# Enforcing $\omega$-Regular Properties in Markov Chains by Restarting

**Javier Esparza**
Technische Universität München, Germany

**Stefan Kiefer**
University of Oxford, UK

**Jan Křetínský**
Technische Universität München, Germany

**Maximilian Weininger**
Technische Universität München, Germany

## Abstract

Restarts are used in many computer systems to improve performance. Examples include reloading a webpage, reissuing a request, or restarting a randomized search. The design of restart strategies has been extensively studied by the performance evaluation community. In this paper, we address the problem of designing universal restart strategies, valid for arbitrary finite-state Markov chains, that enforce a given $\omega$-regular property while not knowing the chain. A strategy enforces a property $\varphi$ if, with probability 1, the number of restarts is finite, and the run of the Markov chain after the last restart satisfies $\varphi$. We design a simple "cautious" strategy that solves the problem, and a more sophisticated "bold" strategy with an almost optimal number of restarts.

**2012 ACM Subject Classification** Theory of computation → Verification by model checking

**Keywords and phrases** Markov chains, omega-regular properties, runtime enforcement

**Digital Object Identifier** 10.4230/LIPIcs.CONCUR.2021.5

**Funding** This work was supported by the Research Training Group GRK 2428 CONVEY of the German Research Council (DFG) and by the ERC project PaVeS (Advanced Grant 787367).
*Stefan Kiefer*: supported by a Royal Society University Research Fellowship.

## 1 Introduction

Many computing systems offer the possibility to *restart* a computation or an interaction that is suspected to have failed, in order to improve performance. The standard example is the reload button of a web browser: when a download seems to "hang", pressing the button can lead to a faster, even to an immediate download. Similar situations appear in networks, where resending a message can lead to a faster acknowledgment, in randomized search, where the search can be restarted with a new seed, or in software rejuvenation, where performing a garbage collection, flushing caches, or simply rebooting may improve performance (see e.g. [7, 10, 12, 13, 20, 22]). Performance evaluation has extensively explored how to find optimal restart strategies in stochastic timed systems, usually under strong assumptions on the distribution of the times at which events occur.

In verification, *liveness properties* are abstractions of performance requirements: "every requested webpage will eventually be downloaded" is an abstraction of "every requested webpage will be downloaded within at most 3 seconds", or of some more complicated statement. The advantage is that they can be checked even when timing information is not available. However, to the best of our knowledge restart strategies for liveness properties have not been studied in the probabilistic, untimed setting. In this paper we study this question.

Consider a liveness property $\varphi$ like "every requested webpage will eventually be downloaded". Assume that the system is modelled by a finite-state Markov chain, including faulty behaviour of the TCP protocol, or the congestion phenomena causing requests to hang. We only know the initial state of the chain, and we can execute a probabilistic program that, given a state of the chain, returns a successor state according to the transition probabilities; we do not have access to the code of this program, or it could be very complex, and so we do not know the probabilities. We can monitor runs of the chain and record the sequence of states they visit; further, we are allowed to restart the system at any moment. The problem is to design a *universal* restart strategy, valid for every chain, satisfying the following two properties:

**(1)** With probability 1, the number of restarts is finite, and the run of the Markov chain after the last restart satisfies $\varphi$.[1]

**(2)** The expected number of restarts $R$, and the expected number of steps $S$ to a restart (conditioned on the occurrence of the restart) are not too high (we make this more precise later).

We say that strategies satisfying (1) *enforce* $\varphi$. If $\varphi$ has zero probability, then no strategy can enforce $\varphi$, and so we assume that $\varphi$ has non-zero probability. Under this assumption, it is easy to design naive enforcing strategies for safety and co-safety properties. For a safety property like $\mathbf{G}p$ it suffices to restart whenever the current state does not satisfy $p$; indeed, since $\mathbf{G}p$ has positive probability by assumption, eventually the chain executes a run satisfying $\mathbf{G}p$ almost surely, and this run is not aborted. Similarly, for $\mathbf{F}p$ we can abort the first execution after one step, the second after two steps etc., until a state satisfying $p$ is reached. Since $\mathbf{F}p$ has positive probability by assumption, at least one reachable state satisfies $p$, and we will almost surely visit it. However, these naive strategies lead to far too many restarts on average. Further, for reactivity properties like "every requested webpage will eventually be downloaded" even the problem of finding any enforcing strategy is already challenging: Unlike for $\mathbf{G}p$ and $\mathbf{F}p$, the strategy can never be sure that every extension of the current execution will satisfy the property, or that every extension will violate it.

Our first result shows that, perhaps surprisingly, ideas introduced in [6] on the detection of strongly connected components at runtime lead to a very simple enforcing strategy. Let $\mathcal{M}$ be the (unknown) Markov chain of the system, and let $\mathcal{A}$ be a deterministic Rabin automaton for $\varphi$. Say that a run of the product chain $\mathcal{M} \times \mathcal{A}$ is *good* if it satisfies $\varphi$, and *bad* otherwise. We define a set of *suspect* finite executions satisfying two properties:

**(a)** bad runs almost surely have a suspect prefix; and

**(b)** if the set of good runs has nonzero probability, then the set of runs without suspect prefixes also has nonzero probability.

The strategy restarts the chain whenever the current execution is suspect. We call it the *cautious strategy*, or, since it must be implemented by monitoring the system, the *cautious monitor*. By property (a) the cautious monitor aborts bad runs almost surely, and by property (b) almost surely the system eventually executes a run without suspect prefixes, which by (a) is necessarily good.

While the cautious monitor is very simple, it does not satisfy condition (2): in the worst case, both $R$ and $S$ are exponential in the number of states of the chain. A simple analysis shows that, without further information on the chain, the exponential dependence of $S$ on the number of states is unavoidable. However, the exponential dependence of $R$ on the number

[1] More precisely, the property is enforced only under the assumption that $\varphi$ has non-zero probability, otherwise there is no such strategy.

of states can be avoided: using a technique of [6], we define a *bold monitor* for which the expected number of restarts is almost optimal. Observe that if the property $\varphi$ has probability $\mathsf{p}_\varphi$, then the number of restarts needed by *any* monitor, even those that know the chain, is $1/\mathsf{p}_\varphi$, because, loosely speaking, that is the expected number of runs until the chain executes a run satisfying $\varphi$. Our bold monitor achieves $c + 1/(\mathsf{p}_\varphi(1-\epsilon))$ restarts on average for any given $\epsilon > 0$, where $c$ is a constant independent of both the chain and $\varphi$.

We also develop an efficient data structure that the monitor may use to keep track of the relevant parameters of the current run. Finally, we illustrate the behaviour of our monitors on models from the standard PRISM Benchmark Suite [15], and compare our theoretical bounds to the experimental values.

**Related work.** Our work is related to runtime enforcement [17]. In this approach a property is enforced by automatically constructing a monitor that inspects the execution online in an incremental way, and takes action whenever it violates the property; only safety properties are enforceable. The ideas of [17] have been extended to some non-safety properties and timed properties (see e.g. [3, 16, 8, 9]). To the best of our knowledge, restarts as runtime enforcers in a probabilistic setting have not been considered. Runtime monitoring of stochastic systems modelled as Hidden Markov Chains (HMM) has been studied by Sistla et al. in a number of papers [18, 11, 19], but these papers concentrate on monitoring violations of a property, not on enforcing it.

Our problem is also related to learning Markov chains, e.g., [5, 4, 21, 1]. A simple and correct restart strategy based on learning is to store *all* the states and transitions of the chain $\mathcal{M} \times \mathcal{A}$ visited so far, yielding a *currently explored graph*, and restart whenever the current state belongs to a non-accepting and non-trivial bottom strongly connected component of this graph. However, the memory consumption of such a strategy is very high. We focus on strategies that only store (part of) the current run, at lower memory cost. Moreover, many real systems exhibit a "fat but shallow" topology, i.e., a large ratio between the number of paths and their length [6]. In such chains, states are rarely revisited, and so storing all past states for eventual use in future runs is inefficient. Our strategies only need a few restarts; more precisely, the number of restarts only depends on the inverse of the probability of $\varphi$, but not on the size of the chain. These points are discussed in more detail in Remark 5 and in Section 4.3.

## 2 Preliminaries and running example

**Directed graphs.** A directed graph is a pair $G = (V, E)$, where $V$ is the set of vertices and $E \subseteq V \times V$ is the set of edges. A path (infinite path) of $G$ is a finite (infinite) sequence $\pi = v_0, v_1 \ldots$ of vertices such that $(v_i, v_{i+1}) \in E$ for every $i = 0, 1 \ldots$. We denote the empty path by $\lambda$ and concatenation of paths $\pi_1$ and $\pi_2$ by $\pi_1 \,.\, \pi_2$. A graph $G$ is strongly connected if for every two vertices $v, v'$ there is a path leading from $v$ to $v'$. A graph $G' = (V', E')$ is a subgraph of $G$, denoted $G' \preceq G$, if $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$; we write $G' \prec G$ if $G' \preceq G$ and $G' \neq G$. A graph $G' \preceq G$ is a strongly connected component (SCC) of $G$ if it is strongly connected and no graph $G''$ satisfying $G' \prec G'' \preceq G$ is strongly connected. An SCC $G' = (V', E')$ of $G$ is a bottom SCC (BSCC) if $v \in V'$ and $(v, v') \in E$ imply $v' \in V'$.

**Markov chains.** A *Markov chain (MC)* is a tuple $\mathcal{M} = (S, \mathbf{P}, \mu)$, where $S$ is a finite set of states, $\mathbf{P} : S \times S \to [0, 1]$ is the transition probability matrix, such that for every $s \in S$ it holds $\sum_{s' \in S} \mathbf{P}(s, s') = 1$, and $\mu$ is a probability distribution over $S$. The graph of $\mathcal{M}$ has $S$

as vertices and $\{(s,s') \mid \mathbf{P}(s,s') > 0\}$ as edges. Abusing language, we also use $\mathcal{M}$ to denote the graph of $\mathcal{M}$. We let $\mathsf{p_{min}} := \min(\{\mathbf{P}(s,s') > 0 \mid s,s' \in S\})$ denote the smallest positive transition probability in $\mathcal{M}$. A *run* of $\mathcal{M}$ is an infinite path $\rho = s_0 s_1 \cdots$ of $\mathcal{M}$; we let $\rho[i]$ denote the state $s_i$. Each (finite) path $\pi$ in $\mathcal{M}$ determines the set of runs $\mathsf{Cone}(\pi)$ consisting of all runs that start with $\pi$. To $\mathcal{M}$ we assign the probability space $(\mathsf{Runs}, \mathcal{F}, \mathbb{P})$, where $\mathsf{Runs}$ is the set of all runs in $\mathcal{M}$, $\mathcal{F}$ is the $\sigma$-algebra generated by all $\mathsf{Cone}(\pi)$, and $\mathbb{P}$ is the unique probability measure such that $\mathbb{P}[\mathsf{Cone}(s_0 s_1 \cdots s_k)] = \mu(s_0) \cdot \prod_{i=1}^{k} \mathbf{P}(s_{i-1}, s_i)$, where the empty product equals 1. The expected value of a random variable $f : \mathsf{Runs} \to \mathbb{R}$ is $\mathbb{E}[f] = \int_{\mathsf{Runs}} f \, d\mathbb{P}$.

Given a finite set $Ap$ of atomic propositions, a *labelled Markov chain* (LMC) is a tuple $\mathcal{M} = (S, \mathbf{P}, \mu, Ap, L)$, where $(S, \mathbf{P}, \mu)$ is a MC and $L : S \to 2^{Ap}$ is a labelling function. Given a labelled Markov chain $\mathcal{M}$ and an $\omega$-regular property $\varphi$, we are interested in the measure $\mathbb{P}[\mathcal{M} \models \varphi] := \mathbb{P}[\{\rho \in \mathsf{Runs} \mid L(\rho) \models \varphi\}]$, where $L$ is naturally extended to runs by $L(\rho)[i] = L(\rho[i])$ for all $i$.

In this work, we assume that $\mathcal{M}$, and in particular $S$, are unknown to the algorithms; when a simulation run is observed, we can distinguish whether the current state has already been seen and at which step, but we cannot match the state with any concrete element of $S$ even if we knew $\mathcal{M}$.

**Deterministic Rabin Automata and product Markov chain.** For every $\omega$-regular property (in particular, for every LTL property) $\varphi$ there is a *deterministic Rabin automaton* (DRA) $\mathcal{A} = (Q, 2^{Ap}, \gamma, q_o, Acc)$ that accepts all runs that satisfy $\varphi$ [2]. Here $Q$ is a finite set of states, $\gamma : Q \times 2^{Ap} \to Q$ is the transition function, $q_o \in Q$ is the initial state, and $Acc \subseteq 2^Q \times 2^Q$ is the acceptance condition.

The product of a MC $\mathcal{M}$ and DRA $\mathcal{A}$ is the Markov chain $\mathcal{M} \otimes \mathcal{A} = (S \times Q, \mathbf{P}', \mu')$, where $\mathbf{P}'((s,q),(s',q')) = \mathbf{P}(s,s')$ if $q' = \gamma(q, L(s'))$ and $\mathbf{P}'((s,q),(s',q')) = 0$ otherwise, and $\mu'(s,q) = \mu(s)$ if $\gamma(q_o, L(s)) = q$ and $\mu'(s,q) = 0$ otherwise. Note that $\mathcal{M} \otimes \mathcal{A}$ has the same smallest transition probability $\mathsf{p_{min}}$ as $\mathcal{M}$.

A run of $\mathcal{M} \otimes \mathcal{A}$ is *good* if it satisfies $\varphi$, i.e., if it is accepted by $\mathcal{A}$, and *bad* otherwise. An SCC $B$ of $\mathcal{M} \otimes \mathcal{A}$ is *good* if there exists a Rabin pair $(E,F) \in Acc$ such that $B \cap (S \times E) = \emptyset$ and $B \cap (S \times F) \neq \emptyset$. Otherwise, the SCC is *bad*. Observe that good runs of $\mathcal{M} \otimes \mathcal{A}$ almost surely reach a good BSCC (i.e., more formally, the probability that a run satisfies $\varphi$ and does not reach a good BSCC is 0), and bad runs almost surely reach a bad BSCC (i.e., more formally, the probability that a run does not satisfy $\varphi$ and does not reach a bad BSCC is also 0).

▶ **Example 1** (running). We present a strongly idealized but illustrative running example. Consider a concurrent system where multiple threads execute transactions (e.g., the database of an online shop). Every thread first acquires locks on a number of variables (which may dynamically depend on the values of the variables it has already acquired), then executes its task, and releases all locks. A thread may have to repeatedly try to acquire a lock on a given

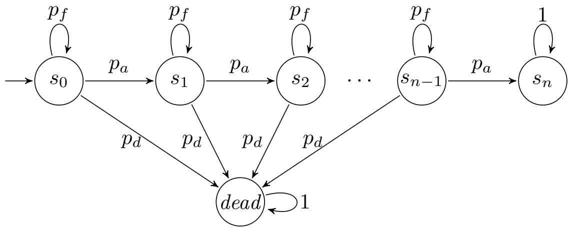

**Figure 1** An idealized example.

variable, since it can be currently held by another thread. Moreover a thread that acquires a lock on a variable, say $v$, reaches a deadlock if the next variable it needs, say $v'$, is currently held by another thread trying to get a lock on $v$.

The deadlock problem can be solved by maintaining a dependency graph of all transactions, and breaking all cyclic dependencies by restarting at least one of the threads involved. However, this graph is usually too large to be practical. Under the assumption that threads are scheduled stochastically, there are simpler mechanisms requiring no communication between threads: Each thread counts the number of attempts it makes at acquiring the same variable, and restarts (i.e., releases all its locks) when the number becomes "too high". We want to design such a mechanism, assuming that the thread has only the following information (or, equivalently, assumes that the following hypotheses are correct): (1) its interaction with the rest of the system can be modelled by a discrete finite-state Markov chain, whose states store the sequence of locks acquired by the thread so far; (2) the probability that an attempt to acquire a lock fails, and that it succeeds but leads to a deadlock only depend on the current state, not on the number of attempts; and (3) the probability of the runs of the chain in which the thread eventually acquires all locks it needs is positive. The thread does not know any probability, or even the number of locks it needs. At every moment in time, the thread can choose between trying to acquire the next lock, or restarting and releasing all locks. The challenge is to design a restarting strategy ensuring that the thread will execute its task with probability 1 while keeping the number of resets low. Observe that, even though the thread does not know the chain, it can tell at each step whether it stays in the same state, or moves to a new state it has not yet visited. Indeed, an attempt to acquire a lock either fails, in which case the thread stays in the same state, or succeeds, in which case it moves to a new state not visited so far, because the set of acquired locks is larger. In particular, the thread can maintain a list $k_0, k_1, k_2, \ldots$ indicating the number $k_i$ of visits to the $i$-th state. However, the thread does not know if the current state is a deadlock or not.

Consider a simple case in which the thread always needs locks on the same $n$ variables, the probability of acquiring the lock and not reaching a deadlock afterwards, acquiring the lock and reaching a deadlock, and failing to acquire the lock are $p_a$, $p_d$ and $p_f = 1 - (p_a + p_d)$. The interaction of the thread with the rest of the system is then modelled by the Markov chain depicted in Figure 1.

## 3 The cautious monitor

We assume the existence of a deterministic Rabin automaton $\mathcal{A} = (Q, 2^{Ap}, \gamma, q_o, Acc)$ for $\varphi$. Our monitors keep track of the path $\pi$ of the product chain $\mathcal{M} \otimes \mathcal{A}$ corresponding to the path of $\mathcal{M}$ executed so far. In order to present the cautious monitor we need some definitions.

**Candidate of a path.** Given a finite or infinite path $\rho = s_0 s_1 \cdots$ of $\mathcal{M} \otimes \mathcal{A}$, the *support* of $\rho$ is the set $\overline{\rho} = \{s_0, s_1, \ldots\}$. The *graph of* $\rho$ is $G_\rho = (\overline{\rho}, E_\rho)$, where $E_\rho = \{(s_i, s_{i+1}) \mid i = 0, 1, \ldots\}$; i.e., $G_\rho$ has exactly the vertices and edges of $\mathcal{M} \otimes \mathcal{A}$ explored by $\rho$.

Let $\pi$ be a finite path of $\mathcal{M} \otimes \mathcal{A}$. If $\pi$ has a suffix $\kappa$ such that $G_\kappa$ is a BSCC of $G_\pi$ (i.e., the suffix $\kappa$ of $\pi$ "looks like" the path $\pi$ has entered a BSCC), we call $\overline{\kappa}$ the *candidate of* $\pi$. Given a path $\pi$, we define $K(\pi)$ as follows: If $\pi$ has a candidate $\overline{\kappa}$, then $K(\pi) := \overline{\kappa}$; otherwise, $K(\pi) := \bot$, meaning that $K(\pi)$ is undefined.

▶ **Example 2.** We have $K(s_0 s_1 s_1) = \{s_1\}$, $K(s_0 s_1 s_1 s_2) = \bot$, and $K(s_0 s_1 s_1 s_2 s_2) = \{s_2\}$ (Figure 1). Also $K(s_0) = K(s_0 s_1) = K(s_0 s_0 s_1) = \bot$, $K(s_0 s_0) = \{s_0\}$, and $K(s_0 s_1 s_0 s_1) = \{s_0, s_1\}$ (Figure 2).

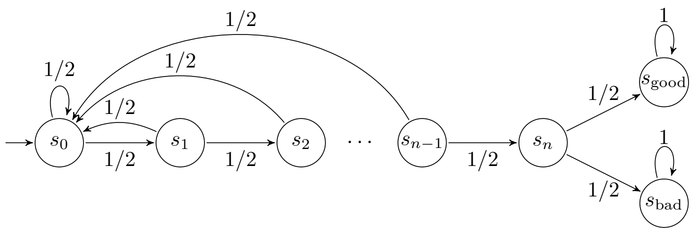

Figure 2 A family of Markov chains.

**Good and bad candidates.** A candidate $K$ is *good* if $\mathcal{A}$ has a Rabin pair $(E, F) \in Acc$ such that $K \cap (S \times E) = \emptyset$ and $K \cap (S \times F) \neq \emptyset$. Otherwise, $K$ is *bad.* A finite path $\pi$ of $\mathcal{M} \otimes \mathcal{A}$ is *suspect* if $K(\pi) \neq \bot$ and $K(\pi)$ is a bad candidate. The function SUSPECT($\pi$) returns **true** if $\pi$ is suspect, and **false** otherwise.

▶ **Proposition 3.**

**(a)** *Bad runs of $\mathcal{M} \otimes \mathcal{A}$ almost surely have a suspect finite prefix.*

**(b)** *If the good runs of $\mathcal{M} \otimes \mathcal{A}$ have nonzero probability, then the set of runs without suspect prefixes also has nonzero probability.*

▶ **Example 4** (running). Consider the running example from Figure 1, with $\{s_n\}$ and $\{dead\}$ as good and bad BSCC, respectively. The good paths of the chain are those reaching $\{s_n\}$. They have nonzero probability. The candidate of any bad path is either $\{s_i\}$ for some $0 \leq i \leq n-1$, or the bad BSCC. Since all these candidates are bad, every bad run is suspect. The only good run without suspect prefixes is the one that reaches $s_n$ without ever looping (i.e., all locks are acquired at the first attempt). It has probability $p_a^n$.

**The cautious monitor.** The cautious monitor is shown in Algorithm 1. The algorithm samples a run of $\mathcal{M} \otimes \mathcal{A}$ step by step, and restarts whenever the current path $\pi$ is suspect.

**Algorithm 1** CAUTIOUSMONITOR.

```
1: while true do
2:     $\pi \leftarrow \lambda$                         ▷ Initialize path with empty path
3:     repeat
4:         $\pi \leftarrow \pi\,.\,\mathsf{NextState}(\pi)$                  ▷ Extend path
5:     until SUSPECT($\pi$)
```

▶ Remark 5. The cautious monitor only needs to store the *first-appearance record* of $\pi$ (i.e., the list of states appearing in $\pi$, sorted according to their first appearance), since this is enough to compute SUSPECT($\pi$). Even further, it only needs the suffix of length mxsc of the record, where mxsc is the maximal size of the SCCs of the chain. So there exists a universal restart strategy that only needs to store mxsc states. Strategies based on learning the Markov chain, which store all states visited so far, also in previous runs, do not satisfy this property.

To state the correctness of Algorithm 1 formally, we define another Markov chain. Consider the infinite-state Markov chain $\mathcal{C}$ (for cautious) defined as follows. The states of $\mathcal{C}$ are pairs $\langle \pi, r\rangle$, where $\pi$ is a finite path of $\mathcal{M} \otimes \mathcal{A}$, and $r \geq 0$. Intuitively, $r$ counts the number of restarts so far. The initial probability distribution assigns probability 1 to $\langle \lambda, 0\rangle$, and 0 to all others. The transition probability matrix $\mathbf{P}_{\mathcal{C}}(\langle \pi, r\rangle, \langle \pi', r'\rangle)$ is defined as follows.

- If $\pi$ is suspect, then $\mathbf{P}_{\mathcal{C}}(\langle \pi, r\rangle, \langle \pi', r'\rangle) = 1$ if $\pi' = \lambda$ and $r' = r+1$, and 0 otherwise. We call such a transition a *restart.*
- If $\pi$ is not suspect, then $\mathbf{P}_{\mathcal{C}}(\langle \pi, r\rangle, \langle \pi', r'\rangle) = \mathbf{P}(p, p')$ if $r' = r$, $\pi = \pi'' . p$ and $\pi' = \pi . p'$, and 0 otherwise.

A run of CautiousMonitor corresponds to a run $\rho = \langle \pi_1, r_1\rangle\langle \pi_2, r_2\rangle \cdots$ of $\mathcal{C}$. Let $R$ be the random variable that assigns to $\rho$ the supremum of $r_1, r_2 \cdots$. Further, let $S_\varphi$ be the set of runs $\rho$ such that $R(\rho) < \infty$ and the suffix of $\rho$ after the last restart satisfies $\varphi$. The following theorem states that CautiousMonitor satisfies the properties that were mentioned in the introduction.

▶ **Theorem 6.** *Let $\varphi$ be an $\omega$-regular property such that $\mathbb{P}[\mathcal{M} \models \varphi] > 0$. Let $\mathcal{C}$ be the Markov chain obtained from $\mathcal{M}$ and $\varphi$ as above. We have (a) $\mathbb{P}_{\mathcal{C}}[R < \infty] = 1$, and (b) $\mathbb{P}_{\mathcal{C}}[S_\varphi \mid R < \infty] = \mathbb{P}_{\mathcal{C}}[S_\varphi] = 1$.*

**Proof.** Since $\mathbb{P}[\mathcal{M} \models \varphi] > 0$, the good runs of $\mathcal{M} \otimes \mathcal{A}$ have nonzero probability. By part (b) of Proposition 3, the set of runs of $\mathcal{M} \otimes \mathcal{A}$ without suspect prefixes also has nonzero probability, say $p$. So, by construction of $\mathcal{C}$, after each restart the probability that $\mathcal{C}$ executes a run of $\mathcal{M} \otimes \mathcal{A}$ without suspect prefixes, and so of executing no more restarts, is at least $p$. So $\mathbb{P}_{\mathcal{C}}[R = \infty] = 0$. ◀

**Performance.** Let $T$ be the random variable that assigns to $\rho$ the number of steps till the last restart, or $\infty$ if the number of restarts is infinite. First of all, we observe that, if we do not make any assumption on the system, $\mathbb{E}(T)$ can grow exponentially in the number of states of the chain. Indeed, consider the family of Markov chains of Figure 2 and the property $\mathbf{F}p$. Assume the only state satisfying $p$ is $s_{\text{good}}$. Then the product of each chain in the family with the DRA for $\mathbf{F}p$ is essentially the same chain, and the good runs are those reaching $s_{\text{good}}$. We show that $\mathbb{E}(T)$ grows at least exponentially for *any* monitor, even for those that have full knowledge of the chain. Indeed, since restarting brings the chain back to $s_0$, optimal monitors never do a restart when the chain is in any of the states $s_0, \ldots, s_n$; further, they always restart in state $s_{\text{bad}}$, because there is zero probability of reaching $s_{\text{good}}$. So the optimal monitor (i.e., the one with the least expected number of steps) is the one that restarts if, and only when, the chain reaches $s_{\text{bad}}$. Since the chain eventually reaches each of $s_{\text{good}}$ and $s_{\text{bad}}$ with probability 1/2, the probability that this monitor performs exactly $k$ restarts is $\frac{1}{2^{k+1}}$, and so the average number of restarts is 1. Hence, $\mathbb{E}(T)$ is the expected number of steps needed to reach $s_{\text{bad}}$, under the assumption that it is indeed reached. To reach $s_{\text{bad}}$ the chain must execute a run ending with the suffix $s_0 s_1 \ldots s_n$. Since the probability of executing that suffix is $1/2^n$, we get $\mathbb{E}(T) \geq 2^n$. We formulate this result as a proposition.

▶ **Proposition 7.** *Let $\mathcal{M}_n$ be the Markov chain of Figure 2 with $n$ states. Given a monitor $\mathcal{N}$ for the property $\mathbf{F}p$, let $T_{\mathcal{N}}$ be the random variable that assigns to a run of the monitor on $\mathcal{M}_n$ the number of steps till the last restart, or $\infty$ if the number of restarts is infinite. Then $\mathbb{E}(T_{\mathcal{N}}) \geq 2^n$ for every monitor $\mathcal{N}$.*

This shows that all monitors have bad performance when the time needed to traverse a non-bottom SCC of the chain is very large. So we conduct a parametric analysis in the maximal size $\mathsf{mxsc}$ of the SCCs of the chain. This reveals the weak point of CautiousMonitor: $\mathbb{E}(T)$ remains exponential even for families satisfying $\mathsf{mxsc} = 1$. Consider our running example in Figure 1. CautiousMonitor restarts whenever it takes any of the self-loops in a state $s_i$ for $i < n$. Indeed, after that the current path $\pi$ ends in $s_i s_i$, and so $K(\pi) = \{s_i\}$, which is a bad candidate. So after the last restart the chain must follow the path $s_0 s_1 s_2 \cdots s_n$, which has probability $p_a^n$, and so $\mathbb{E}(T) \geq p_a^{-n}$. Intuitively, the monitor introduced in the next section is "bolder"; instead of restarting whenever it does not immediately acquire the next lock, it only restarts after trying to acquire a certain number of times, making it "more likely" that the state is *dead*.

## 4 The bold monitor

We proceed in two steps. In Section 4.1, inspired by [6], we design a bold monitor that knows the minimum probability $\mathsf{p}_{\mathsf{min}}$ appearing in $\mathcal{M}$ (more precisely, a lower bound on it). In Section 4.2 we modify this monitor to produce another one that works correctly without any prior knowledge about $\mathcal{M}$, at the price of a performance penalty.

### 4.1 Chains with known minimal probability

The cautious monitor aborts a run if the current candidate is bad. In contrast, the bold monitor keeps track of the *strength* of the current candidate (defined below), a measure of how confident the monitor can be that the current candidate is a BSCC. In deciding whether a restart should be triggered, the bold monitor also takes into account how many candidates it has already seen since the last restart. Intuitively, the monitor becomes bolder over time, which prevents it from restarting too soon on the family of Figure 1, independently of the length of the chain. The monitor is designed so that it restarts almost all bad runs and only a fixed fraction $\varepsilon$ of the good runs. Lemma 11 below shows how to achieve this. We need some additional definitions.

**Strength of a candidate and strength of a path.** Let $\pi$ be a finite path of $\mathcal{M} \otimes \mathcal{A}$. The *strength of* $K(\pi)$ in $\pi$ is undefined if $K(\pi) = \bot$. Otherwise, write $\pi = \pi' \, s \, \kappa$, where $s \in S \times Q$ and $\pi'$ is the shortest prefix of $\pi$ such that $K(\pi' s) = K(\pi)$; the strength of $K(\pi)$ is the largest $k$ such that every state of $K(\pi)$ occurs at least $k$ times in $s\,\kappa$, and the last element of $s\,\kappa$ occurs at least $k+1$ times. Intuitively, if the strength is $k$ then every state of the candidate has been been exited at least $k$ times but, for technical reasons, we start counting only after the candidate is discovered. The function $\textsc{Str}(\pi)$ returns the strength of $K(\pi)$ if $K(\pi) \neq \bot$, and 0 otherwise.

▶ **Example 8.** The following table illustrates the definition of strength.

| $\pi$ | $K(\pi)$ | $\pi'$ | $s$ | $\kappa$ | $\textsc{Str}(\pi)$ |
|---|---|---|---|---|---|
| $p_0p_1$ | $\bot$ | $-$ | $-$ | $-$ | 0 |
| $p_0p_1p_1$ | $\{p_1\}$ | $p_0p_1$ | $p_1$ | $\epsilon$ | 0 |
| $p_0p_1p_1p_1$ | $\{p_1\}$ | $p_0p_1$ | $p_1$ | $p_1$ | 1 |
| $p_0p_1p_1p_1p_0$ | $\{p_0, p_1\}$ | $p_0p_1p_1p_1$ | $p_0$ | $\epsilon$ | 0 |
| $p_0p_1p_1p_1p_0p_1$ | $\{p_0, p_1\}$ | $p_0p_1p_1p_1$ | $p_0$ | $p_1$ | 0 |
| $p_0p_1p_1p_1p_0p_1p_0$ | $\{p_0, p_1\}$ | $p_0p_1p_1p_1$ | $p_0$ | $p_1p_0$ | 1 |
| $p_0p_1p_1p_1p_0p_1p_0p_1p_0$ | $\{p_0, p_1\}$ | $p_0p_1p_1p_1$ | $p_0$ | $p_1p_0p_1p_0$ | 2 |

▶ **Example 9** (running)**.** Assume that for the first variable, we need three tries before finally acquiring the lock. Thus, after looping twice in $s_0$, the strength of the candidate $\{s_0\}$ is 2. If afterwards all variables are acquired on the first try, there are no further candidates until reaching the good BSCC $\{s_n\}$. Then (according to the Markov chain), in state $s_n$ we loop infinitely often and the strength of this candidate goes to infinity in the limit.

**Sequence of candidates of a run.** Let $\rho = s_0 s_1 \cdots$ be a run of $\mathcal{M} \otimes \mathcal{A}$. Consider the sequence of random variables defined by $K(s_0 \dots s_j)$ for $j \geq 0$, and let $(K_i)_{i \geq 1}$ be the subsequence without undefined elements and with no repetition of consecutive elements. For example, for $\varrho = p_0 p_1 p_1 p_1 p_0 p_1 p_2 p_2 \cdots$, we have $K_1 = \{p_1\}$, $K_2 = \{p_0, p_1\}$, $K_3 = \{p_2\}$, etc. Given a run $\rho$ with a sequence of candidates $K_1, K_2 \dots, K_k$ (observe that this sequence is always finite), we call $K_k$ the final candidate. We define the *strength* of $K_i$ in $\rho$ as the supremum of the strengths of $K_i$ in all prefixes $\pi$ of $\rho$ such that $K(\pi) = K_i$. For technical convenience, we define $K_\ell := K_k$ for all $\ell > k$ and $K_\infty := K_k$. Observe that $\rho$ satisfies $\varphi$ iff its final candidate is good. The next lemma follows immediately from the definitions, and the fact that almost surely the runs of a finite-state Markov chain eventually get trapped in a BSCC and visit every state of it infinitely often.

▶ **Lemma 10.** *The final candidate of a run $\rho$ is almost surely a BSCC of $\mathcal{M} \otimes \mathcal{A}$. Moreover, for every $k$ there exists a prefix $\pi_k$ of $\rho$ such that $K(\pi_k)$ is the final candidate and $\textsc{Str}(\pi_k) \geq k$.*

**The bold monitor.** The bold monitor for chains with minimal probability $\mathsf{p_{min}}$ is shown in Algorithm 2. For every $\rho$ and $i \geq 1$, we define two random variables:

- $\textsc{Str}_i(\rho)$ is the strength of $K_i(\rho)$ in $\rho$;
- $\textsc{Bad}_i(\rho)$ is **true** if $K_i(\rho)$ is a bad candidate, and **false** otherwise.

Let $\alpha_{\mathsf{min}} := -1/\log(1 - \mathsf{p_{min}})$. The following lemma states that, for every $\alpha \geq \alpha_{\mathsf{min}}$ and $\varepsilon > 0$, the runs that satisfy $\varphi$ and in which some bad candidate, say $K_i$, reaches a strength of at least $\alpha(i - \log \varepsilon)$, have probability at most $\varepsilon \mathsf{p}_\varphi$. This leads to the following strategy for the monitor: when the monitor is considering the $i$-th candidate, abort only if the strength reaches $\alpha(i - \log \varepsilon)$.

▶ **Lemma 11.** *Let $\varphi$ be an $\omega$-regular property with positive probability $\mathsf{p}_\varphi$. For every Markov chain $\mathcal{M}$ with minimal probability $\mathsf{p_{min}}$, for every $\alpha \geq \alpha_{\mathsf{min}}$ and $\varepsilon > 0$:*

$$\mathbb{P}\left[\ \{\rho \mid \rho \models \varphi \wedge \exists i \geq 1\,.\, \textsc{Bad}_i(\rho) \wedge \textsc{Str}_i(\rho) \geq \alpha(i - \log \varepsilon)\}\ \right] \leq \varepsilon \mathsf{p}_\varphi$$

The monitor is parametric in $\alpha$ and $\varepsilon$. The variable $C$ stores the current candidate, and is used to detect when the candidate changes. The variable $i$ maintains the index of the current candidate, i.e., in every reachable configuration of the algorithm, if $C \neq \bot$ then $C := K_i$.

**Algorithm 2** $\textsc{BoldMonitor}_{\alpha,\epsilon}$.

```
1: while true do
2:     $\pi \leftarrow \lambda$                                    ▷ Initialize path
3:     $C \leftarrow \bot$, $i \leftarrow 0$            ▷ Initialize candidate and candidate counter
4:     repeat
5:         $\pi \leftarrow \pi\,.\,\mathsf{NextState}(\pi)$                   ▷ Extend path
6:         if $\bot \neq K(\pi) \neq C$ then
7:             $C \leftarrow K(\pi)$; $i \leftarrow i + 1$     ▷ Update candidate and candidate counter
8:     until $\textsc{Suspect}(\pi)$ and $\textsc{Str}(\pi) \geq \alpha(i - \log \varepsilon)$
```

The infinite-state Markov chain $\mathcal{B}$ of the bold monitor is defined as the chain $\mathcal{C}$ for the cautious monitor; we just replace the condition that $\pi$ is suspect (and thus has strength at least 1) by the condition that $K(\pi)$ is bad and has strength $\geq \alpha(i - \log \varepsilon)$. The random variable $R$ and the event $S_\varphi$ are also defined as for $\textsc{CautiousMonitor}$.

▶ **Theorem 12.** *Let $\mathcal{M}$ be a finite-state Markov chain with minimum probability $\mathsf{p_{min}}$, and let $\varphi$ be an $\omega$-regular property with probability $\mathsf{p}_\varphi > 0$ in $\mathcal{M}$. Let $\mathcal{B}$ be the Markov chain, defined as above, corresponding to the execution of* BoldMonitor$_{\alpha,\epsilon}$ *on $\mathcal{M} \otimes \mathcal{A}$, where $\alpha \geq \alpha_{\mathsf{min}}$ and $\varepsilon > 0$. We have:*

**(a)** *The random variable $R$ is geometrically distributed, with parameter (success probability) at least $\mathsf{p}_\varphi(1-\varepsilon)$. Hence, we have $\mathbb{P}_\mathcal{B}[R < \infty] = 1$ and $\mathbb{E}_\mathcal{B}(R) \leq 1/\mathsf{p}_\varphi(1-\varepsilon)$.*

**(b)** *$\mathbb{P}_\mathcal{B}[S_\varphi \mid R < \infty] = \mathbb{P}_\mathcal{B}[S_\varphi] = 1$.*

**Proof.** (a) By Lemma 10, almost all bad runs are restarted. By Lemma 11, runs, conditioned under being good, are restarted with probability at most $\varepsilon$. It follows that the probability that a run is good and not restarted is at least $\mathsf{p}_\varphi(1-\varepsilon)$. (b) In runs satisfying $R < \infty$, the suffix after the last restart almost surely reaches a BSCC of $\mathcal{M} \otimes \mathcal{A}$ and visits all its states infinitely often, increasing the strength of the last candidate beyond any bound. So runs satisfying $R < \infty$ belong to $S_\varphi$ with probability 1. ◀

In particular, after each restart, the probability that no further restart occurs is at least $\mathsf{p}_\varphi(1-\varepsilon)$. The theoretical optimum would be $\mathsf{p}_\varphi$, as bad runs need to be restarted almost surely.

**Performance.** Recall that $T$ is the random variable that assigns to a run the number of steps until the last restart. Let $T_j$ be the number of steps between the $j$-th and $(j+1)$-st restart. Observe that all the $T_j$ are identically distributed. We have $T_j = T_j^\perp + T_j^C$, where $T_j^\perp$ and $T_j^C$ are the number of prefixes $\pi$ such that $K(\pi) = \perp$ (no current candidate) and $K(\pi) \neq \perp$ (a candidate), respectively. By deriving bounds on $\mathbb{E}(T_j^\perp)$ and $\mathbb{E}(T_j^C)$, we obtain:

▶ **Theorem 13.** *Let $\mathcal{M}$ be a finite-state Markov chain with minimum probability $\mathsf{p_{min}}$. Let $\varphi$ be an $\omega$-regular property with probability $\mathsf{p}_\varphi > 0$ in $\mathcal{M}$. Suppose the product $\mathcal{M} \otimes \mathcal{A}$ has $n$ states and maximal SCC size $\mathsf{mxsc}$. Let $\alpha \geq \alpha_{\mathsf{min}}$ and $\varepsilon > 0$. Let $T$ be the number of steps taken by* BoldMonitor$_{\alpha,\epsilon}$ *until the last restart (or $\infty$ if there are infinitely many restarts, or $0$ if there is no restart). We have:*

$$\mathbb{E}(T) \leq \frac{1}{\mathsf{p}_\varphi(1-\varepsilon)} \cdot 2n\,\alpha\,(n - \log \varepsilon)\,\mathsf{mxsc}\left(\frac{1}{\mathsf{p_{min}}}\right)^{\mathsf{mxsc}}. \tag{1}$$

Observe the main difference with CautiousMonitor: Instead of the exponential dependence on $n$ of Proposition 7, we only have an exponential dependence on $\mathsf{mxsc}$. So if $\mathsf{mxsc} \ll n$ the bold monitor performs much better than the cautious one.

## 4.2 General chains

We adapt BoldMonitor so that it works for arbitrary finite-state Markov chains, at the price of a performance penalty. The main idea is very simple: given any non-decreasing sequence $\{\alpha_n\}_{n=1}^\infty$ of natural numbers such that $\alpha_1 = 1$ and $\lim_{n\to\infty} \alpha_n = \infty$, we sample as in BoldMonitor$_{\alpha,\epsilon}$ but, instead of using the same value $\alpha$ for every sample, we use $\alpha_j$ for the $j$-th sample. See Algorithm 3, where Sample($\alpha$) is the body of the while loop of BoldMonitor$_{\alpha,\varepsilon}$ (lines 2–8 of Algorithm 2) for a given value of $\alpha$. The intuition is that $\alpha_j \geq \alpha_{\mathsf{min}}$ holds from some index $j_0$ onwards, and so, by the previous analysis, after the $j_0$-th restart the monitor almost surely only executes a finite number of restarts. More formally, the correctness follows from the following two properties.

**Algorithm 3** BoldMonitor$_\varepsilon$ for $\{\alpha_n\}_{n=1}^\infty$.

```
1: $j \leftarrow 0$
2: while true do
3:     $j = j + 1$
4:     Sample$(\alpha_j)$
```

- For every $j \geq 1$, if Sample$(\alpha_j)$ does not terminate then it executes a good run almost surely.
  Indeed, if Sample$(\alpha_j)$ does not terminate then it almost surely reaches a BSCC of $\mathcal{M} \otimes \mathcal{A}$ and visits all its states infinitely often. So from some moment on $K(\pi)$ is and remains equal to this BSCC, and Str$(\pi)$ grows beyond any bound. Since Sample$(\alpha_j)$ does not terminate, the BSCC is good, and it executes a good run.
- If $\alpha_j \geq \alpha_{\mathsf{min}}$ then the probability that Sample$(\alpha_j)$ does not terminate is at least $\varepsilon \mathsf{p}_\varphi$.
  Indeed, by Lemma 11, if $\alpha_j \geq \alpha_{\mathsf{min}}$, the probability is already at least $\varepsilon \mathsf{p}_\varphi$. Increasing $\alpha$ strengthens the exit condition of the until loop. So the probability that the loop terminates is lower, and the probability of non-termination higher.

These two observations immediately lead to the following proposition:

▶ **Proposition 14.** *Let $\mathcal{M}$ be an arbitrary finite-state Markov chain, and let $\varphi$ be an $\omega$-regular property such that $\mathsf{p}_\varphi > 0$. Let $\mathcal{B}$ be the Markov chain corresponding to the execution of BoldMonitor$_\varepsilon$ on $\mathcal{M} \otimes \mathcal{A}$ with sequence $\{\alpha_n\}_{n=1}^\infty$. Let $\mathsf{p}_{\mathsf{min}}$ be the minimum probability of the transitions of $\mathcal{M}$ (which is unknown to BoldMonitor$_\varepsilon$). We have*

**(a)** $\mathbb{P}_\mathcal{B}[R < \infty] = 1$.
**(b)** $\mathbb{P}_\mathcal{B}[S_\varphi | R < \infty] = \mathbb{P}_\mathcal{B}[S_\varphi] = 1$.
**(c)** $\mathbb{E}(R) \leq j_{min} + 1/\mathsf{p}_\varphi(1-\epsilon)$, *where $j_{min}$ is the smallest index $j$ such that $\alpha_j \geq \alpha_{\mathsf{min}}$.*

**Performance.** Different choices of the sequence $\{\alpha_n\}_{n=1}^\infty$ lead to versions of BoldMonitor$_\varepsilon$ with different performance features. Intuitively, if the sequence grows very fast, then $j_{\min}$ is very small, and the expected number of restarts $\mathbb{E}(R)$ is only marginally larger than the number for the case in which the monitor knows $\mathsf{p}_{\mathsf{min}}$. However, in this case the last $1/\mathsf{p}_\varphi(1-\epsilon)$ aborted runs are performed for very large values $\alpha_j$, and so they take many steps. If the sequence grows slowly, then the opposite happens; there are more restarts, but aborted runs have shorter length. Let us analyze two extreme cases: $\alpha_j := 2^j$ and $\alpha_j := j$.

Denote by $f(\alpha)$ the probability that a run is restarted, i.e., the probability that a call Sample$(\alpha)$ terminates. Let $g(\alpha)$ further denote the expected number of steps done in Sample$(\alpha)$ of a run that is restarted (taking the number of steps as 0 if the run is not restarted). According to the analysis underlying Theorem 13, for $\alpha \geq \alpha_{\mathsf{min}}$ we have $g(\alpha) \leq c\alpha$ with $c := 2n\,(n - \log \varepsilon)\,\mathsf{mxsc}\,\mathsf{p}_{\mathsf{min}}^{-\mathsf{mxsc}}$. We can write $T = T_1 + T_2 + \cdots$, where $T_j = 0$ when either the $j$-th run or a previous run is not restarted, and otherwise $T_j$ is the number of steps of the $j$-th run. For $j \leq j_{\min}$ we obtain $\mathbb{E}(T_j) \leq g(\alpha_{j_{\min}})$ and hence we have:

$$\mathbb{E}(T) \;=\; \sum_{j=0}^{\infty} \mathbb{E}(T_j) \;\leq\; j_{\min} g(\alpha_{j_{\min}}) + \sum_{i=0}^{\infty} f(\alpha_{j_{\min}})^i g(\alpha_{j_{\min}+i}).$$

By Theorem 12(a) we have $f(\alpha_{j_{\min}}) \leq 1 - \mathsf{p}_\varphi(1-\varepsilon)$. It follows that choosing $\alpha_j := 2^j$ does not in general lead to a finite bound on $\mathbb{E}(T)$. Choosing instead $\alpha_j := j$, we get

$$\mathbb{E}(T) \;\leq\; c j_{\min}^2 + \sum_{i=0}^{\infty} (1 - \mathsf{p}_\varphi(1-\varepsilon))^i c(j_{\min} + i) \quad\leq\; \left( j_{\min}^2 + \frac{j_{\min}}{\mathsf{p}_\varphi(1-\varepsilon)} + \frac{1}{(\mathsf{p}_\varphi(1-\varepsilon))^2} \right) c\,,$$

where $j_{\min}$ can be bounded by $j_{\min} \leq -1/\log(1-\mathsf{p}_{\mathsf{min}}) + 1 \leq 1/\mathsf{p}_{\mathsf{min}}$. So with $c = 2n(n - \log\varepsilon)\mathsf{mxsc}\,\mathsf{p}_{\mathsf{min}}^{-\mathsf{mxsc}}$ we arrive at

$$\mathbb{E}(T) \leq \left(\frac{1}{\mathsf{p}_{\mathsf{min}}^2} + \frac{1}{\mathsf{p}_{\mathsf{min}}\mathsf{p}_\varphi(1-\varepsilon)} + \frac{1}{(\mathsf{p}_\varphi(1-\varepsilon))^2}\right) 2n\,(n - \log\varepsilon)\,\mathsf{mxsc}\left(\frac{1}{\mathsf{p}_{\mathsf{min}}}\right)^{\mathsf{mxsc}}, \tag{2}$$

a bound broadly similar to the one from Theorem 13, but with the monitor not needing to know $\mathsf{p}_{\mathsf{min}}$.

## 4.3 Implementing the bold monitor

A straightforward implementation of the bold monitor in which the candidate $K(\pi)$ and its strength are computed anew each time the path is extended is very inefficient. We present a far more efficient algorithm that continuously maintains the current candidate and its strength. Maintaining them until the path has length $n$ takes $O(n \log n)$ time and $O(s_n \log s_n)$ memory, where $s_n$ denotes the number of states visited by $\pi$ (which can be much smaller than $n$ when states are visited multiple times). So one update takes $O(\log n)$ amortized time.

Let $\pi$ be a path of $\mathcal{M} \otimes \mathcal{A}$, and let $s \in \pi$. (Observe that $s$ now denotes a state of $\mathcal{M} \otimes \mathcal{A}$, not of $\mathcal{M}$.) We let $G_\pi = (V_\pi, E_\pi)$ denote the subgraph of $\mathcal{M} \otimes \mathcal{A}$ where $V_\pi$ and $E_\pi$ are the sets of states and edges visited by $\pi$, respectively. Intuitively, $G_\pi$ is the fragment of $\mathcal{M} \otimes \mathcal{A}$ explored by the path $\pi$.

In the following we define some additional notions related to $G_\pi$. Afterwards we show how to use these notions to keep track of the current candidate and its strength during the operation of the bold monitor.

- The *discovery index* of a state $s$, denoted by $d_\pi(s)$, is the number of states that appear in the prefix of $\pi$ ending with the first occurrence of $s$. Intuitively, $d_\pi(s) = k$ if $s$ is the $k$-th state discovered by $\pi$. Since different states have different discovery times, and they do not change when the path is extended, we also call $d_\pi(s)$ the *identifier* of $s$.
- A *root* of $G_\pi$ is a state $r \in V_\pi$ such that $d_\pi(r) \leq d_\pi(s)$ for every state $s \in \mathsf{SCC}_\pi(r)$, where $\mathsf{SCC}_\pi(r)$ denotes the SCC of $G_\pi$ containing $r$. Intuitively, $r$ is the first state of $\mathsf{SCC}_\pi(r)$ visited by $\pi$.
- The *root sequence* $R_\pi$ of $\pi$ is the sequence of roots of $G_\pi$, ordered by ascending discovery index.
- Let $R_\pi = r_1\, r_2 \cdots r_m$ be the root sequence of $\pi$. The sequence $S_\pi = S_\pi(r_1)\, S_\pi(r_2) \cdots S_\pi(r_m)$ of sets is defined by $S_\pi(r_i) := \{s \in V_\pi \mid d_\pi(r_i) \leq d_\pi(s) < d_\pi(r_{i+1})\}$ for every $1 \leq i < m$, i.e., $S_\pi(r_i)$ is the set of states discovered after $r_i$ (including $r_i$) and before $r_{i+1}$ (excluding $r_{i+1}$); and $S_\pi(r_m) := \{s \in V_\pi \mid d_\pi(r_m) \leq d_\pi(s)\}$.
- $\mathit{Birthday}_\pi$ is defined as $\bot$ if $K(\pi) = \bot$, and as the length of the shortest prefix $\pi'$ of $\pi$ such that $K(\pi') = K(\pi)$ otherwise. Intuitively, $\mathit{Birthday}_\pi$ is the time at which the current candidate of $\pi$ was created.
- For every state $s$ of $\pi$, let $\pi_s$ be the longest prefix of $\pi$ ending at $s$. We define $\mathit{Visits}_\pi(s)$ as the pair $(\mathit{Birthday}_{\pi_s}, v)$, where $v$ is 0 if $\mathit{Birthday}_{\pi_s} = \bot$, and $v$ is the number of times $\pi_s$ has visited $s$ since $\mathit{Birthday}_{\pi_s}$ otherwise. We define a total order on these pairs: $(b, v) \preceq (b', v')$ iff $b > b'$ (where $\bot > n$ for every number $n$), or $b = b'$ and $v \leq v'$. Observe that, if $\pi$ has a candidate, then the smallest pair w.r.t. $\preceq$ corresponds to the state that is among the states visited since the creation of the candidate, and has been visited the least number of times.

The following lemma is an immediate consequence of the definitions.

▶ **Lemma 15.** *Let $G_\pi = (V_\pi, E_\pi)$. The SCCs of $G_\pi$ are the sets of $S_\pi$. Let $S_\pi(r_m)$ be the last set of $S_\pi$, and let $(b, v) = \min\{\mathit{Visits}_\pi(s) \mid s \in S_\pi(r_m)\}$, where the minimum is w.r.t. $\preceq$. We have $K(\pi) = \bot$ iff $b = \bot$, and if $b \neq \bot$ then $\textsc{Str}(\pi) = v$.*

By the lemma, in order to efficiently implement BoldMonitor it suffices to maintain $R_\pi$, $S_\pi$, and a mapping $\mathit{Visits}_\pi$ that assigns $\mathit{Visits}_\pi(s)$ to each state $s$ of $\pi$. More precisely, assume that the current path $\pi$ leads to a state $s$, and now it is extended to $\pi' = \pi \cdot s'$ by traversing a transition $s \to s'$ of $\mathcal{M} \otimes \mathcal{A}$; it suffices to compute $R_{\pi'}$, $S_{\pi'}$ and $\mathit{Visits}_{\pi'}$ from $R_\pi$, $S_\pi$, and $\mathit{Visits}_{\pi'}$ in $O(\log n)$ amortized time, where $n$ is the length of $\pi$. We first show how to update $R_\pi$, $S_\pi$, and $\mathit{Visits}_\pi$, and then describe data structures to maintain them in $O(\log n)$ amortized time. We consider four cases:

**(1)** $s' \notin V_\pi$. That is, the monitor discovers $s'$ by traversing $s \to s'$. Then the SCCs of $G_{\pi'}$ are those of $G_\pi$, plus a new trivial SCC containing only $s'$, with $s'$ as root. So $R_{\pi'} = R_\pi \cdot s'$ and $S_{\pi'} = S_\pi \cdot \{s'\}$. Since $s'$ has just been discovered, there is no candidate, and so $\mathit{Visits}_{\pi'}(s') = (\bot, 0)$.

**(2)** $s' \in V_\pi$ and $d_\pi(s) < d_\pi(s')$. That is, the monitor had already discovered $s'$, and it had discovered it after $s$. Then $G_{\pi'} = (V_\pi, E_\pi \cup \{(s, s')\})$, but the SCCs of $G_\pi$ and $G_{\pi'}$ coincide, and so $R_{\pi'} = R_\pi$, $S_{\pi'} = S_\pi$, and if $\mathit{Visits}_\pi(s') = (b, v)$, then $\mathit{Visits}_{\pi'}(s') = (b, v + 1)$.

**(3)** $s' \in V_\pi$ and $d_\pi(s) = d_\pi(s')$, i.e., $s' = s$. Then as before the SCCs of $G_\pi$ and $G_{\pi'}$ coincide, and so $R_{\pi'} = R_\pi$, $S_{\pi'} = S_\pi$. If $\mathit{Visits}_\pi(s) = (\bot, 0)$ then $\mathit{Visits}_{\pi'}(s) = (n + 1, 1)$ (recall that $n$ is the length of $\pi$), and if $\mathit{Visits}_\pi(s) = (b, v)$ for $b \neq \bot$ then $\mathit{Visits}_{\pi'}(s) = (b, v + 1)$.

**(4)** $s' \in V_\pi$ and $d_\pi(s) > d_\pi(s')$. That is, the monitor discovered $s'$ before $s$. Let $R_\pi = r_1\, r_2 \cdots r_m$ and let $r_i$ be the root of $\mathsf{SCC}_\pi(s')$. Then $G_{\pi'}$ has a path $r_i \xrightarrow{*} r_m \xrightarrow{*} s \to s' \xrightarrow{*} r_i$. So, if $\mathit{Visits}_\pi(s') = (b, v)$, we get

$$R_{\pi'} = r_1\, r_2 \cdots r_i \quad S_{\pi'} = S_\pi(r_1) \cdots S_\pi(r_{i-1}) \left(\bigcup_{j=i}^{m} S_\pi(r_j)\right) \quad \mathit{Visits}_{\pi'}(s') = (b, v + 1)$$

In order to efficiently update $R_\pi$, $S_\pi$ and $\mathit{Visits}_\pi$ we represent them using the following data structures.

- The number $N$ of different states visited so far.
- A hash map $D$ that assigns to each state $s$ discovered by $\pi$ its discovery index. When $s$ is visited for the first time, $D(s)$ is set to $N + 1$. Subsequent lookups return $N + 1$.
- A structure $R$ containing the identifiers of the roots of $R_\pi$, and supporting the following operations in amortized $O(\log n)$ time: insert$(r)$, which inserts the identifier of $r$ in $R$; extract-max, which returns the largest identifier in $R$; and find$(s)$, which returns the largest identifier of $R$ smaller than or equal to the identifier of $s$. (This is the identifier of the root of the SCC containing $s$.) For example, this is achieved by implementing $R$ both as a balanced search tree and a heap.
- For each root $r$ a structure $S(r)$ representing a set of states and a map assigning to each state $s$ the key $\mathit{Visits}_\pi(s)$, and supporting the following operations in amortized $O(\log n)$ time: find-min, which returns the minimum value of $\mathit{Visits}_\pi(s)$ over the states of $S(r)$; increment-key$(s)$, which increases the value of the second component of $\mathit{Visits}_\pi(s)$ by 1, and merge, which returns the union of two given maps. For example, this is achieved by implementing $S(r)$ as a Fibonacci heap.

▶ **Proposition 16.** *$R_\pi$, $S_\pi$ and $\mathit{Visits}_\pi$ can be updated in $O(\log n)$ amortized time using $N$, $D$, $R$, and $S$ as data structures.*

▶ Remark 17. The implementation above needs $O(s_n \log s_n)$ memory, where $s_n$ denotes the number of states visited by $\pi$. Applying the same trick as for the cautious monitor (see Remark 5), the memory consumption can be reduced to $O(\mathsf{mxsc} \log s_n)$, where $\mathsf{mxsc}$ is the maximal size of the SCCs of the chain.

## 5 Experimental results

In this section, we illustrate the behaviour of our monitors on concrete models from the standard PRISM Benchmark Suite [15] and compare the obtained theoretical bounds to the experimental values.

For our implementation, we have re-used the code provided in [6], which in turn is based on the PRISM model checker [14]. Note that whenever the obtained candidate is a good BSCC, no restart will ever happen and we can safely terminate the experiment.

**Models.** Table 1 lists the models and the properties type, in order of increasing satisfaction probability, which corresponds to increasing optimal number of restarts. The Table contains models from the standard PRISM benchmark suite [15]; the gridworld example, also with the corresponding property negated (written as $\overline{\text{gridworld}}$), in order to represent cases with also very low $\mathsf{p}_\varphi$; and the example of [6, Fig. 4], called "scale" here, with parameter 10.

**Table 1** List of models and types of properties considered. We give the satisfaction probability $\mathsf{p}_\varphi$, size $|S|$ minimum transition probability $\mathsf{p}_{\mathsf{min}}$, number of bottom SCCs, total number of states in them (an upper bound on $\mathsf{mxsc}$), and number of non-bottom SCCs (NSCC). For the model crowds, the "–" denotes the time-out of the PRISM model checker after 2 hours when trying to compute the characteristics.

| Model | Property | $\mathsf{p}_\varphi$ | $\|S\|$ | $\mathsf{p}_{\mathsf{min}}$ | #BSCC | BSCC-states | #NSCC |
|---|---|---|---|---|---|---|---|
| $\overline{\text{gridworld}}$ | $\neg(\mathbf{GF} \to \mathbf{FG})$ | 0.09 | 309 327 | 0.001 | 98 | 238 053 | 3 |
| nand | **GF** | 0.15 | 7 014 252 | 0.02 | 51 | 51 | 0 |
| bluetooth | **GF** | 0.20 | 143 291 | 0.008 | 3 072 | 3,072 | 24 |
| $\text{scale}_{10}$ | **GF** | 0.50 | 121 | 0.5 | 2 | 100 | 20 |
| crowds | **FG** | – | 10 633 591 | 0.067 | – | – | – |
| gridworld | $\mathbf{GF} \to \mathbf{FG}$ | 0.91 | 309 327 | 0.001 | 98 | 244 902 | 1 |
| hermann | **FG** | 1.00 | 524 288 | 1.9e-6 | 1 | 38 | 9 |

**Compared monitors.** The comparison on these models is performed for the following monitors. Firstly, we consider the cautious monitor, which corresponds to the fixed candidate strength 1. Moreover, we also consider the straightforward modification, which requires a different (but still constant) strength, here 10, to practically alleviate the issue with frequent premature restarts. We call these monitors $\text{Cautious}_1$ and $\text{Cautious}_{10}$. Secondly, we consider the bold monitor using knowledge of the minimal probability $\mathsf{p}_{\mathsf{min}}$, once with low precision of $\varepsilon = 0.5$ and once with a higher one of $\varepsilon = 0.1$, called $\text{Bold}_{0.5}$ and $\text{Bold}_{0.1}$. Finally, we list the optimal expected number of restarts, corresponding to an ideal (omniscient) monitor, which always makes the correct decision. Note that in our experiments sometimes the average number of restarts is slightly lower than the optimum, since when computing the empirical average of 100 runs, it can vary around the expected value to some extent.

**Interpretation of results.** Table 2 compares the efficiency of the monitors on the models, in terms of (i) the number of restarts until the monitors leaves the system uninterrupted as it generates a satisfying path, and (ii) the number of steps taken until the last restart when the infinite satisfying run starts. Due to the hugely varying characteristics of the models, we also provide detailed comments on the results model by model in Appendix B.

**Table 2** Experimental comparison of the monitors, showing the average number of restarts : average total length of all runs until the final restart. The average is taken over 100 runs of the algorithm. We compare cautious monitors with required candidate strengths 1 and 10, bold monitor with precisions 0.5 and 0.1 and the expected number of restarts for the omniscient monitor. Time-outs are set to two hours, but whenever the monitor finishes successfully, it so happens within a single minute for all considered models. For those models where a time-out occurs, we double-checked three times and the time-out always occurred. Then we ran 100 experiments for one minute and report the average number of restarts per minute.

| | $\overline{\text{gridw.}}$ | nand | bluetoooth | $\text{scale}_{10}$ | crowds | gridw. | hermann |
|---|---|---|---|---|---|---|---|
| $\text{Cautious}_1$ | $TO\ \frac{14000}{\text{min}}$ | 6.1 : 8 510 | 4.2 : 2 105 | 2033 : 6 131 | 0.98 : 96 | 0.08 : 3 814 | $TO\ \frac{1500}{\text{min}}$ |
| $\text{Cautious}_{10}$ | $TO\ \frac{3}{\text{min}}$ | 4.9 : 6 900 | 4.8 : 2 425 | 0.97 : 3 670 | 1.0 : 101 | 0.09 : 26 361 | $TO\ \frac{1400}{\text{min}}$ |
| $\text{Bold}_{0.5}$ | $TO\ \frac{0}{\text{min}}$ | 5.3 : 8 949 | 4.2 : 4 851 | 2.4 : 15 323 | 1.0 : 220 | $TO\ \frac{0}{\text{min}}$ | 0 |
| $\text{Bold}_{0.1}$ | $TO\ \frac{0}{\text{min}}$ | 5.8 : 10 583 | 3.7 : 4 637 | 1.3 : 14 528 | 1.0 : 199 | $TO\ \frac{0}{\text{min}}$ | 0 |
| $(1/\mathsf{p}_\varphi) - 1$ | 9.9 | 5.7 | 4.1 | 1.0 | ? | 0.1 | 0 |

In summary, the following phenomena can be observed:

- In cases with small BSCCs and few NSCCs (non-bottom SCCs), such as nand or bluetooth, all monitors manage to refute bad BSCCs and find the good one.
- Large BSCCs, as in gridworlds, give a hard time to all monitors. Interestingly, premature restarts of Cautious monitors may quickly yield another chance to hit a good BSCC, where others take long in bad BSCCs to get high confidence to restart. In contrast, premature restarts in intermediate candidates make Cautious monitors keep restarting even in the cases where every simulation goes to the good BSCC, as in hermann.
- Many NSCCs make $\text{Cautious}_1$ restart too often. However, if there is a good chance of leaving the NSCC soon, as in $\text{scale}_{10}$, then the simple modification to $\text{Cautious}_{10}$ fixes the issue.
- Since crowds has more than ten million states the PRISM model checker times out after two hours, not yielding information beyond the size. All the simulations are quite short here, indicating the frequent pattern of "fat but shallow" systems. None of the monitors experiences any difficulties, indicating essentially optimal number of restarts (around 1), hence the property seems to be satisfied with roughly 50% probability. The model checker could not conclude that. While the large size prevents a thorough numeric analysis, it did not prevent our monitors from determining satisfaction on single runs.

## 6 Conclusions

We have presented a universal restart strategy for enforcing arbitrary $\omega$-regular properties in arbitrary finite-state Markov chains. The monitors following the strategy restart the chain whenever the current run is suspect of not satisfying the property. The strategies need no information at all about the chain, its probabilities, or its structure. Contrary to the non-probabilistic settings of [17, 3, 16, 8, 9], they work for arbitrary liveness properties. We have given estimates of the number of steps until the last restart and the total number of steps. The design of dedicated strategies that exploit information on these parameters is an interesting topic for future research.

## References

1 Pranav Ashok, Jan Kretínský, and Maximilian Weininger. PAC statistical model checking for Markov decision processes and stochastic games. In *CAV (1)*, volume 11561 of *Lecture Notes in Computer Science*, pages 497–519. Springer, 2019.

2 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

3 David A. Basin, Vincent Jugé, Felix Klaedtke, and Eugen Zalinescu. Enforceable security policies revisited. *ACM Trans. Inf. Syst. Secur.*, 16(1):3:1–3:26, 2013.

4 Hugo Bazille, Blaise Genest, Cyrille Jégourel, and Jun Sun. Global PAC bounds for learning discrete time Markov chains. In *CAV (2)*, volume 12225 of *Lecture Notes in Computer Science*, pages 304–326. Springer, 2020.

5 Yingke Chen, Hua Mao, Manfred Jaeger, Thomas Dyhre Nielsen, Kim Guldstrand Larsen, and Brian Nielsen. Learning Markov models for stationary system behaviors. In *NASA Formal Methods*, volume 7226 of *Lecture Notes in Computer Science*, pages 216–230. Springer, 2012.

6 Przemyslaw Daca, Thomas A. Henzinger, Jan Kretínský, and Tatjana Petrov. Faster statistical model checking for unbounded temporal properties. *ACM Trans. Comput. Log.*, 18(2):12:1–12:25, 2017.

7 Tadashi Dohi, Kishor Trivedi, and Alberto Avritzer, editors. *Handbook of Software Aging and Rejuvenation*. World Scientific, 2020.

8 Yliès Falcone, Laurent Mounier, Jean-Claude Fernandez, and Jean-Luc Richier. Runtime enforcement monitors: composition, synthesis, and enforcement abilities. *Formal Methods Syst. Des.*, 38(3):223–262, 2011.

9 Yliès Falcone and Srinivas Pinisetty. On the runtime enforcement of timed properties. In *RV*, volume 11757 of *Lecture Notes in Computer Science*, pages 48–69. Springer, 2019.

10 Matteo Gagliolo and Jürgen Schmidhuber. Learning restart strategies. In *IJCAI*, pages 792–797, 2007.

11 Kalpana Gondi, Yogeshkumar Patel, and A. Prasad Sistla. Monitoring the full range of omega-regular properties of stochastic systems. In *VMCAI*, volume 5403 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 2009.

12 Yennun Huang, Chandra M. R. Kintala, Nick Kolettis, and N. Dudley Fulton. Software rejuvenation: Analysis, module and applications. In *FTCS*, pages 381–390. IEEE Computer Society, 1995.

13 Thomas Jansen. On the analysis of dynamic restart strategies for evolutionary algorithms. In *PPSN*, volume 2439 of *Lecture Notes in Computer Science*, pages 33–43. Springer, 2002.

14 Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Prism 4.0: Verification of probabilistic real-time systems. In *CAV*, pages 585–591, 2011.

15 Marta Z. Kwiatkowska, Gethin Norman, and David Parker. The PRISM benchmark suite. In *QEST*, pages 203–204. IEEE Computer Society, 2012.

16 Jay Ligatti, Lujo Bauer, and David Walker. Run-time enforcement of nonsafety policies. *ACM Trans. Inf. Syst. Secur.*, 12(3):19:1–19:41, 2009.

17 Fred B. Schneider. Enforceable security policies. *ACM Trans. Inf. Syst. Secur.*, 3(1):30–50, 2000.

18 A. Prasad Sistla and Abhigna R. Srinivas. Monitoring temporal properties of stochastic systems. In *VMCAI*, volume 4905 of *Lecture Notes in Computer Science*, pages 294–308. Springer, 2008.

19 A. Prasad Sistla, Milos Zefran, and Yao Feng. Runtime monitoring of stochastic cyber-physical systems with hybrid state. In *RV*, volume 7186 of *Lecture Notes in Computer Science*, pages 276–293. Springer, 2011.

20 Aad P. A. van Moorsel and Katinka Wolter. Analysis of restart mechanisms in software systems. *IEEE Trans. Software Eng.*, 32(8):547–558, 2006.

21 Jingyi Wang, Jun Sun, Qixia Yuan, and Jun Pang. Should we learn probabilistic models for model checking? A new approach and an empirical study. In *FASE*, volume 10202 of *Lecture Notes in Computer Science*, pages 3–21. Springer, 2017.

22 Katinka Wolter. *Stochastic Models for Fault Tolerance - Restart, Rejuvenation and Checkpointing*. Springer, 2010.

## A Technical Proofs

### A.1 Proof of Proposition 3

**(a)** By standard properties of Markov chains, bad runs of $\mathcal{M} \otimes \mathcal{A}$ almost surely reach a BSCC of $\mathcal{M} \otimes \mathcal{A}$ and then traverse all edges of that BSCC infinitely often. Therefore, a bad run $\rho$ almost surely has a finite prefix $\pi$ that has reached a bad BSCC, say $B$, of $\mathcal{M} \otimes \mathcal{A}$ and has traversed all edges of $B$ at least once. Then $K(\pi) = B$, and so $\pi$ is suspect.

**(b)** Suppose the good runs of $\mathcal{M} \otimes \mathcal{A}$ have nonzero probability. We construct a finite path $\pi$, starting at $s_0$, so that $K(\pi')$ is good for all extensions $\pi'$ of $\pi$, and $K(\pi'')$ is good or undefined for all prefixes $\pi''$ of $\pi$.

Since the good runs of $\mathcal{M} \otimes \mathcal{A}$ have nonzero probability, $\mathcal{M} \otimes \mathcal{A}$ has a good BSCC $B$. Let $\pi'_1$ be a simple (i.e., no repeated states) path from $s_0$ to a state $s_1 \in B \cap (S \times F)$. Extend $\pi'_1$ by a shortest path back to the set $\overline{\pi'_1}$ (forming a lasso) and denote the resulting path by $\pi_1$. Observe that $K(\pi_1) \subseteq B$ is good, and $K(\pi') = \bot$ holds for all proper prefixes $\pi'$ of $\pi_1$. If $K(\pi_1) = B$, then we can choose $\pi := \pi_1$ and $\pi$ has the required properties. Otherwise, let $\pi'_2$ be a shortest path extending $\pi_1$ such that $\pi'_2$ leads to a state in $B \setminus K(\pi_1)$. Extend that path by a shortest path back to $K(\pi_1)$ and denote the resulting path by $\pi_2$. Then we have $K(\pi_1) \subsetneq K(\pi_2) \subseteq B$, and $K(\pi_2)$ is good, and $K(\pi') \in \{K(\pi_1), \bot\}$ holds for all paths $\pi'$ that extend $\pi_1$ and are proper prefixes of $\pi_2$. Repeat this process until a path $\pi$ is found with $K(\pi) = B$. This path has the required properties.

### A.2 Proof of Lemma 11

Let $\mathsf{BSCC}$ denote the set of BSCCs of the chain-automaton product and $\mathsf{SCC}$ the set of its SCCs.

For a subset $K$ of states of the product, $\mathit{Cand}_k(K)$ denotes the event (random predicate) of $K$ being a candidate with strength at least $k$ on a run of the product. Further, the "weak" version $\mathit{WCand}_k(K)$ denotes the event that $K$ has strength $k$ when counting visits even prior to discovery of $K$, i.e. each state of $K$ has been visited and exited at least $k$ times on a prefix $\pi$ of the run with $K(\pi) = K$. Previous work bounds the probability that a non-BSCC can be falsely deemed BSCC based on the high strength it gets.

▶ **Lemma 18** ([6]). *For every set of states $K \notin \mathsf{BSCC}$, and every $s \in K$, $k \in \mathbb{N}$,*

$$\mathbb{P}_s[\mathit{WCand}_k(K)] \leq (1 - \mathsf{p}_{\min})^k .$$

**Proof.** Since $K$ is not a BSCC, there is a state $t \in K$ with a transition to $t' \notin K$. The set of states $K$ becomes a $k$-candidate of a run starting from $s$, only if $t$ is visited at least $k$ times by the path and was never followed by $t'$ (indeed, even if $t$ is the last state in the path, by definition of a $k$-candidate, there are also at least $k$ previous occurrences of $t$ in the path). Further, since the transition from $t$ to $t'$ has probability at least $\mathsf{p}_{\min}$, the probability of not taking the transition $k$ times is at most $(1 - \mathsf{p}_{\min})^k$. ◀

In contrast to [6], we need to focus on runs where $\varphi$ is satisfied. For clarity of notation, we let $K \models \varphi$ denote that $K$ is good, and $K \not\models \varphi$ denote that $K$ is bad. In particular, $K_\infty \models \varphi$ denotes the event that the run satisfies $\varphi$.

▶ **Lemma 19.** *For every set of states $K \notin \mathsf{BSCC}$, and every $s \in K$, $k \in \mathbb{N}$,*

$$\mathbb{P}_s[\mathit{WCand}_k(K) \mid K_\infty \models \varphi] \leq (1 - \mathsf{p}_{\min})^k .$$

**Proof.** The previous argument applies also in the case where we assume that after this strength is reached the run continues in any concrete way (also satisfying $\varphi$) due to the Markovian nature of the product. In the following derivation, $K\langle t \to t' \rangle$ denotes the event that the candidate $K$ is exited through the transition $t \to t'$:

$$\begin{aligned}
&\mathbb{P}_s[\,WCand_k(K) \mid K_\infty \models \varphi] \\
=&\sum_{t\to t'} \mathbb{P}_s[\,WCand_k(K), K\langle t \to t'\rangle \mid K_\infty \models \varphi] \\
=&\sum_{t\to t'} \mathbb{P}_s[\,WCand_k(K), K\langle t \to t'\rangle, K_\infty \models \varphi]/\mathbb{P}_s[K_\infty \models \varphi] \\
=&\sum_{t\to t'} \mathbb{P}_s[\,WCand_k(K), K\langle t \to t'\rangle] \cdot \mathbb{P}_s[K_\infty \models \varphi \mid WCand_k(K), K\langle t \to t'\rangle] \;/\; \mathbb{P}_s[K_\infty \models \varphi] \\
\overset{(1)}{=}&\sum_{t\to t'} \mathbb{P}_s[\,WCand_k(K), K\langle t \to t'\rangle] \cdot \mathbb{P}_s[K_\infty \models \varphi \mid K\langle t \to t'\rangle] \;/\; \mathbb{P}_s[K_\infty \models \varphi] \\
\overset{(2)}{=}&\sum_{t\to t'} \mathbb{P}_s[\,WCand_k(K), K\langle t \to t'\rangle] \cdot \mathbb{P}_{t'}[K_\infty \models \varphi] \;/\; \mathbb{P}_s[K_\infty \models \varphi] \\
\le&\sum_{t\to t' \text{ exiting } K} \frac{\mathbb{P}_s[\text{reach } t] \cdot \mathbb{P}_t[\text{not take } t \to t' \text{in } k \text{ visits of } t] \cdot \mathbf{P}(t,t') \cdot \mathbb{P}_{t'}[K_\infty \models \varphi]}{\mathbb{P}_s[K_\infty \models \varphi]} \\
=&\sum_{t\to t' \text{ exiting } K} \frac{\mathbb{P}_t[\text{not take } t \to t' \text{in } k \text{ visits of } t] \cdot \mathbb{P}_s[\text{reach } t] \cdot \mathbf{P}(t,t') \cdot \mathbb{P}_{t'}[K_\infty \models \varphi]}{\mathbb{P}_s[K_\infty \models \varphi]} \\
\le&\sum_{t\to t' \text{ exiting } K} \frac{(1-\mathsf{p}_{\mathsf{min}})^k \cdot \mathbb{P}_s[\text{reach } t' \text{ as the first state outside } K] \cdot \mathbb{P}_{t'}[K_\infty \models \varphi]}{\mathbb{P}_s[K_\infty \models \varphi]} \\
=&(1-\mathsf{p}_{\mathsf{min}})^k \cdot \mathbb{P}_s[K_\infty \models \varphi] \;/\; \mathbb{P}_s[K_\infty \models \varphi] \\
=&(1-\mathsf{p}_{\mathsf{min}})^k
\end{aligned}$$

where (1) follows by the Markov property and by (almost surely) $K \neq K_\infty$, (2) by the Markov property. ◀

In the next lemma, we lift the results from fixed designated candidates to arbitrary discovered candidates, at the expense of requiring the (strong version of) strength instead of only the weak strength. To that end, let *birthday* $b_i$ be the moment when $i$th candidate on a run is discovered, i.e., a run is split into $\rho = \pi b_i \rho'$ so that $K_i = K(\pi b_i) \neq K(\pi)$. In other terms, $b_i$ is the moment we start counting the occurences for the strength, whereas the weak strength is already 1 there.

▶ **Lemma 20.** *For every $i, k \in \mathbb{N}$, we have*

$$\mathbb{P}[Cand_k(K_i) \mid K_i \notin \mathsf{BSCC}, K_\infty \models \varphi] \le (1-\mathsf{p}_{\mathsf{min}})^k\,.$$

**Proof.**

$$\begin{aligned}
&\mathbb{P}[Cand_k(K_i) \mid K_i \notin \mathsf{BSCC}, K_\infty \models \varphi] \\
&= \frac{\mathbb{P}[Cand_k(K_i), K_i \notin \mathsf{BSCC}, K_\infty \models \varphi]}{\mathbb{P}[K_i \notin \mathsf{BSCC}, K_\infty \models \varphi]} \\
&= \frac{1}{\mathbb{P}[K_i \notin \mathsf{BSCC}, K_\infty \models \varphi]} \sum_{\substack{C \in \mathsf{SCC}\setminus\mathsf{BSCC} \\ s \in C}} \mathbb{P}[Cand_k(C), K_i = C, b_i = s, K_\infty \models \varphi] \\
&= \frac{1}{\mathbb{P}[K_i \notin \mathsf{BSCC}, K_\infty \models \varphi]} \sum_{\substack{C \in \mathsf{SCC}\setminus\mathsf{BSCC} \\ s \in C}} \mathbb{P}[K_i = C, b_i = s] \cdot \mathbb{P}_s[\,WCand_k(C), K_\infty \models \varphi]
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{\mathbb{P}[K_i \notin \mathsf{BSCC}, K_\infty \models \varphi]} \sum_{\substack{C \in \mathsf{SCC}\setminus\mathsf{BSCC}\\ s \in C}} \mathbb{P}[K_i = C, b_i = s] \cdot \mathbb{P}_s[\mathit{WCand}_k(C) \mid K_\infty \models \varphi] \cdot \mathbb{P}_s[K_\infty \models \varphi] \\
&\leq \frac{(1-\mathsf{p}_{\mathsf{min}})^k}{\mathbb{P}[K_i \notin \mathsf{BSCC}, K_\infty \models \varphi]} \sum_{\substack{C \in \mathsf{SCC}\setminus\mathsf{BSCC}\\ s \in C}} \mathbb{P}[K_i = C, b_i = s] \cdot \mathbb{P}_s[K_\infty \models \varphi] && \text{(by Lemma 19)} \\
&\leq \frac{(1-\mathsf{p}_{\mathsf{min}})^k}{\mathbb{P}[K_i \notin \mathsf{BSCC}, K_\infty \models \varphi]} \sum_{\substack{C \in \mathsf{SCC}\setminus\mathsf{BSCC}\\ s \in C}} \mathbb{P}[K_i = C, b_i = s, K_\infty \models \varphi] \\
&= (1-\mathsf{p}_{\mathsf{min}})^k
\end{aligned}
$$

with the last equality due to

$$K_i \notin \mathsf{BSCC} \cap K_\infty \models \varphi = \biguplus_{\substack{C \in \mathsf{SCC}\setminus\mathsf{BSCC}\\ s \in C}} K_i = C, b_i = s, K_\infty \models \varphi$$ ◀

The set $\mathcal{E}rr$ of the next lemma is actually exactly the set considered in Lemma 11 but in a more convenient notation for the computation.

▶ **Lemma 21.** *For $(k_i)_{i=1}^\infty \in \mathbb{N}^\mathbb{N}$, let $\mathcal{E}rr$ be the set of runs such that for some $i \in \mathbb{N}$, we have $\mathit{Cand}_{k_i}(K_i)$ despite $K_i \not\models \varphi$ and $K_\infty \models \varphi$. Then*

$$\mathbb{P}[\mathcal{E}rr] \leq p_\varphi \sum_{i=1}^\infty (1-\mathsf{p}_{\mathsf{min}})^{k_i} .$$

**Proof.**

$$
\begin{aligned}
\mathbb{P}[\mathcal{E}rr] &= \mathbb{P}\left[\bigcup_{i=1}^\infty \Big(\mathit{Cand}_{k_i}(K_i) \cap K_i \not\models \varphi \cap K_\infty \models \varphi\Big)\right] \\
&\leq \mathbb{P}\left[\bigcup_{i=1}^\infty \Big(\mathit{Cand}_{k_i}(K_i) \cap K_i \notin \mathsf{BSCC} \cap K_\infty \models \varphi\Big)\right] \\
&\leq \sum_{i=1}^\infty \mathbb{P}[\mathit{Cand}_{k_i}(K_i) \cap K_i \notin \mathsf{BSCC} \cap K_\infty \models \varphi] && \text{(by the union bound)} \\
&= \sum_{i=1}^\infty \mathbb{P}[\mathit{Cand}_{k_i}(K_i) \mid K_i \notin \mathsf{BSCC} \cap K_\infty \models \varphi] \cdot \mathbb{P}[K_i \notin \mathsf{BSCC} \mid K_\infty \models \varphi] \cdot \mathbb{P}[K_\infty \models \varphi] \\
&\leq \sum_{i=1}^\infty \mathbb{P}[\mathit{Cand}_{k_i}(K_i) \mid K_i \notin \mathsf{BSCC} \cap K_\infty \models \varphi] \cdot 1 \cdot p_\varphi \\
&\leq p_\varphi \sum_{i=1}^\infty (1-\mathsf{p}_{\mathsf{min}})^{k_i} . && \text{(by Lemma 20)}
\end{aligned}
$$

◀

**Proof of Lemma 11.** Lemma 11 claims that

$$\mathbb{P}[\mathcal{E}rr] \leq \varepsilon p_\varphi$$

where $k_i$ is the smallest natural number with $k_i \geq (i - \log \varepsilon) \cdot \frac{-1}{\log(1-\mathsf{p}_{\mathsf{min}})}$. Directly from the previous lemma, by plugging in these $k_i$, we obtain

$$\mathbb{P}[\mathcal{E}rr] \leq p_\varphi \sum_{i=1}^\infty (1-\mathsf{p}_{\mathsf{min}})^{k_i} \leq p_\varphi \sum_{i=1}^\infty 2^{-i} 2^{\log \varepsilon} = p_\varphi \varepsilon .$$ ◀

## A.3 Proof of Theorem 13

Let $I_{ji,k}$ be the number of steps between the $j$-th and $(j+1)$th reset such that the current candidate is $K_i$, and its strength is $k$. Observe that for a Markov chain with $n$ states we have $I_{ji,k} = 0$ if $i > n$ or $j > \alpha(i - \log\varepsilon)$. Indeed, if the Markov chain has $n$ states, then along the run there are at most $n$ candidates; moreover, the strength of the $K_i$ stays strictly below $\alpha(i - \log\varepsilon)$, because otherwise the run is aborted. So we have

$$T = \sum_{j=1}^{\infty} T_j = \sum_{j=1}^{\infty} T_j^{\perp} + \sum_{j=1}^{\infty} T_j^{C} = \sum_{j=1}^{\infty} T_j^{\perp} + \sum_{j=1}^{\infty}\sum_{i=1}^{n}\sum_{k=1}^{\alpha(i-\log\varepsilon)} I_{ji,k} \tag{3}$$

and so, by linearity of expectations,

$$\mathbb{E}(T) = \mathbb{E}\left(\sum_{j=1}^{\infty}\left(T_j^{\perp} + \sum_{i=1}^{n}\sum_{k=1}^{\alpha(i-\log\varepsilon)} I_{ji,k}\right)\right) = \mathbb{E}\left(\sum_{j=1}^{\infty} T_j^{\perp}\right) + \sum_{i=1}^{n}\sum_{k=1}^{\alpha(i-\log\varepsilon)}\sum_{j=1}^{\infty}\mathbb{E}\left(I_{ji,k}\right) \tag{4}$$

Let us bound the first summand. Since $K(\pi) = \perp$ only holds when the last state of $\pi$ is visited for the first time, we have $T_j^{\perp} \leq n$. Moreover, $T_j^{\perp} = 0$ for every $j \geq R$, the number of restarts. So we get

$$\mathbb{E}\left(\sum_{j=1}^{\infty} T_j^{\perp}\right) \leq \mathbb{E}(n \cdot R) = n \cdot \mathbb{E}(R) \tag{5}$$

Consider now the variables $I_{ji,k}$. If $j \geq R$ then $I_{ji,k} = 0$ by definition, since there is no $(j+1)$-th restart. Moreover, under the condition $j < R$ the variables $I_{ji,k}$ and $I_{(j+1)i,k}$ have the same expectation, because they refer to different runs. By Theorem 12(a) $R$ is geometrically distributed with parameter at least $\mathsf{p}_\varphi(1-\varepsilon)$, and so we get

$$\mathbb{E}(I_{(j+1)i,k}) \leq \mathbb{E}(I_{ji,k}) \cdot (1 - \mathsf{p}_\varphi(1-\varepsilon)) \tag{6}$$

Plugging (4) and (5) into (3), and taking into account that $\mathbb{E}(R) \leq 1/\mathsf{p}_\varphi(1-\varepsilon)$, we obtain

$$\begin{aligned}
\mathbb{E}(T) &\leq \mathbb{E}(n \cdot R) + \sum_{i=1}^{n}\sum_{k=1}^{\alpha(i-\log\varepsilon)}\left(\mathbb{E}(I_{0i,k})\sum_{j=0}^{\infty}(1 - \mathsf{p}_\varphi(1-\varepsilon))^j\right) \\
&= n \cdot \mathbb{E}(R) + \sum_{i=1}^{n}\sum_{k=1}^{\alpha(i-\log\varepsilon)} \frac{\mathbb{E}(I_{0i,k})}{\mathsf{p}_\varphi(1-\varepsilon)} \\
&\leq \frac{1}{\mathsf{p}_\varphi(1-\varepsilon)}\left(n + \sum_{i=1}^{n}\sum_{k=1}^{\alpha(i-\log\varepsilon)}\mathbb{E}(I_{0i,k})\right)
\end{aligned} \tag{7}$$

If we can find an upper bound $I \geq \mathbb{E}(I_{0i,k})$ for every $i, k$, then we finally get:

$$\mathbb{E}(T) \leq \frac{1}{\mathsf{p}_\varphi(1-\varepsilon)} \cdot n \cdot (1 + \alpha(n - \log\varepsilon) \cdot I) \leq \frac{1}{\mathsf{p}_\varphi(1-\varepsilon)} \cdot 2n\alpha(n - \log\varepsilon) \cdot I \tag{8}$$

We now compute a bound $I \geq \mathbb{E}(I_{0i,k})$ valid for arbitrary chains. Recall that $\mathbb{E}(I_{0i,k})$ is the number of steps it takes to increase the strength of the $i$-th candidate $K_i$ of the 0-th run from $k$ to $k+1$. This is bounded by the number of steps it takes to visit every state of $K_i$ once. Let $\mathsf{mxsc} \in O(n)$ be the maximal size of a SCC. Given any two states $s, s'$ of an SCC, the probability of reaching $s'$ from $s$ after at most $\mathsf{mxsc}$ steps is at least $\mathsf{p}_{\mathsf{min}}^{\mathsf{mxsc}}$. So the expected time it takes to visit every state of an SCC at least once is bounded by $\mathsf{mxsc} \cdot \mathsf{p}_{\mathsf{min}}^{-\mathsf{mxsc}}$. So taking $I := \mathsf{mxsc} \cdot \mathsf{p}_{\mathsf{min}}^{-\mathsf{mxsc}}$ we obtain the final result.

## A.4 Proof of Proposition 16

When the algorithm explores an edge $s \to s'$ of the Markov chain $\mathcal{M} \otimes \mathcal{A}$, it updates $N$, $D$, $R$, and $S$ as follows. The algorithm first computes $D(s')$, and then proceeds according to the cases (1)-(4):

**(1)** If $s'$ had not been visited before (i.e., $D(s') = N+1$), then the algorithm sets $N := N+1$, inserts $D(s')$ in $R$, and creates a new Fibonacci heap $S(s')$ containing only the state $s'$ with key $(\bot, 0)$.

**(2)** If $s'$ had been visited before (i.e., $D(s') \leq N$), and $D(s) < D(s')$, then the algorithm executes FIND$(s)$ to find the root $r$ of the SCC containing $s$, and updates $S(r)$.

**(3)** If $s'$ had been visited before (i.e., $D(s') \leq N$), and $D(s) = D(s')$, then the algorithm updates the key of $s$ in $S(s)$.

**(4)** If $s'$ had been visited before (i.e., $D(s') \leq N$), and $D(s) > D(s')$, then the algorithm executes the following pseudocode, where $\sigma$ is an auxiliary Fibonacci heap:

```
1: $\sigma \leftarrow \emptyset$
2: repeat
3:     $r \leftarrow$ EXTRACT-MAX$(R)$
4:     $\sigma \leftarrow$ MERGE$(\sigma, S(r))$
5: until $D(r) \leq D(s')$
6: INSERT$(r, R)$
```

At every moment in time the current candidate is the set $S(r)$, where $r =$ EXTRACT-MAX$(R)$, and its strength can be obtained from FIND-MIN$(S(r))$.

Let us now examine the amortized runtime of the implementation. Let $n$ be the total number of updates, and let $n_1, \dots, n_4$ be the number of steps executed by the algorithm corresponding to the cases (1)-(4). In cases (1)-(3), the algorithm executes a constant number of heap operations per step, and so it takes $O((n_1 + n_2 + n_3)\log n)$ amortized time for all steps together. This is no longer so for case (4) steps. For example, if the Markov chain is a big elementary circuit $s_0 \to s_1 \to \cdots \to s_{n-1} \to s_0$, then at each step but the last one we insert one state into the heap, and at the last step we extract them all; that is, the last step takes $O(n)$ heap operations. However, observe that each state is inserted in the heap exactly once, when it is discovered, and extracted at most once. So the algorithm executes at most $n$ EXTRACT-MAX and MERGE heap operations for all case (3) steps together, and the amortized time over all of them is $O(n_3 \log n)$. This gives an overall runtime of $O(n \log n)$, and so an amortized time of $O(\log n)$.

## B Detailed Experimental Results

We briefly interpret the results for every single model.

**gridworld** has a very low satisfaction probability and large BSCCs. Hence Bold monitors (and even Cautious$_{10}$) are easily trapped in a large bad BSCC, so it takes very long to realize that with high confidence (also due to small $\mathsf{p}_{\mathsf{min}}$) and restart. In contrast, Cautious$_1$ restarts very often and thus has many chances to get to the good BSCC. Still, the probability of going to the right BSCC and additionally not looping before it has been explored enough to conclude that it satisfies the property is very low, resulting in many restarts and the time-out.

**nand** also has a low satisfaction probability, but now small BSCCs and no NSCCs (non-bottom SCCs). Hence all monitors manage to refute bad ones and find the good one. Bold monitors spend a bit more time in the bad BSCCs, while Cautious monitors cut earlier. The number of restarts is the same for all (modulo statistical imprecision on 100 runs) since all simply wait for their 1 in 5.7 chance.

**bluetooth** is similar to nand, but with a few NSCCs. Hence Cautious monitors may occasionally give up on the way, very slightly increasing the number of restarts. But again, Bold monitors need a bit more time in BSCCs to become confident, and thus need more steps.

**scale$_{10}$** is designed to have multiple NSCCs on each way, each with 50% self-loop and 50% leaving. Hence Cautious$_1$ can get very easily stuck in one of them, drastically increasing the number of restarts. However, since each NSCC is very probably left within a very few steps, the simple modification to Cautious$_{10}$ fixes the issue, getting essentially the optimum. Bold$_{0.1}$ has no problem with the false restarts, but as it sees many NSCCs, it increases the required strength on the way, resulting in more time to get confidence in the BSCC. Bold$_{0.5}$ has a more relaxed requirements on the strength (starting with less than 10, which is constant for Cautious$_{10}$), making it restart occasionally at the earlier NSCC.

**crowds** has more than ten million states, making the PRISM model checker time out after two hours, not yielding information beyond the size. All the simulations are quite short, indicating the frequent pattern of "fat but shallow" systems. None of the monitors experiences any difficulties, indicating essentially optimal number of restarts (around 1), hence the property seems to be satisfied with roughly 50%. The model checker could not conclude that. While the large size prevents a thorough numeric analysis, it did not prevent our monitors from determining satisfaction on single runs.

**gridworld** has a high satisfaction probability and large BSCCs. There is a single NSCC and the good BSCC has 2499 states, hence many simulations just run into the good BSCC. For the one-in-ten chance $(1 - \mathsf{p}_\varphi)$ of reaching a bad BSCC, Cautious monitors just restart, while Bold monitors need to stay there for an impractically large amount of steps to realize they are stuck and want to restart.

**hermann** satisfies the property almost surely. Bold monitors thus have no problem reaching a good BSCC and exploring the whole of it, which guarantees they will never restart (although the required strength is large, also due to small $\mathsf{p}_{\mathsf{min}}$). In contrast, Cautious monitors keep restarting due to many intermediate candidates, although actually every simulation goes to the good BSCC.