexity of this transformation is $\mathcal{O}(n)$, where $n = |S|$. It is evident that the validity of $\Phi\mathrm{U}^{\leq h}\Psi$ is not affected by this amendment of the DTMC. By construction, any finite path $\sigma' = \sigma{\cdot}t$ in $\mathcal{D}'$ with $0 < |\sigma'| \leq h+1$ satisfies $(\Phi \vee \Psi)\mathrm{U}^{\leq h+1}at_t$ and the prefix σ in $\mathcal{D}$ satisfies $\Phi\mathrm{U}^{\leq h}\Psi$, where σ' and σ are equally probable.

Example 4. Applying the above transformation to the DTMC $\mathcal{D}$ in Fig. 1 and path formula $a\mathrm{U}b$ yields the DTMC $\mathcal{D}'$ illustrated in Fig. 3. The $(\neg a \wedge \neg b)$ state u is made absorbing and both b-states (i.e., t_1 and t_2) are equipped with a transition with probability 1 to the new absorbing state t (indicated by a double circle).

4.2 Step 2: Conversion into a Weighted Digraph

As a second preprocessing step, the DTMC obtained in the first step is transformed into a weighted digraph, i.e., a triple $\mathcal{G} = (V, E, w)$, where V is a finite set of vertices, $E \subseteq V \times V$ is a set of edges, and $w : E \to \mathbb{R}_{\geq 0}$ is a weight function.

Definition 8 (Weighted digraph of a DTMC). *For DTMC $\mathcal{D} = (S, \mathbf{P}, L)$, the weighted digraph $\mathcal{G}_\mathcal{D} = (V, E, w)$, where $V = S$, $(v, v') \in E$ iff $\mathbf{P}(v, v') > 0$, and $w(v, v') = -\log \mathbf{P}(v, v')$.*

The edge weights are obtained by taking the negation of the logarithm of the corresponding transition probabilities. Note that $w(s, s') \in [0, \infty)$ if $\mathbf{P}(s, s') > 0$. Thus, we indeed obtain a digraph with nonnegative weights. This transformation can be done in $\mathcal{O}(m)$, where m is the number of nonzero elements in $\mathbf{P}$. We often omit the self-loop on vertex t in $\mathcal{G}_\mathcal{D}$, as it has weight 0.

Example 5 (Continuing Example 4). Applying this transformation to the DTMC $\mathcal{D}'$ in Fig. 3 yields the weighted digraph in Fig. 4.

A path σ from s to t in the digraph $\mathcal{G}$ is a sequence $\sigma = v_0{\cdot}v_1{\cdots}v_j \in V^+$, where $v_0 = s, v_j = t$ and $(v_i, v_{i+1}) \in E$, for $0 \leq i < |\sigma|$. As for paths in DTMCs, $|\sigma|$ denotes the length of σ. The *weight* of finite path $\sigma = v_0{\cdot}v_1{\cdots}v_j$ in graph $\mathcal{G}$ is $w(\sigma) = \sum_{i=0}^{j-1} w(v_i, v_{i+1})$. Path weights in $\mathcal{G}$ and path probabilities in DTMC $\mathcal{D}$ are related as follows:

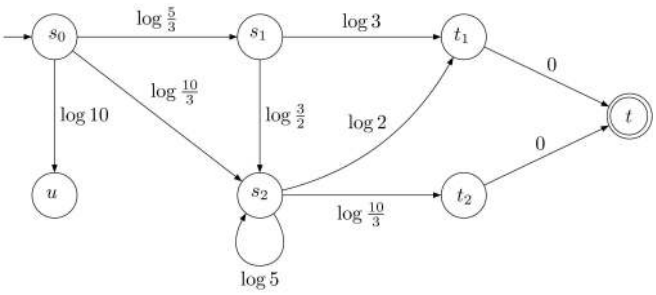

Fig. 4. Transformation from a DTMC to a weighted digraph: Step 2.

$$\begin{aligned} w(\sigma) &= \sum_{i=0}^{j-1} w(v_i, v_{i+1}) \;=\; \sum_{i=0}^{j-1} -\log \mathbf{P}(v_i, v_{i+1}) \\ &= -\sum_{i=0}^{j-1} \log \mathbf{P}(v_i, v_{i+1}) \;=\; -\log \prod_{i=0}^{j-1} \mathbf{P}(v_i, v_{i+1}) \\ &= -\log \mathbb{P}(\sigma). \end{aligned}$$

Now the multiplication of probabilities in $\mathcal{D}$ corresponds to addition of weights in $\mathcal{G}_{\mathcal{D}}$ and the next two lemmas directly follow:

Lemma 2. *Let σ and σ' be finite paths in DTMC $\mathcal{D}$ and its graph $\mathcal{G}_{\mathcal{D}}$. Then, $\mathbb{P}(\sigma') \geq \mathbb{P}(\sigma)$ iff $w(\sigma') \leq w(\sigma)$.*

This result implies that the most probable path between two states in DTMC $\mathcal{D}$ equals the shortest path (i.e., the path with the least weight) between these states in the weighted digraph $\mathcal{G}_{\mathcal{D}}$. It is easy to see that this result can be generalized to paths of a certain length (or, equivalently, number of *hops*), and to the second, third, etc., most probable paths. This yields:

Lemma 3. *For any path σ with $|\sigma| = h$ from s to t in DTMC $\mathcal{D}$, $k \in \mathbb{N}_{>0}$: σ is a kth most probable path of h hops in $\mathcal{D}$ iff σ is a kth shortest path of h hops in $\mathcal{G}_{\mathcal{D}}$.*

A path σ is a kth shortest path if, whenever all paths (between the same states as σ) are ranked in a descending order w.r.t. their weights, σ is at the kth position. Note that such rankings are not necessarily unique (as paths may have equal weights) and, so, a kth shortest path may not be unique. The kth most probable path is defined in a similar way. This lemma provides the basis for the remaining algorithms in the following sections.

5 Finding Strongest Evidences

5.1 Unbounded Until

Based on Lemma 3 with $k = 1$ and $h = \infty$, we consider the well-known shortest path problem:

Definition 9 (SP problem). *Given a weighted digraph $\mathcal{G} = (V, E, w)$ and $s, t \in V$, the* shortest path (SP) *problem is to determine a path σ from s to t such that $w(\sigma) \leq w(\sigma')$ for any path σ' from s to t in $\mathcal{G}$.*

From Lemma 3, together with the transformation of a DTMC into a weighted digraph, it follows that there is a polynomial reduction from the SE problem for unbounded until to the SP problem. As the SP problem is in PTIME, it follows:

Theorem 4. *The SE problem for unbounded until is in PTIME.*

Various efficient algorithms [24], [12], [18] exist for the SP problem, e.g., when using Dijkstra's algorithm, the SE problem for unbounded until can be solved in time $\mathcal{O}(m + n \log n)$, where $m = |E|$ and $n = |V|$, provided appropriate data structures such as Fibonacci heaps are used.

5.2 Bounded Until

Lemma 3 for $k = 1$ and $h \in \mathbb{N}_{\geq 0}$ suggests considering the hop-constrained SP problem.

Definition 10 (HSP problem). *Given a weighted digraph $\mathcal{G} = (V, E, w)$, $s, t \in V$, and $h \in \mathbb{N}_{\geq 0}$, the* hop-constrained SP (HSP) *problem is to determine a path σ in $\mathcal{G}$ from s to t with $|\sigma| \leq h$ such that $w(\sigma) \leq w(\sigma')$ for any path σ' from s to t with $|\sigma'| \leq h$.*

The HSP problem is a special case of the (resource) constrained shortest path (CSP) problem [54], [2], where the only constraint is the hop count. Besides the weight w on each edge, it may consume other resources $w_1, \ldots, w_c$ and the sum of each resource should be bounded by the resource constraints $\lambda_1, \ldots, \lambda_c$, where c is the number of resources. Weighted digraphs with multiple resources are obtained by allowing multiple weights to edges.

Definition 11 (CSP problem). *Let $\mathcal{G}$ be a multiweighted digraph $(V, E, \{w\} \cup \{w_1, \ldots, w_c\})$ with $s, t \in V$ and resource constraints λ_i, for $1 \leq i \leq c$. Edge $e \in E$ uses $w_i(e) \geq 0$ units of resource i. The* (resource) constrained SP (CSP) *problem is to determine a shortest path σ w.r.t. the weight $\sum_{e \in \sigma} w(\sigma)$ in $\mathcal{G}$ from s to t such that $\sum_{e \in \sigma} w_i(e) \leq \lambda_i$ for $1 \leq i \leq c$.*

The CSP problem is NP-complete, even for a single resource constraint [2]. However, if each edge uses a constant unit of that resource (such as the hop count), the CSP problem can be solved in polynomial time, cf. [27, problem [ND30].

Theorem 5. *The SE problem for bounded until is in* PTIME.

For $h \geq n - 1$, it is possible to use Dijkstra's SP algorithm (as for unbounded until) as a shortest path does not contain cycles. If $h < n - 1$, however, Dijkstra's algorithm does not guarantee to obtain a shortest path of at most h hops. We therefore adopt the Bellman-Ford (BF) algorithm [12], [18], which fits well to our problem as it proceeds by increasing hop count. It can be readily modified to generate a shortest path within a given hop count. In the remainder of the paper, this algorithm is generalized for computing smallest counterexamples. The BF algorithm is based on a set of recursive equations; we extend it with the hop count h. For $v \in V$, let $\pi_{\leq h}(s, v)$ denote the shortest path from s to v of at most h hops (if it exists). Then:

$$\pi_{\leq h}(s, v) = \begin{cases} s & \text{if } v = s \text{ and } h \geq 0 \\ \bot & \text{if } v \neq s \text{ and } h = 0 \\ \arg\min_u \{ w(\pi_{\leq h-1}(s, u) \cdot v) \mid (u, v) \in E \} & \text{o.w.} \end{cases}$$

where $\bot$ denotes the nonexistence of a such a path.[1] The last clause states that $\pi_{\leq h}(s, v)$ consists of the shortest path to v's direct predecessor u, i.e., $\pi_{\leq h-1}(s, u)$, extended with edge

1. Where $\bot \cdot v = v$ for any v, $\{\bot\} = \emptyset$, and $\arg\min \emptyset = \bot$.

(u,v). Note that $\min_u\{w(\pi_{\leq h-1}(s,u)\cdot v) \mid (u,v)\in E\}$ is the weight of a shortest path; by means of $\arg$, such a shortest path is obtained. It follows (cf. [49]) that $\pi_{\leq h}(s,v)$ characterizes the shortest path from s to v in at most h hops and can be solved in time $\mathcal{O}(hm)$. As $h<n-1$, this is indeed in PTIME. Recall that, for $h\geq n-1$, Dijkstra's algorithm has a favorable time complexity.

Remark 2 (Exploiting the Viterbi algorithm). An alternative to the BF algorithm is to adopt the *Viterbi algorithm* [40], [61], [60]. In fact, to apply this algorithm, the transformation into a weighted digraph is not needed. The Viterbi algorithm is based on dynamic programming and aims to find the most likely sequence of hidden states (i.e., a finite path) that result in a sequence of observed events (a trace). It is used in the context of hidden Markov models, which are used in, e.g., speech recognition, and bioinformatics. Let DTMC $\mathcal{D}$ be obtained after the first step described in Section 4, and suppose that $L(s)$ is extended with all subformulas of the formula under consideration that hold in s. (Note that these labels are known due to the recursive descent nature of the PCTL model-checking algorithm.) Let $tr(\sigma)$ denote the projection of a path $\sigma=s_0\cdot s_1\cdots s_h$ on its trace, i.e., $tr(\sigma)=L(s_0)L(s_1)\cdots L(s_h)$. Recall that $\sigma{\downarrow_i}$ denotes the prefix of path σ truncated at length i (thus ending in s_i), thus $tr(\sigma{\downarrow_i})=L(s_0)L(s_1)\cdots L(s_i)$. $\gamma{\downarrow_i}$ denotes the prefix of trace γ with length $i+1$. Note that the length of a trace is one more than the length of the corresponding path. Let $\varrho(\gamma,i,v)$ denote the probability of the most probable path $\sigma{\downarrow_i}$ whose trace equals $\gamma{\downarrow_i}$ and reaches state v. Formally,

$$\varrho(\gamma,i,v)=\max_{tr(\sigma{\downarrow_i})=\gamma_i\wedge\sigma\in Paths^{s}(s_0)}\prod_{j=0}^{i-1}\mathbf{P}(s_j,s_{j+1})\cdot\mathbf{1}_v(s_i),$$

where $\mathbf{1}_v(s_i)$ is the characteristic function of v, i.e., $\mathbf{1}_v(s_i)=1$ iff $s_i=v$. The Viterbi algorithm provides an algorithmic solution to compute $\varrho(\gamma,i,v)$:

$$\varrho(\gamma,i,v)=\begin{cases}1 & \text{if} s=v \text{ and } i=0\\ 0 & \text{if} s\neq v \text{ and } i=0\\ \max_{u\in S}\{\varrho(\gamma,i-1,u)\cdot\mathbf{P}(u,v)\} & \text{o.w.}\end{cases}$$

By computing $\varrho(\Phi^h\Psi,h,s_h)$, the Viterbi algorithm determines the most probable path $\sigma=s_0\cdot s_1\cdots s_h$ that generates the trace $\gamma=L'(s_0)L'(s_1)\cdots L'(s_h)=\Phi^h\Psi$ with length $h+1$. Here, $L'(s)=L(s)\cap\{\Phi,\Psi\}$, i.e., L' is the labeling restricted to the subformulas Φ and Ψ. For the SE problem for bounded until, the trace of the most probable hop-constrained path from s to t is among $\{\Psi at_t,\Phi\Psi at_t,\ldots,\Phi^h\Psi at_t\}$. The self-loop at vertex t with probability 1 ensures that all these paths have length $h+1$ while not changing their probabilities. For instance, the path with trace $\Phi^i\Psi at_t$ can be extended so that the trace becomes $\Phi^i\Psi {at_t}^{h+1-i}$, where $i\leq h$. Since the DTMC is already transformed as in Step 1 (cf. Section 4.1), we can obtain the most probable path for $\Phi\mathrm{U}^{\leq h}\Psi$ by computing $\varrho((\Phi\vee\Psi\vee at_t)^{h+1}at_t,h{+}1,t)$ using the Viterbi algorithm. The time complexity is $\mathcal{O}(hm)$, as for the BF algorithm.

6 Finding Smallest Counterexamples

Recall that a smallest counterexample is a minimal counterexample, whose probability, among all minimal counterexamples, deviates maximally from the required probability bound. In this section, we investigate algorithms and their time and space complexity for computing smallest counterexamples.

6.1 Unbounded Until

Lemma 3 is applicable here for $k>1$ and $h=\infty$. This suggests considering the k shortest paths problem.

Definition 12 (KSP problem). *Given a weighted digraph $\mathcal{G}=(V,E,w)$, $s,t\in V$, and $k\in\mathbb{N}_{>0}$, the* k shortest paths (KSP) *problem is to find k distinct paths $\sigma^1,\ldots,\sigma^k$ between s and t in $\mathcal{G}$ (if such paths exist) such that 1) for $1\leq i<j\leq k$, $w(\sigma^i)\leq w(\sigma^j)$ and 2) for every σ between s and t, if $\sigma\notin\{\sigma^1,\ldots,\sigma^k\}$, then $w(\sigma)\geq w(\sigma^k)$.*

Note that σ^i denotes the ith shortest path and, for $i\neq j$, it is possible that $w(\sigma^i)=w(\sigma^j)$. Stated in words, the ith shortest path is not necessarily "strictly shorter" than the jth one, for $i<j$.

Theorem 6. *The SC problem for unbounded until is a* KSP *problem.*

Proof. We prove by contraposition that a smallest counterexample of size k, contains k most probable paths. Let C be a smallest counterexample for ϕ with $|C|=k$ and assume C does not contain the k most probable paths satisfying ϕ. Then, there is a path $\sigma\notin C$ satisfying ϕ such that $\mathbb{P}(\sigma)>\mathbb{P}(\sigma')$ for some $\sigma'\in C$. Let $C'=C\setminus\{\sigma'\}\cup\{\sigma\}$. Then, C' is a counterexample for ϕ, $|C|=|C'|$ and $\mathbb{P}(C)>\mathbb{P}(C')$. This contradicts C being a smallest counterexample. □

The question remains how to obtain k. Various algorithms for the KSP problem require k to be known a priori. This is inapplicable in our setting as the number of paths in a smallest counterexample is not known in advance. We therefore consider algorithms that allow to determine k *on the fly*, i.e., that can halt at any k and resume if necessary. A good candidate is Eppstein's algorithm [26]. Although this algorithm has the best known asymptotic time complexity, viz. $\mathcal{O}(m+n\log n+k)$, in practice, the recursive enumeration algorithm (REA) by Jiménez and Marzal [38] prevails. This algorithm has a time complexity in $\mathcal{O}(m+kn\log\frac{m}{n})$ and is based on a generalization of the recursive equations for the BF algorithm. Besides, it is readily adaptable to the case for bounded h, as we demonstrate below. Note that the time complexity of all known KSP algorithms depend on k and, as k can be exponential in the size of the digraph, their complexity is *pseudopolynomial*.

6.2 Bounded Until

Similar to strongest evidences for bounded until, we now consider the KSP problem with constrained path lengths.

Definition 13 (HKSP problem). *Given a weighted digraph $\mathcal{G}=(V,E,w)$, $s,t\in V$, $h\in\mathbb{N}_{\geq 0}$, and $k\in\mathbb{N}_{>0}$, the* hop-constrained KSP (HKSP) *problem is to determine k shortest paths each of length at most h between s and t.*

Theorem 7. *The SC problem for bounded until is an* HKSP *problem.*

To our knowledge, algorithms for the HKSP problem do not exist. In order to solve the HKSP problem, we propose adapting Jiménez and Marzal's REA algorithm [38]. The advantage of this algorithm is that k can be determined on the fly, an essential characteristic for our setting. For $v \in V$, let $\pi^k_{\leq h}(s,v)$ denote the kth shortest path from s to v of length at most h (if it exists). As before, we use $\perp$ to denote the nonexistence of a path. We establish:

$$\pi^k_{\leq h}(s,v) = \begin{cases} s & \text{if } k=1, v=s \text{ and } h \geq 0 \\ \perp & \text{if } h=v \text{ and } (v \neq s \text{ or } v=s \ \wedge \ k{>}1) \\ \arg\min_\sigma \{ w(\sigma) \mid \sigma \in Q^k_{\leq h}(s,v) \} & \text{o.w.,} \end{cases} \tag{2}$$

where $Q^k_{\leq h}(s,v)$ is defined by:

$$\begin{cases} \{\pi^1_{\leq h-1}(s,u')\cdot v \mid (u',v) \in E\} \\ \qquad \text{if } k=1, v \neq s, h>0 \text{ or } k=2, v=s, h>0, \\ \big(Q^{k-1}_{\leq h}(s,v) - \{\pi^{k'}_{\leq h-1}(s,u)\cdot v\}\big) \cup \{\pi^{k'+1}_{\leq h-1}(s,u)\cdot v\} \\ \text{if } k>1, h>0, \text{ and } \exists u,k'.\big(\pi^{k-1}_{\leq h}(s,v) = \pi^{k'}_{\leq h-1}(s,u)\cdot v\big), \\ \emptyset \qquad \text{o.w.} \end{cases} \tag{3}$$

Let us explain these equations. The kth shortest path of length h is chosen from a set $Q^k_{\leq h}(s,v)$ of "candidate" paths. This principle is identical to that in the Bellman-Ford equations given earlier. In particular, if this set contains several shortest paths, a nondeterministic selection is made. The main difference with the BF equations is the more complex definition of the set of candidate paths. The first clause of $Q^k_{\leq h}(s,v)$ is self-explanatory. Let $k>1$, $h>0$, and $v \neq s$. By the inductive nature, the set $Q^{k-1}_{\leq h}(s,v)$ is at our disposal. Assume that the path $\pi^{k-1}_{\leq h}(s,v)$ has the form $s\cdots u\cdot v$, where prefix $s\cdots u$ is the k'th shortest path between s and u (for some k') of at most $h-1$ hops, i.e., $s\cdots u$ equals $\pi^{k'}_{\leq h-1}(s,u)$. Then, $Q^k_{\leq h}(s,v)$ is obtained from $Q^{k-1}_{\leq h}(s,v)$ by replacing the path $s\cdots u\cdot v$ (as it has just been selected) by the path $\pi^{k'+1}_{\leq h-1}(s,u)\cdot v$, if this exists. Thus, as a result of the removal of a $(k-1)$th shortest path which reaches v via u, say, the set of candidate paths is updated with the next shortest path from s to v that goes via u. If such path does not exist (i.e., equals $\perp$), then the candidate set is not extended (as $\{\perp\} = \emptyset$). In case there is no k' such that $\pi^{k-1}_{\leq h}(s,v)$ can be decomposed into a k'th shortest path between s and some direct predecessor u of v, it means that $Q^{k-1}_{\leq h}(s,v)$ is empty, and we return the empty set (last clause).

Lemma 8. *Equations (2) and (3) characterize the hop-constrained k shortest paths from s to v in at most h hops.*

Proof. This proof goes along similar lines as [38]. Let $\mathcal{X}^k_{\leq h}(s,v)$ denote the set of k shortest paths from s to v in at most h hops. Each path in $\mathcal{X}^k_{\leq h}(s,v)$ reaches v from some vertex $u \in Pred(v) = \{v \in V \mid (v,v) \in E\}$. In order to compute $\pi^k_{\leq h}(s,v)$, we should consider, for every $u \in Pred(v)$, *all* paths from s to u that do not yield a path in $\mathcal{X}^{k-1}_{\leq h}(s,v)$. However, since $k_1 < k_2$ implies that $w\big(\pi^{k_1}_{\leq h-1}(s,u)\big) + w(u,v) \leq w\big(\pi^{k_2}_{\leq h-1}(s,u)\big) + w(u,v)$, only *the shortest* of these paths needs to be taken into account when computing $\pi^k_{\leq h}(s,v)$. Thus, we can associate to (v,h) a set of candidate paths $Q^k_{\leq h}(s,v)$ among which $\pi^k_{\leq h}(s,v)$ can be chosen, that contains at most one path for each predecessor $u \in Pred(v)$. This set $Q^k_{\leq h}(s,v)$ is recursively defined by (3). □

6.3 Adapted Recursive Enumeration Algorithm

Equations (2) and (3) provide the basis for the adapted REA for the HKSP problem. In the main program (Algorithm 1), first the shortest path from s to t is determined using, e.g., BF. Then, the k shortest paths are determined iteratively using the subroutine $NextPath$ (Algorithm 2). The computation terminates when the total probability mass of the k shortest paths so far exceeds the bound p (Algorithm 1, line 4). Recall that p is the upper probability bound of the PCTL formula to be checked. Note that $Q[v,h,k]$ in the algorithm corresponds to $Q^k_{\leq h}(s,v)$. The paths in the priority queue $Q[v,h,k]$ are ordered w.r.t. their weights. When $k=1$, $Q[v,h,k-1]$ and $\pi^{k-1}_{\leq h}(s,v)$ do not exist and are $\emptyset$ and $\perp$, respectively. $Q[v,h,k]$ is constructed explicitly in two cases (Algorithm 2, lines 4-5) and inherits from $Q[v,h,k-1]$ for the remaining cases (line 12). In the latter case, σ' is the path $\sigma = \pi^{k-1}_{\leq h}(s,v)$ without the last state v, i.e., $\sigma = \sigma'\cdot v$; u is the last state on σ', or equivalently, the predecessor state of v on σ with $\sigma = s\cdots u\cdot v$ and σ' is the k'th shortest path from s to u within $h-1$ hops, i.e., $\sigma' = \pi^{k'}_{\leq h-1}(s,u)$. In other words, the function $index\,(s\cdots u, h{-}1)$ returns k', where $s\cdots u$ is the k'th shortest s-u path within $h-1$ hops. The set $Q^k_{\leq h}(s,v)$ is updated according to (3) (Algorithm 2, lines 6-13). In line 14, $\pi^k_{\leq h}(s,v)$ is selected from $Q^k_{\leq h}(s,v)$ according to the third clause in (2).

Time complexity. Before we analyze the time complexity of the algorithm, we first prove that the recursive calls to $NextPath$ to compute the $\pi^k_{\leq h}(s,t)$ visit in the worst case all the vertices in $\pi^{k-1}_{\leq h}(s,t)$, which is at most h.

Algorithm 1 Hop-constrained k shortest paths
Require: weighted digraph $\mathcal{G}$, states s,t, $h \in \mathbb{N}_{\geq 0}$, $p \in [0,1]$
Ensure: $C = \{\pi^1_{\leq h}(s,t), \ldots, \pi^k_{\leq h}(s,t)\}$ with $\mathbb{P}(C) > p$
1: compute $\pi^1_{\leq h}(s,t)$ by BF;
2: $k := 1$;
3: $pr := \mathbb{P}(\pi^1_{\leq h}(s,t))$;
4: **while** $pr \leq p$ **do**
5: $\quad k := k+1$;
6: $\quad \pi^k_{\leq h}(s,t) := NextPath(t,h,k)$;
7: $\quad pr := pr + \mathbb{P}(\pi^k_{\leq h}(s,t))$;
8: **end while**;
9: **return** $\pi^1_{\leq h}(s,t), \ldots, \pi^k_{\leq h}(s,t)$;

Algorithm 2 $NextPath(v,h,k)$
Require: weighted digraph $\mathcal{G}$, $\pi^{k-1}_{\leq h}(s,v)$ (if it exists), and candidate path set $Q[v,h,k{-}1]$ (if it exists)
Ensure: $\pi^k_{\leq h}(s,v)$
1: PriorityQueue $Q[v,h,k]$;
2: **if** $k=1, v=s, h \geq 0$ **then return** s;
3: **if** $(h=0) \wedge ((k>1 \wedge v=s) \vee (v \neq s))$ **then return** $\perp$;
4: **if** $(k=1, v \neq s, h>0) \vee (k=2, v=s, h>0)$ **then**

5: $Q[v,h,k] := \{\pi^{1}_{\leq h-1}(s,u')\cdot v \mid (u',v) \in E\}$;
6: **else**
7: Path $\sigma' := \pi^{k-1}_{\leq h}(s,v) \setminus \{v\}$;
8: State $u := last(\sigma')$;
9: Int $k' := index(\sigma', h-1)$;
10: **if** $\pi^{k'+1}_{\leq h-1}(s,u)$ is not computed yet **then**
11: $\pi^{k'+1}_{\leq h-1}(s,u) := NextPath(u, h-1, k'+1)$;
12: $Q[v,h,k] := Q[v,h,k-1]$;
13: $Q[v,h,k].enqueue\big(\pi^{k'+1}_{\leq h-1}(s,u)\cdot v\big)$;
14: **return** $Q[v,h,k].dequeue()$;

Lemma 9. *Let $k > 1$ and $v \in V$. If $NextPath(v,h,k)$ calls $NextPath(u, h-1, j)$, then vertex u occurs in $\pi^{k-1}_{\leq h}(s,v)$.*

Proof. Consider $NextPath(v,h,k)$ and let $\pi^{k-1}_{\leq h}(s,v) = u_1 \cdots u_\ell$ with $u_1 = s$ and $u_\ell = v$. Let k_i be the index such that $\pi^{k_i}_{\leq h-1}(s,u_i) = u_1 \cdots u_i$, for $0 < i \leq \ell$. As $\pi^{k-1}_{\leq h}(s,v) = \pi^{k_{\ell-1}}_{\leq h-1}(s,u_{\ell-1})\cdot v$, $NextPath(v,h,k)$ needs to recursively invoke $NextPath(u_{\ell-1}, h-1, k_{\ell-1}+1)$ in case the path $\pi^{k_{\ell-1}+1}_{\leq h-1}(s,u_{\ell-1})$ has not been computed yet. By a similar reasoning, the path $\pi^{k_{\ell-1}}_{\leq h-1}(s,u_{\ell-1})$ is of the form $\pi^{k_{\ell-2}}_{\leq h-2}(s,u_{\ell-2})\cdot u_{\ell-1}$, and $NextPath(u_{\ell-1}, h-1, k_{\ell-1}+1)$ may need to invoke $NextPath(u_{\ell-2}, h-2, k_{\ell-2}+1)$, and so on. In the worst case, this sequence of recursive calls covers the vertices $u_\ell, u_{\ell-1}, \ldots, u_1$ and ends when it either reaches $\pi^{1}_{\leq h'}(s,s)$ for some $0 < h' \leq h$ or a hop bound zero. This conforms to the termination conditions in (2) or Algorithm 2 lines 2-3 hold. □

To determine the computational complexity of the algorithm, we assume the candidate sets to be implemented by heaps [38]. The k shortest paths to a vertex v can be stored in a linked list, where each path $\pi^{k}_{\leq h}(s,v) = \pi^{k'}_{\leq h-1}(s,u)\cdot v$ is compactly represented by its length and a back pointer to $\pi^{k'}_{\leq h-1}(s,u)$. Using these data structures, we obtain:

Theorem 10. *The time complexity of the adapted* REA *is* $\mathcal{O}(hm + hk\log(\frac{m}{n}))$.

Proof. The computation of the first step takes $\mathcal{O}(hm)$ using the BF algorithm. Due to Lemma 9, the number of recursive invocations to $NextPath$ is bounded by h, the maximum length of $\pi^{k-1}_{\leq h}(s,t)$. At any given time, the set $Q^{k}_{\leq h}(s,v)$ contains at most $|Pred(v)|$ paths, where $Pred(v) = \{u \in V \mid (u,v) \in E\}$, i.e., one path for each predecessor vertex of v. By using heaps to store the candidate sets, a minimal element can be determined and deleted (cf. Algorithm 2, line 14) in $\mathcal{O}(\log|Pred(v)|)$ time. Insertion of a path (as in Algorithm 2, line 5,13) takes the same time complexity. Since $\sum_{v\in V}|Pred(v)| = m$, $\sum_{v\in V}\log|Pred(v)|$ is maximized when all vertices have an equal number of predecessors, i.e., $|Pred(v)| = \frac{m}{n}$. Hence, it takes $\mathcal{O}(h\log(\frac{m}{n}))$ to compute $\pi^{k}_{\leq h}(s,v)$. We have k such paths to compute, yielding $\mathcal{O}(hm + hk\log(\frac{m}{n}))$. □

Note that the time complexity is pseudopolynomial due to the dependence on k which may be exponential in n. As in our setting, k is not known in advance, hence this cannot be reduced to a polynomial time complexity.

7 Other Probability Bounds

So far we have considered properties of the form $\mathcal{P}_{\leq p}(\phi)$ for $0 < p < 1$. In this section, we will show how for the cases $\mathcal{P}_{\geq p}(\phi)$, $\mathcal{P}_{=1}(\phi)$, and $\mathcal{P}_{>0}(\phi)$, counterexamples can be generated.

7.1 Lower Bounds

In order to generate smallest counterexamples for formulas of the form $\mathcal{P}_{\geq p}(\phi)$, we propose a reduction to the case with upper probability bounds. This is done by a transformation of the formula and the DTMC at hand, while enabling us to use the algorithms presented before. As before, we distinguish unbounded and bounded until.

For $h = \infty$, we have:

$$\begin{aligned}\mathcal{P}_{\geq p}\big(\Phi \mathrm{U} \Psi\big) &\equiv \mathcal{P}_{\leq 1-p}\big(\underbrace{(\Phi \wedge \neg\Psi)}_{\Phi^*} \mathrm{W} \underbrace{(\neg\Phi \wedge \neg\Psi)}_{\Psi^*}\big)\\ &\equiv \mathcal{P}_{\leq 1-p}\big(\Phi^* \mathrm{U}(\Psi^* \vee at_b)\big),\end{aligned}$$

where at_b is a new atomic proposition such that $s \models at_b$ iff $s \in B$, where B is a bottom strongly connected component (BSCC) such that $B \subseteq Sat(\Phi^*)$, or shortly $s \in B_{\Phi^*}$. A BSCC B is a maximal strongly connected subgraph that has no transitions leaving B. Algorithmically, the DTMC is first transformed such that all the $(\neg\Phi^* \wedge \neg\Psi^*)$ states are made absorbing. Note that, once those states are reached, $\Phi^* \mathrm{W} \Psi^*$ will never be satisfied. As a second step, all the Ψ^* states are made absorbing. Finally, all BSCCs are obtained and all states in B_{Φ^*} are labeled with at_b. The obtained DTMC now acts as the starting point for applying all the model transformations and algorithms in Sections 4-6 to generate a counterexample for $\mathcal{P}_{\leq 1-p}(\Phi^* \mathrm{U}(\Psi^* \vee at_b))$.

For finite h, identifying all states in BSCCs B_{Φ^*} is not sufficient, as a path satisfying $\Box^{=h}\, \Phi^*$ may never reach such a BSCC. Instead, we transform the DTMC and use:

$$\mathcal{P}_{\geq p}(\Phi \mathrm{U}^{\leq h} \Psi) \equiv \mathcal{P}_{\leq 1-p}\big((\Phi^* \vee \Psi^*)\mathrm{U}^{=h}(\Psi^* \vee at_h)\big),$$

where at_h is an atomic proposition such that $s' \models at_h$ iff there exists $\sigma \in Paths^{\star}(s)$ such that $\sigma[h] = s'$ and $\sigma \models \Box^{=h}\, \Phi^*$. Algorithmically, the $(\neg\Phi^* \wedge \neg\Psi^*)$ states and Ψ^* states are made absorbing; and all of the Φ^*-states that can be reached in exactly h hops are computed by e.g., a breadth-first search (BFS) algorithm. The obtained DTMC now acts as the starting point for applying all the model transformations and algorithms in Sections 4-6 to generate a counterexample for $\mathcal{P}_{\leq 1-p}\big((\Phi^* \vee \Psi^*)\mathrm{U}^{=h}(\Psi^* \vee at_h)\big)$. Finite paths of exactly h hops suffice to check the validity of $\sigma \models \Box^{=h}\Phi^*$, as all Ψ^* states are absorbing.

In the explained above way, counterexamples for (bounded) until-formulas with a lower bound on their probability are obtained by considering formulas on slightly adapted DTMCs with upper bounds on probabilities. Intuitively, the fact that s refutes $\mathcal{P}_{\geq p}(\phi)$ is witnessed by showing that violating paths of s are too probable, i.e., carry more probability mass than p.

7.2 0-1 Bounds

Quantitative questions relate to the numerical value of the probability with which the property holds in the system; *qualitative* questions ask whether the property holds with

probability 0 or 1. Typically, a qualitative property can be checked using graph analysis, i.e., by just considering the underlying digraph of the DTMC and ignoring the transition probabilities. With the qualitative fragment of PCTL we can specify properties that hold *almost surely* (i.e., with probability 1) or, dually, *almost never* (i.e., with probability 0). The qualitative fragment of PCTL only allows 0 and 1 as probability bounds and only covers unbounded (weak) until [9]. Due to the fact that

$$\mathcal{P}_{=0}(\phi) \equiv \neg\mathcal{P}_{>0}(\phi) \qquad \text{and} \qquad \mathcal{P}_{<1}(\phi) \equiv \neg\mathcal{P}_{=1}(\phi),$$

it suffices to only consider formulas of the form $\mathcal{P}_{>0}(.)$ and $\mathcal{P}_{=1}(.)$. Qualitative PCTL is closely related to CTL.

Lemma 11 ([9]). *For state s of DTMC $\mathcal{D}$, it holds that:*

$$\begin{array}{lll} s \models \mathcal{P}_{>0}(a\mathrm{U}b) & \text{iff} & s \models \exists(a\mathrm{U}b), \\ s \models \mathcal{P}_{=1}(a\mathrm{U}b) & \text{iff} & s \models \forall(\exists(a\mathrm{U}b)\mathrm{W}b). \end{array}$$

As a result, a counterexample for a qualitative PCTL property is a counterexample for the corresponding CTL formula. For the violation of CTL formula $\forall\big(\exists(a\mathrm{U}b)\mathrm{W}b\big)$ in state s, it suffices to find one path $\sigma \in Paths^{\star}(s)$ such that $\sigma \models \big(\exists(a\mathrm{U}b) \wedge \neg b\big)\mathrm{U}\big(\neg\exists(a\mathrm{U}b) \wedge \neg b\big)$. Counterexamples for formulas of the form $\exists(a\mathrm{U}b)$ can be solved using the techniques in [58].

8 Rewards

Both DTMCs and PCTL can be augmented with costs, or dually rewards, which can specify standard and complex measures in a precise, unambiguous, and lucid manner. A reward function r is added to the DTMC, which associates a real reward (or: cost) to any transition. Formally, $r_i : S \times S \to \mathbb{R}_{\geq 0}$ for $1 \leq i \leq c$, where c is the number of resources in the model. $r_i(s, s')$ denotes the reward for resource i earned when taking transition $s \to s'$. The *cumulative reward* along a finite path σ is the sum of the reward on each transition along the path. Formally, $r_i(\sigma) = \sum_{l=0}^{|\sigma|-1} r_i(\sigma[l], \sigma[l+1])$.

Let $J_i \subseteq \mathbb{R}_{\geq 0}$ $(1 \leq i \leq c)$ be an interval on the real line, $p \in [0, 1]$. We use $\vec{J}$ to denote the vector of intervals, i.e., $\vec{J} = \{J_1, \ldots, J_c\}$. The formula $\mathcal{P}_{\leq p}(\Phi \mathrm{U}_{\vec{J}}\, \Psi)$ asserts that, with probability at most p, Ψ will be satisfied such that all preceding states satisfy Φ, and that the cumulative reward r_i until reaching the Ψ-state lies in the interval J_i, for $1 \leq i \leq c$. The formal semantics can be found in [6]. Note that the hop constraint $\leq h$ can be considered as a reward constraint over a simple auxiliary reward structure, which assigns cost 1 to each edge.

It holds that $s \not\models \mathcal{P}_{\leq p}(\Phi \mathrm{U}_{\vec{J}}\, \Psi)$ iff $Prob(s, \Phi \mathrm{U}_{\vec{J}}\, \Psi) > p$. As before, we cast the SE problem into a SP problem. Obviously, the weight (probability) of a path is of primary concern, which is required to be optimal. The rewards are of secondary concern; they are not required to be optimal but need to fulfil some constraints. This is exactly an instance of the (resource) constrained shortest path (CSP) problem which is NP-complete [34]. Approximation or heuristic methods are surveyed in [45]. There are some special case CSP problems. For the case $c = 1$ (a single resource) and if this resource increases in a constant unit for each edge (e.g., hop counts), the CSP problem, as is mentioned before, can be solved in PTIME. For the case $c = 1$ and not with a uniformly allocated resource and the case for $c = 2$, the CSP problem is *not* strongly NP-complete since there are pseudopolynomial algorithms to solve it exactly, in which the computational complexity depends on the values of edge weight in addition to the graph size [37]. The other cases are strong NP-complete problem.

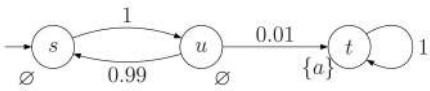

Fig. 5. A DTMC with excessive number of evidences.

For finding smallest counterexamples, we need to obtain k shortest paths subject to multiple constraints, denoted k-CSP or KMCSP [52], which is NP-complete. The KMCSP problem has received scant attention, where an exact solution is given in [52].

9 Experimentation

Smallest counterexamples may contain an excessive number of evidences, which is illustrated by the violation of $s \models \mathcal{P}_{\leq 0.9999}(\Diamond\, at_t)$ in the DTMC in Fig. 5. The smallest counterexample consists of the evidences $s{\cdot}(u{\cdot}s)^0{\cdot}u{\cdot}t, \ldots, s{\cdot}(u{\cdot}s)^{k-1}{\cdot}u{\cdot}t$, where $(u{\cdot}s)^i$ is a short form of traversing the loop $s{\cdot}u{\cdot}s$ for i times and k is the smallest integer such that $1 - 0.99^{k-1} > 0.9999$ holds. As a result, the smallest counterexample has $k = 689$ evidences. In fact, the large number of evidences degrades the significance of each evidence.

To illustrate that such phenomena also occur in real-life cases, we made a prototypical implementation (in Python) to enable generating counterexamples for more practical case studies. Our implementation uses the same input format as the probabilistic model checker MRMC [42]. Using the export facilities of PRISM [46], counterexamples can be generated for various case studies.

Let us report on one case study: the *synchronous leader election protocol* [36]. In this protocol, N processes are arranged in a unidirectional ring to elect a leader. For this purpose, they randomly select an identity (id, for short) according to a uniform distribution on $\{1, \ldots, K\}$. We call each such selection by all processes a *configuration*. By means of synchronous message passing, processes send their ids around the ring till every process sees all the ids of the others, and can thus determine whether a leader (the one with the highest unique id) can be elected. If yes, the protocol terminates; if no, a new round will be started.

We intend to find a counterexample for the following formula: $\mathcal{P}_{\leq p}(\Diamond\, leader\_elected)$, where $leader\_elected$ characterizes the global state of the protocol in which a leader has been selected. It is clear that a leader will be elected eventually. What interests us, is the number of evidences needed to converge to probability 1. We are especially interested in the relationship between the number of evidences and the bound p and R, where R is the round number. Starting a new round means that each process reselects an id and repeats the procedure.

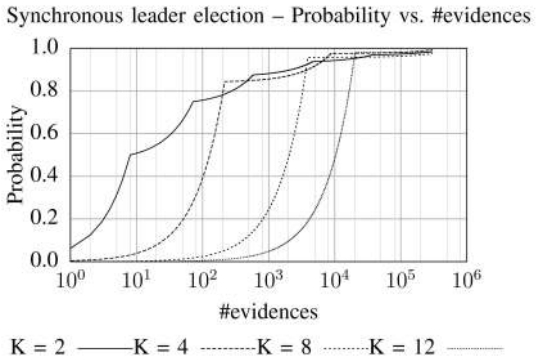

Fig. 6. Probability versus number of evidences for leader election ($N = 4$).

9.1 Experimental Results

To find the number of evidences contained in a counterexample, we used the PRISM-model of the protocol [1] and ran the counterexample generation using our implemented algorithm. The results for a fixed N ($N = 4$) and varying K are depicted in Fig. 6, where the y-axis is the accumulated probability and the x-axis (log-scale) is the number of evidences that are contained in a counterexample. The abrupt changes in the curves correspond to the start of a new round, i.e., a new election, in the protocol. Due to the fact that the probability of all evidences in one round is the same, the curves in Fig. 6 are actually piecewise linear if the x-axis were not log-scale. The curves shift more to the right when K increases since there are more possible configurations and, thus, more evidences. The larger K is, the more quickly the probability of the counterexample approaches 1. This is due to the fact that it is less probable that no process selects a unique id. All curves approach 1, which indicates that eventually a leader will be elected. The number of evidences in a counterexample, however, grows drastically to millions; whereas the probability of having elected a leader (Prob. mass) decreases drastically in each round, thus the probability per evidence decreases tremendously.

9.2 Mathematical Analysis

To obtain more insight into this rapid growth of the size of a counterexample, we carry out a brief combinatorial analysis. Let us first consider the number of possibilities (denoted $W(N,K)$) of putting N *labeled* balls into K *labeled* boxes such that each box contains at least two balls. Actually, $W(N,K)$ characterizes the number of possibilities of assigning K ids to N processes such that each id is assigned to more than one process, in which case a leader is not selected. $W(N,K)$ can be solved by using the "associated Stirling number of the second kind (S_2)" [17]:

$$W(N,K) = \sum_{j=1}^{\min(\lfloor N/2 \rfloor, K)} S_2(N,j)\frac{K!}{(K-j)!}, \qquad (4)$$

where $S_2(N,K) = K{\cdot}S_2(N{-}1,K) + (N{-}1){\cdot}S_2(N{-}2,K{-}1)$ indicates the number of ways to put N labeled balls into K *unlabeled* boxes. Obviously, it makes no sense to have more than $\lfloor N/2 \rfloor$ boxes or else it would be impossible to allocate all the balls in the right way. The factor $\frac{K!}{(K-j)!}$ expresses that there are $K!$ ways to permute the boxes (including the empty ones); for these empty boxes, the order does not matter, so we divide by $(K-j)!$.

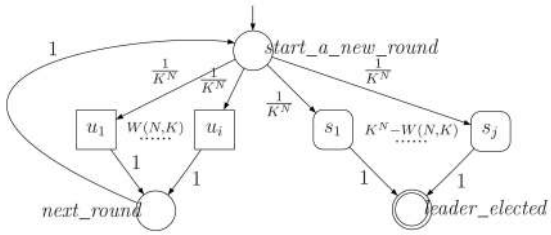

Fig. 7. Abstract leader election model.

The nonrecursive equation for $S_2(N,K)$ is:

$$S_2(N,K) = \sum_{i=0}^{K}(-1)^i\binom{N}{i}\left(\sum_{j=0}^{K-i}(-1)^j\frac{(K-i-j)^{N-i}}{j!(K-i-j)!}\right). \qquad (5)$$

For each round in the leader election protocol, the number of possibilities for a process to choose an id is K^N. Thus, the probability that N processes with K ids elect a leader in round R, denoted by $P(N,K,R)$, is:

$$P(N,K,R) = \left(\frac{W(N,K)}{K^N}\right)^{R-1}\frac{K^N - W(N,K)}{K^N}, \qquad (6)$$

where $(\frac{W(N,K)}{K^N})^{R-1}$ is the probability that a leader is not elected in the first $(R{-}1)$ rounds and $\frac{K^N-W(N,K)}{K^N}$ indicates the probability that a leader is elected in the Rth round.

We now calculate the probabilities of each evidence per round using (6). The model of the synchronous leader election protocol is depicted in Fig. 7. When we *start a new round*, there are K^N possible configurations, among which in $W(N,K)$ (square states, unsuccessful) configurations no unique id will be selected. For these states, we start the *next round*, while, in $K^N{-}W(N,K)$ (round-angle states, successful) configurations, a unique id will be selected with a *leader elected*. Thus:

Proposition 1. *The number of evidences that can reach the state leader elected in round R is:*

$$\#Evi(N,K,R) = W(N,K)^{R-1}{\cdot}\big(K^N - W(N,K)\big).$$

Proposition 1 shows that the number of evidences is exponential in R. Note that $W(N,K)$ is exponential in N and K, which makes $\#Evi(N,K,R)$ double exponential.

The number of evidences thus grows extremely fast. This results in two problems. First, it leads to the storage problem as counterexamples may simply get too large to be kept in memory. Second, and more important, counterexamples will be incomprehensible to the user. We therefore need to find ways to reduce the number of evidences in a counterexample, and to obtain a compact and user-friendly representation. To that purpose we suggest to use *regular expressions*.

10 Succinct Counterexamples

This approach is inspired by classical automata theory and is based on representing sets of paths by regular expressions.

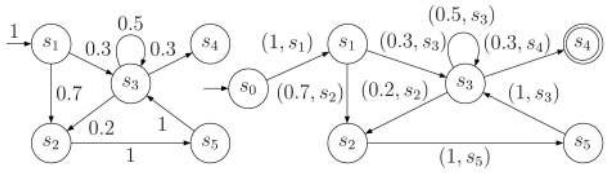

Fig. 8. An example DTMC $\mathcal{D}$ and its automaton $\mathcal{A}_{\mathcal{D}}$.

A major difference with usual regular expressions is that we need to keep track of the transition probabilities. To tackle this, we adopt the approach proposed by Daws [20]. He uses regular expressions to represent sets of paths and calculates the exact rational value of the probability measure in DTMC model checking (provided all transition probabilities are rational). We adapt this approach to obtain compact representations of counterexamples. The main idea is to consider a counterexample as a set of probable branches (subexpressions) that go from the initial state to the goal state and to provide a function to evaluate the probability measure of those expressions. To simplify the presentation, we will assume that the DTMC at hand has been subject to the transformation in Step 1, cf. Section 4. This is not a limitation since $s \models \mathcal{P}_{\trianglelefteq p}(\Phi \mathrm{U}^{\leq h}\, \Psi)$ in a DTMC iff $s \models \mathcal{P}_{\trianglelefteq p}(\Diamond^{\leq h+1}\ at_t)$ in the transformed DTMC where at_t uniquely identifies t.

10.1 Turning a DTMC into an Automaton

For DTMC $\mathcal{D} = (S, \mathbf{P}, L)$ with initial state $\hat{s} \in S$ and goal state t, let the deterministic finite automaton (DFA) $\mathcal{A}_{\mathcal{D}} = (S', \Sigma, \tilde{s}, \delta, \{t\})$, where:

- $S' = S \cup \{\tilde{s}\}$ is the state space with start state $\tilde{s} \notin S$;
- $\Sigma \subseteq (0, 1] \times S$ is the (finite) alphabet;
- $\delta \subseteq S' \times \Sigma \times S'$ is the transition relation such that $\delta(s, (p, s')) = s'$ iff $\mathbf{P}(s, s') = p$, and $\delta(\tilde{s}, (1, \hat{s})) = \hat{s}$;
- $t \in S$ is the accepting state.

The automaton is equipped with a start state $\tilde{s}$ with a transition of probability one to the initial state of $\mathcal{D}$. Symbols in the alphabet are pairs (p, s) with p a probability and s a state. Transition $s \xrightarrow{p} s'$ in $\mathcal{D}$ is turned into a transition from s to s' labeled with (p, s'). (Obviously, this yields a deterministic automaton.) This is a slight, though important deviation from [20], where labels are just probabilities. The probabilities are needed to determine the path probabilities (see Definition 14), while the target states are used for recovering the evidences. For simplicity, probability labels are omitted if they are clear from the context.

Example 6. Fig. 8 (left) depicts an abstract example of a DTMC $\mathcal{D}$ with initial state $\hat{s} = s_1$ and goal state $t = s_4$ and its DFA $\mathcal{A}_{\mathcal{D}}$ (right). The new start state is $\tilde{s} = s_0$, which has a transition equipped with symbol $(1, s_1)$ to s_1.

10.2 Evaluation of Regular Expressions

Regular expressions will be used to represent a counterexample C. To determine the probability of C, $\mathbb{P}(C)$, from its regular expression we use an evaluation function. Let $\mathcal{R}(\Sigma)$ be the set of regular expressions over the finite alphabet Σ. It contains the elements of Σ, the empty word ε, and is closed under union ($|$), concatenation (.), and Kleene star ($*$). Let $\mathcal{L}(r)$ denote the regular language (a set of words) described by the regular expression $r \in \mathcal{R}(\Sigma)$ and $\mathcal{L}(\Sigma)$ denote the regular language that can be generated by any regular expression over Σ. The length $|z|$ and $|r|$ denote the number of symbols in the word z and regular expression r, respectively. We sometimes omit . and write $r.r'$ as rr' for short. Note that in our setting, $\Sigma \subseteq (0, 1] \times S$.

Definition 14 ([20], Evaluating regular expressions). *Let* $val : \mathcal{R}(\Sigma) \mapsto \mathbb{R}$ *be defined as:*

$$\begin{array}{ll} val(\varepsilon) = 1 & val(r|r') = val(r) + val(r') \\ val((p, s)) = p & val(r.r') = val(r) \times val(r') \end{array}$$

$$val(r^*) = \begin{cases} 1 & \text{if } val(r) = 1 \\ \frac{1}{1 - val(r)} & \text{o.w.} \end{cases}$$

If we limit the transition probabilities to being rational values, then exact values are obtained. It can be proven that $val(r) = \mathbb{P}\big(Paths^{\star}_{\min}(\hat{s}, \Diamond^{\leq h}\, at_t)\big)$, for $h = \infty$ [20].

Definition 15. r_1 *is a* maximal union subexpression (MUS) *of a regular expression* r *if* $r = r_1 \,|\, r_2$ *modulo* $(\mathbf{R_1})$-$(\mathbf{R_3})$, *for some* $r_2 \in \mathcal{R}(\Sigma)$, *where:*

$$\begin{array}{lrcl} (\mathbf{R_1}) & r & \equiv & r \mid \varepsilon, \\ (\mathbf{R_2}) & r_1 \mid r_2 & \equiv & r_2 \mid r_1, \\ (\mathbf{R_3}) & (r_2 \mid r_3) & \equiv & (r_1 \mid r_2) \mid r_3. \end{array}$$

r_1 is maximal because it is at the topmost level of a union operator. If the topmost level operator is not union, then $r_1 = r$ (cf. $\mathbf{R_1}$). A regular expression represents a set of paths and each MUS can be regarded as a main branch from the start state to the accepting state.

Example 7. A regular expression for the automaton $\mathcal{A}_{\mathcal{D}}$ in Fig. 8 (right) is:

$$r_0 = \underbrace{s_1 s_3 s_3^* s_4}_{r_1} \mid \underbrace{s_1 (s_2 | s_3 s_3^* s_2)(s_5 s_3 s_3^* s_2)^* s_5 s_3 s_3^* s_4}_{r_2}.$$

r_1 and r_2 are the MUSs of r_0 with $val(r_1) = 1 \times 0.3 \times \frac{1}{1-0.5} \times 0.3 = 0.18$ and $val(r_2) = 0.82$. Note that $|r_1| = 4$ and $|r_2| = 13$; $z = s_1 s_3 s_3 s_3 s_4$ is a word generated by r_1 and $|z| = 5$. We can distribute $|$ over . in r_2 and obtain two more MUSs instead: $r_3 = s_1 s_2 (s_5 s_3 s_3^* s_2)^*\, s_5 s_3 s_3^* s_4$ and $r_4 = s_1 s_3 s_3^* s_2 (s_5 s_3 s_3^* s_2)^* s_5 s_3 s_3^* s_4$. r_1, r_3 and r_4 characterize all paths from s_1 to s_4, which fall into the above three branches. Note that r_1 *cannot* be written as $s_1 s_3^+ s_4$ since, from the full form of $r_1 = (1, s_1)(0.3, s_3)\ (0.5, s_3)^*(0.3, s_4)$, the probability of the first s_3 is different from that of s_3^*.

10.3 Regular Expressions as Counterexamples

The equivalence of DFAs and regular expressions, as well as converting DFAs to regular expressions has been widely studied. Several techniques are known, e.g., the transitive closure method [44], Brzozowski's algebraic method [14], [13], and state elimination [25], [51]. State elimination is based on removing states one by one, while labeling transitions by regular expressions. It terminates only once the start and accepting state remain; the transition connecting these states is labeled with the resulting regular expression. This technique is suitable for manual inspection

but is less straightforward to implement. The transitive closure method gives a clear and simple implementation but tends to create rather long regular expressions. The algebraic method is elegant and generates reasonably compact regular expressions. For a more detailed comparison, see [55]. In order to obtain a minimal counterexample in an *on-the-fly* manner, we take the state elimination method. This allows us to stop once the value of the obtained regular expression exceeds the probability threshold. The algebraic method does not support this.

By using regular expressions for representing counterexamples, we will, instead of obtaining evidences one by one, derive a larger number of evidences at a time, which hopefully yields a quick convergence to the required probability threshold and a clear explanation of the violation. As a result, we will not insist on obtaining the smallest counterexample but instead prefer to find the branches (MUSs) with large probabilities and short length. Thus, a (good) regular expression should be

1. shorter (w.r.t. its length), to improve comprehensibility;
2. more probable, such that it is more informative and the algorithm will terminate with less MUSs;
3. minimal, where a compact counterexample is *minimal* if the omission of any of its MUSs would no longer result in a counterexample.

However, it has been recently proven that the size of a shortest regular expression of a given DFA cannot be efficiently approximated [28]. Therefore, it is not easy to, e.g., by state elimination, compute an optimal removal sequence for state elimination in polynomial time [32]. We could adapt the heuristics proposed in, e.g., [32], [23] to get a better order to eliminate states. For 2, we could take the advantage of the KSP or HKSP algorithms as well as the model-checking results. The states on the more probable evidences should be eliminated first.

We take the following iterative strategy: In each iteration, we take the strongest evidence $\sigma = \tilde{s}\cdot\hat{s}\cdot s_1 \cdots s_j \cdot t$ in the remaining automaton—recall that this amounts to an SP problem—and eliminate all of the intermediate states on σ (i.e., $\hat{s}, s_1, \ldots, s_j$) one by one according to a recently proposed heuristic order [32]. After eliminating a state, possibly a new MUS r_k, say, is created where k MUSs have been created so far, and $val(r_k)$ can be determined. If $\sum_{i=1}^{k} val(r_i) > p$, then the algorithm terminates. Otherwise, the transition labeled with r_k is removed from the automaton and either a next state is selected for elimination or a new evidence is to be found, cf. Algorithm 3.

Algorithm 3 Regular expression counterexamples
Require: DFA $\mathcal{A}_{\mathcal{D}} = (S, \Sigma, \tilde{s}, \delta, \{t\})$ and $p \in [0,1]$
Ensure: regular expression $r \in \mathcal{R}(\Sigma)$ with $val(r) > p$
1: $\mathcal{A} := \mathcal{A}_{\mathcal{D}}$; $pr := 0$; Priority queue $q := \emptyset$; $k := 1$;
2: **while** $pr \leq p$ **do**
3: $\sigma :=$ the strongest evidence in $\mathcal{A}$;
4: **forall** $s' \in \sigma \setminus \{\tilde{s}, t\}$ **do** *q.enqueue*(s'); **endforall**;
5: **while** $q \neq \emptyset$ **do**
6: $\mathcal{A} :=$ *eliminate*(*q.dequeue*()); $r_k :=$ the created MUS;
7: $pr := pr + val(r_k)$; $\mathcal{A} :=$ *eliminate*(r_k);
8: **if** $(pr > p)$ **then break**; **else** $k := k + 1$;
9: **endwhile**;
10: **endwhile**;
11: **return** $r_1 \mid \ldots \mid r_k$;

Priority queue q keeps the states to be eliminated in the current iteration. The order in which states are dequeued from q is given by the heuristics in [32]. The function "*eliminate*$(\cdot)$" can eliminate both states and regular expressions, where the latter simply means the deletion of the transitions labeled with the regular expression.

Example 8. Let us apply the algorithm on $\mathcal{A}_{\mathcal{D}}$ of Fig. 8 and $\mathcal{P}_{\leq 0.7}(\Diamond\, s_4)$. In the first iteration, $s_0{\cdot}s_1{\cdot}s_2{\cdot}s_5{\cdot}s_3{\cdot}s_4$ is found as the strongest evidence. Assuming the order to eliminate the states by [32] is s_5, s_2, s_1, s_3, we obtain the regular expression $r_5{=}s_1(s_3|s_2s_5s_3)(s_3|s_2s_5s_3)^*s_4$ with $val(r_5){=}1$. Since all states are eliminated and the threshold 0.7 is exceeded, the algorithm terminates. This expression gives a clear reason that traversing the cycle s_3 or $s_2s_5s_3$ infinitely many times causes the probability exceeding 0.7.

Let us *change the elimination order* to s_5, s_1, s_3, s_2. Then, the regular expression is $r_0 = s_1s_3s_3^*s_4 \mid s_1(s_2|s_3s_3^*s_2)(s_5s_3s_3^*s_2)^*s_5s_3s_3^*s_4$. After eliminating s_3, the first MUS $r_1 = s_1s_3s_3^*s_4$ is generated and the probability is $0.18 < 0.7$. The algorithm continues (i.e., eliminates s_2) to find more MUSs till r_0 is found. Note that r_0 is longer than r_5 and, thus, less intuitive to comprehend. The cycles s_3 and $s_3s_2s_5$ are, however, indicated.

Let us pick a *less probable evidence* $s_0{\cdot}s_1{\cdot}s_3{\cdot}s_4$ to be eliminated in the first iteration. After eliminating s_3, the resulting expression is $r_1{=}s_1s_3s_3^*s_4$. Then, r_1 is removed from the automaton and the strongest evidence in the remaining automaton is $s_0{\cdot}s_1{\cdot}s_2{\cdot}s_5{\cdot}s_4$. After eliminating s_2, s_5, we obtain the regular expression: r_2, as in Example 7. The final regular expression is again r_0 and the analysis in the last case applies.

Proposition 2. *The regular expression counterexample generated by Algorithm 3 is minimal.*

This property immediately follows from the fact that Algorithm 3 terminates immediately once the cumulative probability exceeds the threshold. We would like to emphasize that the regular expression representation is not applicable for formulas with nested probabilistic operators, e.g., $\mathcal{P}_{\leq p_1}\big(\Diamond\, \mathcal{P}_{\leq p_2}(\Diamond\, at_t)\big)$. However, this is not a real constraint in practice since those formulas are rarely used. In addition, it is important to mention that the algorithm in this section not only applies to nonstrict probability bounds, but also to strict bounds as, e.g., $\mathcal{P}_{<p}(\Diamond^{\leq h}\, at_t)$.

10.4 Bounded Reachability

For bounded reachability formula $\Diamond^{\leq h}\, at_t$, a regular expression, e.g. $r = r_1|r_2^*$, may not be valid because it is possible that the length of the words generated by r_1 or the expansion of r_2 exceeds h. Thus, $val(r)$ might be larger than the actual probability. In order to obtain a precise valuation, we consider *constrained regular expressions*.

Definition 16 (Constrained regular expressions). *For $r \in \mathcal{R}(\Sigma)$ and $h \in \mathbb{N}_{\geq 0}$, $\mathcal{L}(r[h]) = \{z \in \mathcal{L}(r) \mid |z| \leq h\}$.*

In fact, $\mathcal{L}(r[h]) \subseteq \mathcal{L}(r)$ and $r[h]$ can be expressed equivalently by a union of possible enumerations, namely $r[h] = r\langle 0\rangle | r\langle 1\rangle | \cdots | r\langle h\rangle$, where $r\langle i\rangle$ denotes the set of words generated by r and having exactly i symbols. Constrained regular expressions can be obtained in the same way as presented just before, only their valuation is different.

Definition 17. *For $r \in \mathcal{R}(\Sigma)$ and $h \in \mathbb{N}_{\geq 0}$, the function val for $r[h]$ and $r\langle h\rangle$ is defined by:*

$$
\begin{aligned}
val(r[h]) &= \sum_{i=0}^{h} val(r\langle h\rangle),\\
val(\varepsilon\langle h\rangle) &= \begin{cases} 1, & \text{if } h = 0,\\ 0, & \text{o.w.,}\end{cases}\\
val((p,s)\langle h\rangle) &= \begin{cases} p, & \text{if } h = 1,\\ 0, & \text{o.w.,}\end{cases}\\
val(r_1|r_2\langle h\rangle) &= val(r_1\langle h\rangle) + val(r_2\langle h\rangle),\\
val(r_1.r_2\langle h\rangle) &= \sum_{i=0}^{h} val(r_1\langle i\rangle)\cdot val(r_2\langle h-i\rangle),\\
val(r^*\langle h\rangle) &= val(\varepsilon\langle h\rangle) + \sum_{i=1}^{h} val(r\langle i\rangle)\cdot val(r^*\langle h-i\rangle).
\end{aligned}
$$

Note that the complexity of the above evaluation function is, however, very high. It remains to establish that constrained regular expressions are counterexamples for bounded until-formulas.

Theorem 12. *Let r be the regular expression for* DFA *$\mathcal{A}_{\mathcal{D}} = (S', \Sigma, \tilde{s}, \delta, \{t\})$, where $\mathcal{D} = (S, \mathbf{P}, L)$ with initial state $\hat{s}$ and $h \in \mathbb{N}_{\geq 0}$. Then:*

$$val(r[h]) \;=\; \mathbb{P}\big(Paths^{\star}_{\min}(\hat{s}, \Diamond^{\leq h} at_t)\big).$$

10.5 Leader Election Example

We conclude this section by reconsidering the leader election protocol. For the original DTMC, the regular expression, denoted $r(N,K)$, is:

$$start.\big((u_1|\cdots|u_i).next.start\big)^*.\big(s_1|\cdots|s_j\big).leader,$$

where *start*, *next*, and *leader* are the obvious short forms. The regular expression lists all the unsuccessful configurations, as well as the successful ones. As a result, $|r(N,K)| = K^N{+}4$. Compared to the number of evidences computed directly, i.e., $\sum_{i=1}^{R} \#Evi(N,K,i)$, $|r(N,K)|$ is much shorter, but it is still exponentially long. On the other hand, however, the structure of $r(N,K)$ clearly indicates the reason of violation, i.e., the repeated unsuccessful configurations followed by a successful one.

Regular expression counterexamples are feasible when the excessive number of evidences are caused by traversing loops. Clearly, the number of states also affects the size of the regular expression. Thus, any model reduction prior to counterexample generation would be helpful. Two strategies may be utilized to slim down the model size, viz., *bisimulation minimization* and SCC *minimization*. Bisimulation minimization [41], [48] lumps bisimilar states and yields a typically much smaller quotient DTMC. Strongly-connected-component (SCC) minimization [50], instead, only lumps SCCs. Bisimulation minimization preserves both unbounded and bounded probabilistic reachability properties, while SCC minimization only preserves the former one. An evidence $[s_0]_{\sim}\cdot[s_1]_{\sim}\cdot\cdots[s_n]_{\sim}$ in the quotient DTMC represents a set of evidences in the original DTMC, viz., $\{s'_0\cdot s'_1\cdots s'_n \mid s'_i \in [s_i]_{\sim} \text{ and } 0 \leq i \leq n\}$, where $[s_i]_{\sim}$ is the equivalence class under equivalence $\sim$ with s_i as its representative.

For the leader election protocol, the regular expression counterexample in the bisimulation quotient DTMC is:

$$r_{\sim}(N,K) = start.(u.next.start)^*.s.leader,$$

where $u_1,\ldots,u_i$ are wrapped as u; $s_1,\ldots,s_j$ as s in Fig. 7. Note that $|r_{\sim}(N,K)| = 6$ is independent of N and K. The SCC-quotient DTMC is obtained by replacing the left half of the model (an SCC) by a self-loop on the initial state. The regular expression counterexample is:

$$r^{SCC}(N,K) = start.start^*.(s_1|\cdots|s_j).leader,$$

where the intuition of the self-loop is "still unsuccessful." Applying both techniques yield:

$$r^{SCC}_{\sim}(N,K) = start.start^*.s.leader.$$

11 Case Study—Crowds Protocol

We now illustrate our techniques on a more serious example. The *Crowds* protocol [56] is aimed to provide users with a mechanism for anonymous Web browsing. The main idea behind Crowds is to hide each user's communication by routing randomly within a group of similar users. Even if a local eavesdropper or a bad group member observes a message being sent by a particular user, it can never be sure whether the user is the actual sender, or is simply routing another user's message.

The protocol works in the following way: 1) The sender selects a crowd member at random (possibly itself), and forwards the message to it, encrypted by the corresponding pairwise key. 2) The selected router flips a biased coin. With probability $1 - PF$, where PF (forwarding probability) is a parameter of the system, it delivers the message directly to the destination. With probability PF, it selects a crowd member at random (possibly itself) as the next router in the path, and forwards the message to it, reencrypted with the appropriate pairwise key. The next router repeats this step.

In our experiments, we assume that: 1) If a sender has been observed by the bad member twice, then it has been *positively identified* (*Pos* for short), thus the anonymity is not preserved. 2) The bad member will deliver the message with probability 1, as in [57]. This protocol is executed every time a crowd member wants to establish an anonymous connection to a Web server. We call one run of the protocol a *session* and denote the number of sessions by R. Other parameters are the number N of good members and the number M of bad members.

We take the Crowds protocol modeled by PRISM [1] and the property is $\mathcal{P}_{\leq p}(\Diamond\, Pos)$, which characterizes the

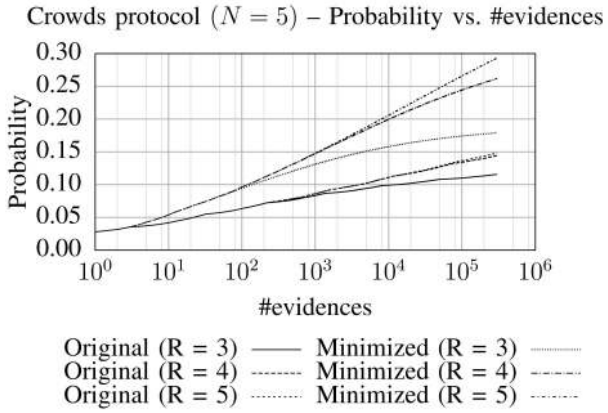

Fig. 9. Probability versus number of evidences.

probability threshold that the original sender's id 0 is positively identified by any of the bad members. The relation between the number of evidences and the probability threshold for different number of sessions R is shown in Fig. 9 ($N = 5$, $M = 1$, $PF = 0.8$), both for the original DTMC and its bisimulation quotient. As one evidence in the minimized model may represent a set of evidences in the original model, given the same number of evidences, the cumulative probability in the minimized model is higher than that in the original model. This is illustrated in Fig. 9 by the fact that the curves for the minimized model are above the ones for the original model. It also holds that the larger the round number we allow, the larger the cumulative probability is, as the smaller round case is subsumed in the larger round case. For instance, in the case of $R = 2$, only consecutively sending to two bad members causes a Pos (shortly as BB); whereas in the case of $R = 3$, besides BB, there are two more situations that causes Pos, viz. GBB and BGB (G represents sending to a good member). This also explains why the curves overlap in the beginning of the x-axis.

We choose a configuration with a small state space ($N = 2$, $M = 1$, $R = 2$, and $PF = 0.8$) as this suffices to illustrate the algorithm. Bisimulation minimization reduces the state space from 77 to 34; cf. the quotient DTMC in Fig. 10. To make the figure comprehensible, sequences of states with probability 1 are depicted by a square state. States i, G, B, Del, Pos represent *i*nitiating a new session, sending a message to a *G*ood member, to a *B*ad member, a message being *Del*ivered, a *Pos*itive result obtained, respectively. $G0$ and $G1$ are the two

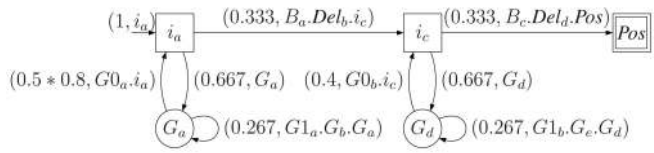

Fig. 11. A more compact automaton.

good members, where $G0$ is assumed always to be the original sender when a new session starts. $G0 \vee G1$ is a lumped state where either $G0$ or $G1$ is reached. The subscripts $a, b, \ldots$ are to distinguish the states in similar situations. Since the goal state Pos can be reached by only the gray states, the regular expression (thus the automaton) only depends on those states. Note that Del_a and Del_b denote the end of the first session, while Del_c and Del_b denote the end of the second. Only the case that two messages are both delivered by the bad member indicates a positive identification of the sender.

An intermediate automaton (see Fig. 11) can be derived after eliminating some states. The start state $\tilde{s}$ of the automaton is also omitted. This shows the basic structure of the model: i_a and i_c are the starting points of two sessions. The horizontal transitions indicate the observation of $G0$ by the bad member, which lead to Pos. In each session, a message can be forwarded to $G0$ or $G1$ many times (captured by the self-loops). Once a message is delivered, a new session is assumed to be started (the transitions back to i_a and i_c). Thus, a regular expression that can be generated from the automaton is $r = r_0 r_1^* r_2 r_3^* r_4$, where:

$$
\begin{aligned}
r_0 &= (1, i_a),\\
r_1 &= (0.667, G_a)(0.267, G1_a.G_b.G_a)^*(0.4, G0_a.i_a),\\
r_2 &= (0.333, B_a.Del_b.i_c),\\
r_3 &= (0.667, G_d)(0.267, G1_b.G_e.G_d)^*(0.4, G0_b.i_c),\\
r_4 &= (0.333, B_c.Del_d.Pos).
\end{aligned}
$$

If we omit the probabilities and the subscripts and merge the stuttering steps G, then we obtain:

$$
r' = i\,\underbrace{(G.(G1.G)^*G0.i)^*}_{good}.\underbrace{(B.Del.i)}_{bad}.\underbrace{(G.(G1.G)^*G0.i)^*}_{good}.\underbrace{B}_{bad},
$$

which is highly compact and informative in the sense that it indicates the observation of the bad members twice with arbitrary number of observing the good members. r' can be

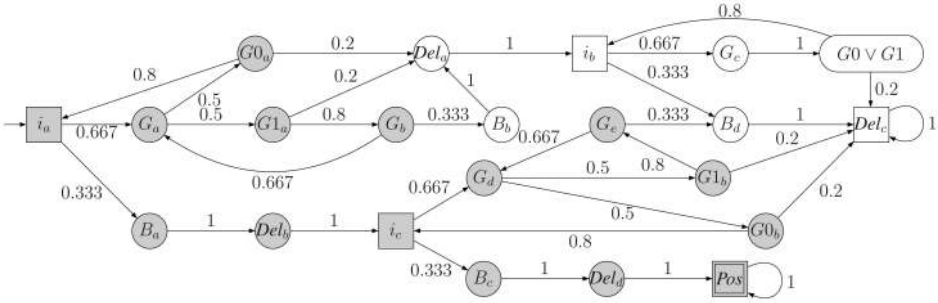

Fig. 10. State space of the quotient DTMC for the Crowds protocol $(N = 2, M = 1, R = 2, PF = 0.8)$.

TABLE 1
Time Complexity of the Counterexample Problems

counterex. problem	SP problem	algorithm	time complexity
SE (U)	SP	Dijkstra	$\mathcal{O}(m + n \log n)$
SE ($U^{\leq h}$)	HSP	BF/Viterbi	$\mathcal{O}(hm)$
SC (U)	KSP	Eppstein	$\mathcal{O}(m + n \log n + k)$
SC ($U^{\leq h}$)	HKSP	Adapted REA	$\mathcal{O}(hm + hk \log(\frac{m}{n}))$

further compacted if the SCCs are identified and replaced by self-loops. In this case, $r'' = i.i^*.(B.Del.i).i^*.B$.

The probability of r is $val(r) = 0.274$, which coincides with the model checking result. These probabilities depend, among others, on the parameters of the protocol (N, M, R, PF, etc.). For instance, the probability of the strongest evidence is $(\frac{M}{N+M})^R = (\frac{1}{3})^2 = \frac{1}{9}$, which loops 0 times at r_1 and r_3. The probability of r_2 and r_4 is $\frac{a}{1-a} = \frac{4}{11}$, where a is the probability of the inner loop: $\frac{1}{N+M} \cdot PF \cdot (1 - \frac{M}{N+M}) = 0.267$, as is shown in the intermediate automaton. Note that this closed-form expression can now be used for arbitrary parameter values.

12 Conclusion

In this paper, we have provided the theoretical and algorithmic foundations for counterexample generation in probabilistic model checking, in particular for discrete-time Markov chains. One of the key principles has been the casting of the concepts of strongest evidence and smallest counterexample as (variants of) shortest path (SP) problems. This enabled the use of efficient and well-studied graph algorithms for counterexample generation.

It has been shown in detail which algorithms can be applied when checking properties with (nonstrict) probability upper bounds. These algorithms are central to counterexample generation for PCTL, both for upper and lower probability bounds. All cases can be treated by standard SP algorithms or small amendments thereof. An overview of the main cases is given in Table 1, where n and m are the number of states and transitions in the Markov chain, h is the hop bound, and k is the number of shortest paths. Generating strongest evidences and smallest counterexamples for reward extensions of Markov chains is NP-complete.

A second main contribution is an algorithm to generate regular expressions that can act as (rather compact) counterexamples.

The results in this paper are applicable to LTL model checking, as LTL model checking of DTMCs reduces to probabilistic reachability. In addition, the results are useful for MDP model checking: When a state refutes a property, such as the maximal reachability probability never exceeds a bound p, say, a memoryless policy is obtained that yields this violation. Applying our algorithms to the Markov chain induced by this policy provides useful diagnostic feedback. In fact, this strategy has recently been adopted in probabilistic CEGAR for MDPs [35]. Other applications include CTMC counterexample generation [31] and counterexample generation for cpCTL [8]. Recently, Andrés et al. [7] have applied SCC minimization to counterexample generation for MDPs. We foresee that applying the CEGAR-framework to other abstraction techniques, such as [43], [47], may also profit from the results in this paper.

Acknowledgments

Christel Baier, David N. Jansen and Alexandru Mereacre are kindly acknowledged for their useful remarks on the paper. The authors thank the reviewers for their detailed and helpful feedback. This research has been supported by the NWO project QUPES and the EU FP7 project QUASIMODO.

References

[1] PRISM Website, http://www.prismmodelchecker.org, 2009.
[2] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows: Algorithms, Theory and Applications*. Prentice Hall, Inc., 1993.
[3] H. Aljazzar, H. Hermanns, and S. Leue, "Counterexamples for Timed Probabilistic Reachability," *Proc. Int'l Conf. Formal Modelling and Analysis of Timed Sequences*, pp. 177-195, 2005.
[4] H. Aljazzar and S. Leue, "Extended Directed Search for Probabilistic Timed Reachability," *Proc. Int'l Conf. Formal Modelling and Analysis of Timed Sequences*, pp. 33-51, 2006.
[5] C. Andersson, S. Fischer-Hübner, and R. Lundin, "Enabling Anonymity for the Mobile Internet Using the Mcrowds System," *IFIP WG 9.2, 9.6/11.7 Summer School on Risks and Challenges of the Network Soc.*, p. 35, 2004.
[6] S. Andova, H. Hermanns, and J.-P. Katoen, "Discrete-Time Rewards Model-Checked," *Proc. Int'l Conf. Formal Modelling and Analysis of Timed Sequences*, pp. 88-104, 2003.
[7] M.E. Andrés, P. D'Argenio, and P. van Rossum, "Significant Diagnostic Counterexamples in Probabilistic Model Checking," *Proc. Fourth Haifa Verification Conf.*, to be published.
[8] M.E. Andrés and P. van Rossum, "Conditional Probabilities over Probabilistic and Nondeterministic Systems," *Proc. Int'l Conf. Tools and Algorithms for the Construction and Analysis of Systems*, pp. 157-172, 2008.
[9] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.
[10] T. Ball, M. Naik, and S.K. Rajamani, "From Symptom to Cause: Localizing Errors in Counterexample Traces," *Proc. Symp. Principles of Programming Language*, pp. 97-105, 2003.
[11] G. Behrmann, K.G. Larsen, and J.I. Rasmussen, "Optimal Scheduling Using Priced Timed Automata," *ACM SIGMETRICS Performance Evaluation Rev.*, vol. 32, no. 4, pp. 34-40, 2005.
[12] R. Bellman, "On a Routing Problem," *Quarterly Applications Math.*, vol. 16, no. 1, pp. 87-90, 1958.
[13] G. Berry and R. Sethi, "From Regular Expressions to Deterministic Automata," *Theoretical Computer Science*, vol. 48, no. 3, pp. 117-126, 1986.
[14] J.A. Brzozowski, "Derivatives of Regular Expressions," *J. ACM* vol. 11, no. 4, pp. 481-494, 1964.
[15] E.M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Counterexample-Guided Abstraction Refinement," *Proc. Int'l Conf. Computer Aided Verification*, pp. 154-169, 2000.
[16] E.M. Clarke, S. Jha, Y. Lu, and H. Veith, "Tree-Like Counterexamples in Model Checking," *Proc. IEEE Symp. Logic in Computer Science*, pp. 19-29, 2002.
[17] L. Comtet, *Advanced Combinatorics: The Art of Finite and Infinite Expansion*. D. Reidel Publishing Co., 1974.
[18] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, "The Bellman-Ford Algorithm," *Introduction to Algorithms*, chapter 24.1, pp. 588-592, MIT Press, 2001.
[19] B. Damman, T. Han, and J.-P. Katoen, "Regular Expressions for PCTL Counterexamples," *Proc. Int'l Conf. Quantitative Evaluation of Systems*, 2008.
[20] C. Daws, "Symbolic and Parametric Model Checking of Discrete-Time Markov Chains," *Proc. Int'l Colloquim Theoretical Aspects of Computing*, pp. 280-294, 2004.
[21] L. de Alfaro, T.A. Henzinger, and F.Y. Mang, "Detecting Errors Before Reaching Them," *Proc. Int'l Conf. Computer Aided Verification*, pp. 186-201, 2000.

[22] E. de Queirós Vieira Martins, M.M.B. Pascoal, and J.L.E. dos Santos, "Deviation Algorithms for Ranking Shortest Paths," *Int'l J. Foundations of Computer Science,* vol. 10, no. 3, pp. 247-262, 1999.
[23] M. Delgado and J. Morais, "Approximation to the Smallest Regular Expression for a Given Regular Language," *Proc. Int'l Conf. Implementation and Application of Automata,* pp. 312-314, 2004.
[24] E.W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik,* vol. 1, pp. 269-271, 1959.
[25] D.-S. Du and K.I. Ko, *Problem Solving in Automata, Languages, and Complexity.* John Wiley and Sons, 2001.
[26] D. Eppstein, "Finding the k Shortest Paths," *SIAM J. Computing,* vol. 28, no. 2 pp. 652-673, 1998.
[27] M.R. Garey and D.S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness.* Freeman, 1979.
[28] G. Gramlich and G. Schnitger, "Minimizing NFA's and Regular Expressions," *J. Computer Systems and Sciences* vol. 73, no. 6, pp. 908-923, 2007.
[29] A. Gurfinkel and M. Chechik, "Proof-Like Counter-Examples," *Proc. Int'l Conf. Tools and Algorithms for the Construction and Analysis of Systems,* pp. 160-175, 2003.
[30] T. Han and J.-P. Katoen, "Counterexamples in Probabilistic Model Checking," *Proc. Int'l Conf. Tools and Algorithms for the Construction and Analysis of Systems,* pp. 72-86, 2007.
[31] T. Han and J.-P. Katoen, "Providing Evidence of Likely Being on Time: Counterexample Generation for CTMC Model Checking," *Proc. Int'l Symp. Automated Technology for Verification and Analysis,* pp. 331-346, 2007.
[32] Y.-S. Han, D. Wood, "Obtaining Shorter Regular Expressions from Finite-State Automata," *Theoretical Computer Science,* vol. 370, nos. 1-3, pp. 110-120, 2007.
[33] H. Hansson and B. Jonsson, "A Logic for Reasoning About Time and Reliability," *Proc. Int'l Workshop Formal Aspects of Component Software,* vol. 6, no. 5, pp. 512-535, 1994.
[34] S. Hart, M. Sharir, and A. Pnueli, "Termination of Probabilistic Concurrent Programs." *ACM Trans. Programing Languages Systems,* vol. 5, no. 3, pp. 356-380, 1983.
[35] H. Hermanns, B. Wachter, and L. Zhang, "Probabilistic CEGAR," *Proc. Int'l Conf. Computer Aided Verification,* pp. 162-175, 2008.
[36] A. Itai and M. Rodeh, "Symmetry Breaking in Distributed Networks," *Information and Computation,* vol. 88, no. 1, pp. 60-87, 1990.
[37] J.M. Jaffe, "Algorithms for Finding Paths with Multiple Constraints," *IEEE Network,* vol. 14, pp. 95-116, 1984.
[38] V.M. Jiménez and A. Marzal, "Computing the k Shortest Paths: A New Algorithm and an Experimental Comparison," *Proc. Workshop Algorithmic Eng.,* pp. 15-29, 1999.
[39] H. Jin, K. Ravi, and F. Somenzi, "Fate and Free Will in Error Traces," *Proc. Conf. Software Tools for Technology Transfer,* vol. 6, no. 2, pp. 102-116, 2004.
[40] G.D.F. Jr, "The Viterbi Algorithm," *Proc. IEEE,* vol. 61, no. 3, pp. 268-278, 1973.
[41] J.-P. Katoen, T. Kemna, I. Zapreev, and D. Jansen, "Bisimulation Minimisation Mostly Speeds Up Probabilistic Model Checking," *Proc. Int'l Conf. Tools and Algorithms for the Construction and Analysis of Systems,* pp. 87-101, 2007.
[42] J.-P. Katoen, M. Khattri, and I.S. Zapreev, "A Markov Reward Model Checker," *Proc. Int'l Conf. Quantitative Evaluation of Systems,* pp. 243-244, 2005.
[43] J.-P. Katoen, D. Klink, M. Leucker, and V. Wolf, "Three-Valued Abstraction for Continuous-Time Markov Chains," *Proc. Int'l Conf. Computer Aided Verification,* pp. 311-324, 2007.
[44] S.C. Kleene, *Representation of Events in Nerve Nets and Finite Automata,* pp. 3-42, Princeton Univ. Press, 1956.
[45] F.A. Kuipers, T. Korkmaz, M. Krunz, and P. van Mieghem, "Performance Evaluation of Constraint-Based Path Selection Algorithms," *IEEE Network* vol. 18, no. 5, pp. 16-23, 2004.
[46] M.Z. Kwiatkowska, G. Norman, and D. Parker, "Probabilistic Symbolic Model Checking with PRISM: A Hybrid Approach," *Proc. Conf. Software Tools for Technology Transfer* vol. 6, no. 2, pp. 128-142, 2004.
[47] M.Z. Kwiatkowska, G. Norman, and D. Parker, "Game-Based Abstraction for Markov Decision Processes," *Proc. Int'l Conf. Quantitative Evaluation of Systems,* pp. 157-166, 2006.
[48] K.G. Larsen and A. Skou, "Bisimulation Through Probabilistic Testing," *Information and Computation,* vol. 94, no. 1, pp. 1-28, 1991.
[49] E. Lawler, *Combinatorial Optimization: Networks and Matroids.* Holt, Reinhart and Winston, 1976.
[50] H. le Guen and R.A. Marie, "Visiting Probabilities in Non-Irreducible Markov Chains with Strongly Connected Components," *Proc. European Conf. Simulation and Modelling,* pp. 548-552, 2002.
[51] P. Linz, *An Introduction to Formal Languages and Automata.* Jones and Bartless, 2001.
[52] G. Liu and K.G. Ramakrishnan, "A*prune: An Algorithm for Finding k Shortest Paths Subject to Multiple Constraints," *Proc. INFOCOM,* pp. 743-749, 2001.
[53] A. McIver, C. Morgan, and C. Gonzalia, "Proofs and Refutations for Probabilistic Systems," *Proc. Technical Conf. Formal Methods,* pp. 100-115, 2008.
[54] K. Mehlhorn and M. Ziegelmann, "Resource Constrained Shortest Paths," *Proc. Int'l Conf. Embedded Systems and Applications,* pp. 326-337, 2000.
[55] C. Neumann, "Converting Deterministic Finite Automata to Regular Expressions," http://neumannhaus.com/christoph/papers/2005-03-16.DFA_to_RegEx.pdf, 2005.
[56] M.K. Reiter and A.D. Rubin, "Crowds: Anonymity for Web Transactions," *ACM Trans. Information and System Security,* vol. 1, no. 1, pp. 66-92, 1998.
[57] V. Shmatikov, "Probabilistic Analysis of an Anonymity System," *J. Computer Security,* vol. 12, nos. 3/4, pp. 355-377, 2004.
[58] S. Shoham and O. Grumberg, "A Game-Based Framework for CTL Counterexamples and 3-Valued Abstraction-Refinement," *Proc. Int'l Conf. Computer Aided Verification,* pp. 275-287, 2003.
[59] A. Vaha-Sipila and T. Virtanen, "BT-Crowds: Crowds-Style Anonymity with Bluetooth and Java," *Proc. Hawaii Int'l Conf. System Sciences,* 2005.
[60] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R.C. Carrasco, "Probabilistic Finite-State Machines-Part I," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 7, pp. 1013-1025, July 2005.
[61] A.J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Trans. Information Theory,* vol. 13, no. 2, pp. 260-269, 1967.

Tingting Han received the bachelor's degree in 2003 and the master's degree in 2006 in computer science, both from Nanjing University, China. She is currently a PhD student at the Software Modeling and Verification (MOVES) Group at RWTH Aachen University and in the Formal Methods and Tools Group at the University of Twente. Her research interests include specification and verification of probabilistic and stochastic systems, model checking, diagnostics, and process algebra.

Joost-Pieter Katoen has been a full professor at the RWTH Aachen University since 2004 and is affiliated with the University of Twente. His research interests are concurrency theory, model checking, timed and probabilistic systems, and semantics. He has coauthored more than 100 journal and conference papers, and recently published a comprehensive book (with Christel Baier) on *Principles of Model Checking*. In the list of 10,000 most cited authors in Computer Science of March 2008 as maintained by the NEC Research Institute (see citeseerx.ist.psu.edu), he is ranked at position 4383 with 958 citations. His h-index is 27. He is a member of the EPSRC Review College, IFIP WG 1.8 on Concurrency Theory, of the editorial board of the *Journal of Software*, and of the steering committees of ETAPS and QEST. He is a member of the IEEE Computer Society.

Berteun Damman is currently working toward the MS degree in Computer Science in Formal Methods and Tools Group at the University of Twente, The Netherlands, after having spent a year in the MOVES group at the RWTH in Aachen, Germany, to work on the theory behind counterexamples for PCTL and to extend the available toolset, which are his main research interests.