Int J Softw Tools Technol Transfer
DOI 10.1007/s10009-010-0146-x

SPIN 09

Probabilistic reachability for parametric Markov models

Ernst Moritz Hahn · Holger Hermanns · Lijun Zhang

Abstract Given a parametric Markov model, we consider the problem of computing the rational function expressing the probability of reaching a given set of states. To attack this principal problem, Daws has suggested to first convert the Markov chain into a finite automaton, from which a regular expression is computed. Afterwards, this expression is evaluated to a closed form function representing the reachability probability. This paper investigates how this idea can be turned into an effective procedure. It turns out that the bottleneck lies in the growth of the regular expression relative to the number of states ($n^{\Theta(\log n)}$). We therefore proceed differently, by tightly intertwining the regular expression computation with its evaluation. This allows us to arrive at an effective method that avoids this blow up in most practical cases. We give a detailed account of the approach, also extending to parametric models with rewards and with non-determinism. Experimental evidence is provided, illustrating that our implementation provides meaningful insights on non-trivial models.

Keywords Parametric model analysis · Markov chains · Model checking · Reachability

Part of this work was done while L. Zhang was at Saarland University and Oxford University.

E. M. Hahn (✉) · H. Hermanns
Saarland University, Saarbrücken, Germany
e-mail: emh@cs.uni-sb.de

H. Hermanns
INRIA Grenoble, Rhône-Alpes, France

L. Zhang
DTU Informatics, Technical University of Denmark,
Copenhagen, Denmark

Published online: 07 April 2010

1 Introduction

Markov processes have been applied successfully to reason about quantitative properties in a large number of areas such as computer science, engineering, mathematics, biological systems. This paper is about *parametric* Markov processes. In this model class certain aspects are not fixed, but depend on parameters of the model. As an example, consider a communication network with a lossy channel, where whenever a package is sent, it is received with probability x but lost with probability $1 - x$. Such a network can be specified in a probabilistic variation of the PROMELA language [2] or in the language of the PRISM [19] model checker. In this context, we might aim, for instance, at determining the parametric reachability probability, i.e., the probability to reach a given set of target states. This probability is a *rational function* in x.

For (discrete-time) Markov chains (MCs), Daws [10] has devised a language-theoretic approach to solve this problem. In this approach, the transition probabilities are considered as letters of an alphabet. Thus, the model can be viewed as a finite automaton. Then, based on the *state elimination* [20] method, the regular expression describing the language of such an automaton is calculated. In a post-processing step, this regular expression is recursively evaluated, resulting in a rational function over the parameters of the model. Recently, Gruber and Johannsen [14] have shown, however, that the size of the regular expression of a finite automaton explodes: it is $n^{\Theta(\log n)}$ where n is the number of states.

This excessive growth is not only a theoretical insight, but also a very practical problem, as we will discuss. The goal of this paper is to nevertheless present an efficient and effective algorithm for parametric Markov models. Apart from Markov chains, we also consider extensions with rewards or non-determinism.

Our method core is also rooted in the state elimination algorithm. The key difference to the method used by Daws [10] is that instead of post-processing a (possibly prohibitively large) regular expression, we intertwine the state elimination and the computation of the rational function. More precisely, in a state elimination step, we do not use regular expressions to label edges, but label the edges directly with the appropriate rational function representing the flow of probabilities. This also means that we do not work on a finite automaton representation, but instead stay in the domain of MCs all along the process.

We obtain the rational functions in a way inspired by the evaluation put forward by Daws [10]. But since we do this as early as possible, we can exploit symmetries, cancellations and simplifications of arithmetic expressions, especially if most of the transition probabilities of the input model are constants. In practice, this induces drastic savings in the size of the intermediate rational function representations, and hence, is the key to an efficient implementation, as experimental evidence shows.

We apply this idea to parametric Markov *reward* models (PMRMs), in which states and transitions are additionally equipped with reward structures, and these reward structures are possibly parametric. We discuss how to compute the expected accumulated reward with respect to a set of target states **B**. Intuitively, this amounts to the sum of probabilities of paths leading to **B**, each weighted with the reward accumulated along this path. A direct extension of the state elimination approach does not work for PMRMs, as in the final regular expression the probability of individual paths leading to **B** is lost. Surprisingly, our modified approach for parametric MCs can be extended in a straightforward way to handle reward models. Each time we eliminate a state, the transition probabilities are updated as for the underlying parametric MCs. Notably, we update the reward function corresponding to the weighted sum representing the *local* expected accumulated rewards.

To round off the applicability of our method, we present an extension which can tackle parametric Markov decision processes (MDPs), models which involve both probabilistic and non-determinism choices. Here, we are interested in the *maximum* probability of reaching a given set of target states. In order to answer this question, we encode the non-deterministic choices via additional binary parameters, which induce a parametric MC. This modified model is then submitted to the dedicated algorithm for parametric MCs.

In most settings, we reduce the state space prior to state elimination, by extending standard strong [11] and weak bisimulation [3] lumping techniques (computing the quotient model for further analysis) to parametric MCs. A very important observation is that for parametric MDPs we can apply the lumping on the encoded parametric MC, since this preserves the maximum reachability probability. This allows us to minimise parametric MDPs efficiently. We have implemented the algorithms in our tool PARAM [15], including the parametric bisimulation lumping. We illustrate the feasibility of the entire approach on a number of non-trivial parametric MCs, MRMs and MDPs.

Organisation of the paper. In Sect. 2 we introduce the parametric models used in this paper. Then, in Sect. 3, we present our main algorithms for parametric models and discuss bisimulation minimisation for parametric models and the complexity of our algorithms. We provide experimental results in Sect. 4. In Sect. 5, we compare our method to the original approach of Daws. Related work is discussed in Sect. 6. Finally, Sect. 7 concludes this paper.

2 Parametric models

In this section, we present the parametric models which we will use throughout the paper. First, we introduce some general notations. Let S be a finite set. We let $V = \{x_1, \ldots, x_n\}$ denote a set of variables with domain $\mathbb{R}$. An *evaluation* u is a partial function $u : V \rightharpoonup \mathbb{R}$. We let $Dom(u)$ denote the domain of u. We say that u is total if $Dom(u) = V$. A *polynomial* g over V is a sum of monomials

$$g(x_1, \ldots, x_n) = \sum_{i_1,\ldots,i_n} a_{i_1,\ldots,i_n} x_1^{i_1} \ldots x_n^{i_n}$$

where each $i_j \in \mathbb{N}_0$ and each $a_{i_1,\ldots,i_n} \in \mathbb{R}$. A *rational function* f over a set of variables V is a fraction $f(x_1, \ldots, x_n) = \frac{f_1(x_1,\ldots,x_n)}{f_2(x_1,\ldots,x_n)}$ of two polynomials f_1, f_2 over V. Let $\mathcal{F}_V$ denote the set of rational functions from V to $\mathbb{R}$. Given $f \in \mathcal{F}_V$, a set of variables $X \subseteq V$, and an evaluation u, we let $f[X/u]$ denote the rational function obtained by substituting each occurrence of $x \in X \cap Dom(u)$ with $u(x)$.

Definition 1 A parametric Markov chain (PMC) is a tuple $\mathcal{D} = (S, s_0, \mathbf{P}, V)$ where S is a finite set of states, s_0 is the initial state, $V = \{v_1, \ldots, v_n\}$ is a finite set of parameters and $\mathbf{P}$ is the probability matrix $\mathbf{P} : S \times S \to \mathcal{F}_V$.

PMCs have already been introduced in previous publications [10,27]. We now define the underlying graph of a PMC.

Definition 2 The *underlying graph* $\mathcal{G}_\mathcal{D}$ of a PMC $\mathcal{D} = (S, s_0, \mathbf{P}, V)$ is defined as $\mathcal{G}_\mathcal{D} = (S, E_\mathcal{D})$ where $E_\mathcal{D} = \{(s, s') \mid \mathbf{P}(s, s') \neq 0\}$.

We omit the subscript $\mathcal{D}$ if it is clear from the context. Based on the above definition, we introduce some more notations. For $s \in S$, we let $pre(s) = \{s' \mid (s', s) \in E\}$ denote the set of predecessors of s, and let $post(s) = \{s' \mid (s, s') \in E\}$ denote the set of successors of s. We say that s' is reachable from s, denoted by $reach^{\mathcal{G}_\mathcal{D}}(s, s')$, if s' is reachable from s in the underlying graph $\mathcal{G}_\mathcal{D}$. For $A \subseteq S$, we write $reach^{\mathcal{G}_\mathcal{D}}(s, A)$

if $reach^{\mathcal{G}_{\mathcal{D}}}(s, s')$ for any $s' \in A$. We omit the superscript $\mathcal{G}_{\mathcal{D}}$ if it is clear from the context. Now we define the induced PMC with respect to an evaluation:

Definition 3 Let $\mathcal{D} = (S, s_0, \mathbf{P}, V)$ be a PMC. The PMC $\mathcal{D}_u$ induced by an evaluation u is defined as $\mathcal{D}_u = (S, s_0, \mathbf{P}_u, V \backslash Dom(u))$ where the transition matrix $\mathbf{P}_u : S \times S \to \mathcal{F}_{V \backslash Dom(u)}$ is given by $\mathbf{P}_u(s, s') = \mathbf{P}(s, s')[Dom(u)/u]$.

We introduce the notion of *well-defined* evaluations. A *total* evaluation u is *well defined* for $\mathcal{D}$ if $\mathbf{P}_u(s, s') \in [0, 1]$ for all $s, s' \in S$, and $\mathbf{P}_u(s, S) \in [0, 1]$ for all $s \in S$ where $\mathbf{P}_u(s, S)$ denotes the sum $\sum_{s' \in S} \mathbf{P}_u(s, s')$. Intuitively, u is well defined if and only if the resulting PMC $\mathcal{D}_u$ is then an ordinary MC without parameters. For a well-defined evaluation, state s is called *stochastic* if $\mathbf{P}_u(s, S) = 1$, *sub-stochastic* if $\mathbf{P}_u(s, S) < 1$. If $\mathbf{P}_u(s, S) = 0$, s is called *absorbing*.

Let u be a well-defined evaluation, and consider the underlying graphs $\mathcal{G}_{\mathcal{D}} = (S, E_{\mathcal{D}})$ and $\mathcal{G}_{\mathcal{D}_u} = (S, E_{\mathcal{D}_u})$. Obviously, it holds that $E_{\mathcal{D}_u} \subseteq E_{\mathcal{D}}$. We say that the total evaluation function u is *strictly well defined* if the equality holds, i.e., $E_{\mathcal{D}_u} = E_{\mathcal{D}}$. Intuitively, a strictly well-defined evaluation does not destroy the reachability property of the underlying graph. This implies that any edge with function f evaluates to a non-zero probability. As probabilities often correspond to failure probabilities, we often do not need to consider non-strictly well-defined evaluations; we are usually not interested in considering that the probability of a failure is 0 or, on the other hand, that the probability of success is 0.

An infinite path is an infinite sequence $\sigma = s_0 s_1 s_2 \ldots$ of states, and a finite path is a finite sequence $\sigma = s_0 s_1 s_2 \ldots s_n$ of states. A finite path $\sigma = s_0 s_1 s_2 \ldots s_n$ has length $|\sigma| = n$. Let $first(\sigma) = s_0$ denote the first state, and $last(\sigma) = s_n$ denote the last state (for a finite path). A maximal path is either an infinite path or a finite path σ such that $last(\sigma)$ is absorbing. With $\sigma[i]$ we denote the ith state of σ (starting at 0). Let $Path^{\mathcal{D}}$ denote the set of maximal paths of $\mathcal{D}$ and $Path^{\mathcal{D}}_{\text{fin}}$ the set of finite paths. Let $Path^{\mathcal{D}}(s)$ ($Path^{\mathcal{D}}_{\text{fin}}(s)$) be the set of maximal (finite) paths starting in s. For a finite path σ, we define a rational function $Pr^{\mathcal{D}}(\sigma) \in \mathcal{F}_V$ by

$$Pr^{\mathcal{D}}(\sigma) = \prod_{i=0}^{|\sigma|-1} \mathbf{P}(\sigma[i], \sigma[i+1]).$$

For a set of paths $C \subseteq Path^{\mathcal{D}}_{\text{fin}}$ such that there are no $\sigma, \sigma' \in C$ where σ is a prefix of σ', let $Pr^{\mathcal{D}}(C) = \sum_{\sigma \in C} Pr^{\mathcal{D}}(\sigma)$. The function $Pr^{\mathcal{D}}$ can be uniquely extended to the set of paths $Path^{\mathcal{D}}$. For a well-defined evaluation u, $Pr^{\mathcal{D}}(\sigma)[V/u]$ is exactly the probability of the path σ in $\mathcal{D}_u$, and $Pr^{\mathcal{D}_u}$ is the uniquely defined probability measure [31] over the set of paths $Path^{\mathcal{D}}$ given a fixed initial state s_0. We omit the superscript $\mathcal{D}$ if it is clear from the context. Now we consider the extension of PMCs with rewards:

Definition 4 A Parametric Markov Reward Model (PMRM) is a tuple $\mathcal{R} = (\mathcal{D}, r)$ where $\mathcal{D} = (S, s_0, \mathbf{P}, V)$ is a PMC and $r : (S \cup (S \times S)) \to \mathcal{F}_V$ is the reward function.

Intuitively, for states $s, s' \in S$, $r(s)$ is the reward gained by visiting s, and $r(s, s')$ denotes the reward gained if the transition from s to s' is taken. Both $r(s)$ and $r(s, s')$ are rational functions over V. As for PMCs, we define the induced PMRM.

Definition 5 Let $\mathcal{R} = (\mathcal{D}, r)$ be a PMRM. The PMRM $\mathcal{R}_u$ induced by an evaluation u is defined as $\mathcal{R}_u = (\mathcal{D}_u, r_u)$. Here $\mathcal{D}_u$ is defined as in Definition 3. For $s \in S$ and $(s_1, s_2) \in S \times s$ we define $r_u(s) = r(s)[Dom(u)/u]$ and $r_u(s_1, s_2) = r(s_1, s_2)[Dom(u)/u]$.

The evaluation u is well defined and strictly well defined for $\mathcal{R}$ iff it is well defined and strictly well defined for $\mathcal{D}$, respectively. We will also use paths as well as the underlying graph of $\mathcal{R} = (\mathcal{D}, r)$ without referring to $\mathcal{D}$ explicitly. Finally, we consider parametric Markov decision processes which are extensions of PMCs with non-deterministic decisions:

Definition 6 A parametric Markov decision process (PMDP) is a tuple $\mathcal{M} = (S, s_0, Act, \mathbf{P}, V)$ where S, s_0 and V are as for PMCs, and Act is a finite set of *actions*. The transition probability matrix $\mathbf{P}$ is a function $\mathbf{P} : S \times Act \times S \to \mathcal{F}_V$.

As for PMCs, we introduce the PMDP induced by a valuation function:

Definition 7 Given a PMDP $\mathcal{M} = (S, s_0, Act, \mathbf{P}, V)$ and an evaluation u, the PMDP induced by u is defined by $\mathcal{M}_u = (S, s_0, Act, \mathbf{P}_u, V \backslash Dom(u))$ where $\mathbf{P}_u : S \times Act \times S \to \mathcal{F}_{V \backslash Dom(u)}$ is defined by

$$\mathbf{P}_u(s, a, s') = \mathbf{P}(s, a, s')[Dom(u)/u]$$

With $Act(s) = \{a \mid \exists s' \in S.\ \mathbf{P}(s, a, s') \neq 0\}$ we specify the set of *enabled* actions of a state. An infinite path of $\mathcal{M}$ is an infinite sequence $\sigma = s_0 a_0 s_1 a_1 \ldots$, and a finite path is a finite sequence $\sigma = s_0 a_0 s_1 a_1 \ldots s_n$. The notations *maximal path*, $\sigma[i]$, $Path^{\mathcal{M}}$, $Path^{\mathcal{M}}_{\text{fin}}$, $Path^{\mathcal{M}}(s)$ and $Path^{\mathcal{M}}_{\text{fin}}(s)$ are defined in a similar way as for PMCs. The non-deterministic choices are resolved by the notion of *schedulers*. A scheduler is a function

$$A : Path^{\mathcal{M}}_{\text{fin}}(s_0) \to Act$$

satisfying that for $\sigma \in Path^{\mathcal{M}}_{\text{fin}}(s_0)$, $A(\sigma) = a$ implies $a \in Act(last(\sigma))$. We say that A is *stationary* (or called *memoryless* in the literature) if A depends only on the last state, i.e., A is a function $A : S \to Act$. With MD($\mathcal{M}$) we denote the set of stationary schedulers of $\mathcal{M}$. A stationary scheduler induces a PMC as follows:

Definition 8 Given a PMDP $\mathcal{M} = (S, s_0, Act, \mathbf{P}, V)$ and a stationary scheduler A, the PMC *induced* by A is defined as $\mathcal{M}_A = (S, s_0, \mathbf{P}_A, V)$ where the transition matrix $\mathbf{P}_A : S \times S \to \mathcal{F}_V$ is defined by $\mathbf{P}_A(s, s') = \mathbf{P}(s, A(s), s')$.

A total evaluation u is called *strictly well defined* for $\mathcal{M}$ if for each stationary scheduler $A \in \mathrm{MD}(\mathcal{M})$, u is strictly well defined for $\mathcal{M}_A$. For strictly well-defined evaluation u, let $Pr^{\mathcal{M}_u, A}$ denote the probability measure in the PMC $(\mathcal{M}_A)_u$.

2.1 Bisimulation relations

A bisimulation is an equivalence relation on states which subsumes states satisfying the same reachability probability properties. Now we extend the standard strong [11,23] and weak bisimulation [3,4] relations for Markov models to our parametric setting in an obvious way.

Definition 9 Let $\mathcal{D} = (S, s_0, \mathbf{P}, V)$ be a PMC and R be an equivalence relation on S. R is a *strong bisimulation* on $\mathcal{D}$ with respect to $\mathbf{B}$ if for all $s_1 R s_2$ it holds $s_1 \in \mathbf{B}$ iff $s_2 \in \mathbf{B}$, and for all $C \in S/R$ it holds $\mathbf{P}(s_1, C) = \mathbf{P}(s_2, C)$.

States s_1 and s_2 are strongly bisimilar, denoted $s_1 \sim_{\mathcal{D}} s_2$ iff there exists a strong bisimulation R on $\mathcal{D}$. Note that we have operations on functions in the definition, instead of numbers. Strong bisimulation can be extended for PMRMs without transition rewards by additionally requiring that $r(s_1) = r(s_2)$ for all $C \in S/R$ if $s_1 R s_2$. Now we give the notion of weak bisimulation:

Definition 10 Let $\mathcal{D} = (S, s_0, \mathbf{P}, V)$ be a PMC and R be an equivalence relation on S. Let $\mathbf{B}$ be a set of target states. R is a *weak bisimulation* on $\mathcal{D}$ with respect to $\mathbf{B}$ if for all $s_1 R s_2$ $s_1 \in \mathbf{B}$ iff $s_2 \in \mathbf{B}$, and

1. If $\mathbf{P}(s_i, [s_i]_R) \neq 1$ for $i = 1, 2$ then for all $C \in S/R$, $C \neq [s_1]_R = [s_2]_R$:
$$\frac{\mathbf{P}(s_1, C)}{1 - \mathbf{P}(s_1, [s_1]_R)} = \frac{\mathbf{P}(s_2, C)}{1 - \mathbf{P}(s_2, [s_2]_R)}.$$
2. s_1 can reach a state outside $[s_1]_R$ iff s_2 can reach a state outside $[s_2]_R$.

We say that states s_1 and s_2 are weakly bisimilar, denoted $s_1 \approx_{\mathcal{D}} s_2$ iff there exists a weak bisimulation R on $\mathcal{D}$. Weak bisimulation is strictly coarser than strong bisimulation. To the best of our knowledge, weak bisimulation has not been introduced for Markov reward models. The largest (strong or weak) bisimulation equivalence allows us to obtain the *quotient*, which is the smallest model bisimilar to the original model. As the reachability properties are preserved under bisimulations, we can work with the quotient instead of the original model.

3 Algorithms

In this section we first present an algorithm for the reachability probability for PMCs. Then, we extend our algorithm to PMRMs and PMDPs in Sects. 3.2 and 3.3, respectively. In Sect. 3.4, we discuss how to minimise parametric models using bisimulation, and we discuss the complexity of our algorithm in Sect. 3.5.

3.1 Parametric MCs

Let $\mathcal{D}$ be a PMC and let $\mathbf{B}$ be a set of target states. We are interested in the *parametric reachability probability*, i.e., the function representing the probability to reach a set of target states $\mathbf{B}$ from s_0, for all well-defined evaluations. This is defined by

$$Pr^{\mathcal{D}}(\{\sigma \in Path^{\mathcal{D}} \mid \sigma[0] = s_0 \wedge \exists i.\sigma[i] \in \mathbf{B}\}).$$

Daws [10] has already solved this problem as follows: first, the PMC is transformed into a finite automaton, with the same initial state, and $\mathbf{B}$ as the final states. Transition probabilities are described by symbols from an alphabet of the automaton of the form $\frac{p}{q}$ or x representing rational numbers, or variables. Afterwards, based on the *state elimination* [20] method, the regular expression describing the language of such an automaton is calculated. Then, these regular expressions are evaluated into rational functions representing the probability to finally reach the target states. However, this approach can become very costly, as the length of a regular expression obtained from an automaton is $n^{\Theta(\log n)}$ [14].

In this section, we present an improved algorithm in which state elimination and the computation of rational functions are intertwined. As we do not compute the regular expressions as an intermediate step anymore, this allows for a more efficient implementation. The reason is that the rational functions can be simplified during the state elimination steps, thus avoiding the blowup of regular expressions.

The algorithm is presented in Algorithm 1. The input is a PMC $\mathcal{D}$ and a set of target states $\mathbf{B}$. Since we are interested in the reachability probability, w.l.o.g., we can make the target states absorbing, and remove states (and corresponding edges) not reachable from s_0, or which cannot reach $\mathbf{B}$ a priori. We note that removing states could induce sub-stochastic states. A usual search algorithm is sufficient for this preparation. In the algorithm $+, -$, etc. are operations for rational functions, and *exact arithmetic* is used to avoid numerical problems. The key idea of the algorithm is to eliminate states from the PMC one by one, while maintaining the reachability probability. The elimination of a single state $s \notin \{s_0\} \cup \mathbf{B}$ is illustrated in Fig. 1. The labels represent the corresponding transition probabilities. The function *eliminate*(s) eliminates state s from $\mathcal{D}$. When eliminating s, we consider all pairs $(s_1, s_2) \in pre(s) \times post(s)$. After eliminating s, the new

Algorithm 1 Parametric Reachability Probability for PMCs

```
Require: PMC $\mathcal{D} = (S, s_0, \mathbf{P}, V)$ and the set of target states $\mathbf{B}$. State $s \in \mathbf{B}$ is absorbing. For all $s \in S$, it holds $reach(s_0, s)$ and $reach(s, \mathbf{B})$.
for all $s \in S \setminus (\{s_0\} \cup \mathbf{B})$ do
    for all $(s_1, s_2) \in pre(s) \times post(s)$ do
        $\mathbf{P}(s_1, s_2) = \mathbf{P}(s_1, s_2) + \mathbf{P}(s_1, s) \frac{1}{1-\mathbf{P}(s,s)} \mathbf{P}(s, s_2)$
    eliminate(s)
return $\frac{1}{1-\mathbf{P}(s_0,s_0)} \mathbf{P}(s_0, \mathbf{B})$
```

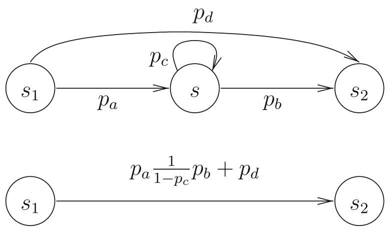

Fig. 1 State elimination for PMCs

transition probability from s_1 to s_2 becomes $f(s_1, s_2) := p_d + \frac{p_a p_b}{1-p_c}$. The second term $\frac{p_a p_b}{1-p_c}$ of $f(s_1, s_2)$ is the geometric sum $\sum_{i=0}^{\infty} p_a p_c^i p_b = \frac{p_a p_b}{1-p_c}$, which corresponds to the probability of reaching s_2 from s_1 through s.

We now discuss the correctness of our algorithm. Consider the simple PMC in Fig. 1. Assume that we have $V = \{p_a, p_b, p_c, p_d\}$. For strictly well-defined evaluation, our computed rational function $f(s_1, s_2)$ is correct, which can be seen as follows. If u is strictly well defined, we have that $E_{\mathcal{D}} = E_{\mathcal{D}_u}$, implying that $u(p_c) > 0$, $u(p_b) > 0$ and $u(p_c) + u(p_b) \leq 1$. This indicates also that the denominator $1 - u(p_c)$ is not zero. Obviously, for a well-defined evaluation u with $u(p_c) = 1$, our result $f(s_1, s_2)$ is not defined at all. The problem is that state s cannot reach s_2 in $\mathcal{G}_{\mathcal{D}_u}$ any more. Now consider another well-defined (but not strictly well-defined) evaluation u satisfying $u(p_c) = 0$ and $u(p_b) = 1$. It is easy to check that $f(s_1, s_2)$ returns the right result in this case. We introduce the notion of *maximal well-defined* evaluations for this purpose:

Definition 11 Assume that the PMC $\mathcal{D}$ and the set of states $\mathbf{B}$ satisfy the requirement of Algorithm 1. The total evaluation u is maximal well defined, if it is well defined, and if for each $s \in S$ it holds that $reach^{\mathcal{D}_u}(s, \mathbf{B})$.

This means that under maximal well-defined evaluations we can still reach the set of target states from all states of the model after inserting values into the parametric model according to the evaluation. This does not mean that the reachability probability is 1, because we allow sub-stochastic models. Now we give the correctness of Algorithm 1.

Lemma 1 *Assume that the PMC $\mathcal{D}$ and the set of states $\mathbf{B}$ satisfy the requirement of Algorithm 1. Assume that the algorithm returns $f \in \mathcal{F}_V$. Then, for maximal well-defined evaluation u it holds that $Pr^{\mathcal{D}_u}(s_0, \mathbf{B}) = f[V/u]$.*

The detailed induction-based proof can be found in Appendix A.1. We can handle non-maximal evaluations by reducing them to maximal evaluations. The details are skipped here.

3.2 Parametric MRMs

Let $\mathcal{R} = (\mathcal{D}, r)$ be a PMRM with $\mathcal{D} = (S, s_0, \mathbf{P}, V)$. Let $\mathbf{B} \subseteq S$ be a set of target states. We are interested in the *parametric expected accumulated reward* [26] until $\mathbf{B}$ is reach. We denote this value by $R^{\mathcal{R}}(s_0, \mathbf{B})$. Formally, $R^{\mathcal{R}}(s_0, \mathbf{B})$ is the *expectation* of the random variable

$$X^{\mathcal{R}} : \sigma \in Path^{\mathcal{D}}(s_0) \to \mathbb{R}_{\geq 0}$$

which is defined by $X^{\mathcal{R}}(\sigma)$ equals 0 if $first(\sigma) \in \mathbf{B}$, ∞ if $\sigma[i] \notin \mathbf{B}$ for all i, and otherwise, equals

$$\sum_{i=0}^{\min\{j \mid \sigma[j] \in \mathbf{B}\}-1} r(\sigma[i]) + r(\sigma[i], \sigma[i+1]).$$

In Algorithm 2, we extend the algorithm for PMCs to handle PMRMs. The input model is a PMRM $\mathcal{R} = (\mathcal{D}, r)$ where we have the same requirement of $\mathcal{D}$ as Algorithm 1 plus the assumption that the set of target states is reached with probability 1 (can be checked by Algorithm 1) for the evaluations under consideration. We discuss briefly how other special cases can be dealt with by means of simple search algorithms. As for PMCs, states not reachable from s_0 need not be considered. Assume that there exists a state s satisfying the property that $reach(s_0, s)$ and that $\neg reach(s, \mathbf{B})$. By definition, any path σ containing s would have infinite reward, which implies also that $R^{\mathcal{R}}(s_0, \mathbf{B}) = \infty$. Assume that $\mathcal{D}$ satisfies the requirement of the algorithm. In this case we have

$$R^{\mathcal{R}}(s_0, \mathbf{B}) = \sum_{\sigma} Pr^{\mathcal{D}}(\sigma) \cdot X^{\mathcal{R}}(\sigma)$$

where σ ranges over all maximal paths of $\mathcal{D}$. The key part of the algorithm is the adaption of the state elimination algorithm for $\mathcal{R}$. Consider the pair $(s_1, s_2) \in pre(s) \times post(s)$. The core is how to obtain the transition reward for the new transition (s_1, s_2) after eliminating s. Consider Fig. 2, where the label p/r of the edge (s, s') denotes the transition probability and the transition reward of the transition, respectively. We construct the transition from s_1 to s_2 in two steps. In the first

Algorithm 2 Parametric Expected Reward for PMRM

```
Require: PMRM $\mathcal{R} = (\mathcal{D}, r)$ with $\mathcal{D} = (S, s_0, \mathbf{P}, V)$, the set of target states $\mathbf{B}$. State $s \in \mathbf{B}$ is absorbing. For all $s \in S$, it holds that $reach(s_0, s)$
    and for all maximal well-defined evaluations $u$ it is $Pr^{\mathcal{D}_u}(s, \mathbf{B}) = 1$
for all $s \in S \backslash (\{s_0\} \cup \mathbf{B})$ do
    for all $(s_1, s_2) \in pre(s) \times post(s)$ do
        $p_e = \mathbf{P}(s_1, s) \frac{1}{1-\mathbf{P}(s,s)} \mathbf{P}(s, s_2)$
        $r_e = r(s_1, s) + r(s, s_2) + r(s) + \frac{\mathbf{P}(s,s)}{1-\mathbf{P}(s,s)} (r(s, s) + r(s))$
        $r(s_1, s_2) = \frac{p_e r_e + \mathbf{P}(s_1,s_2) r(s_1,s_2)}{p_e + \mathbf{P}(s_1,s_2)}$
        $\mathbf{P}(s_1, s_2) = \mathbf{P}(s_1, s_2) + p_e$
    eliminate(s)
return $\sum_{s \in \mathbf{B}} \left\{ \frac{\mathbf{P}(s_0,s)}{1-\mathbf{P}(s_0,s_0)} \cdot (r(s_0) + r(s_0, s)) + \frac{\mathbf{P}(s_0,s_0)\mathbf{P}(s_0,s)}{(1-\mathbf{P}(s_0,s_0))^2} \cdot (r(s_0, s_0) + r(s_0)) \right\}$
```

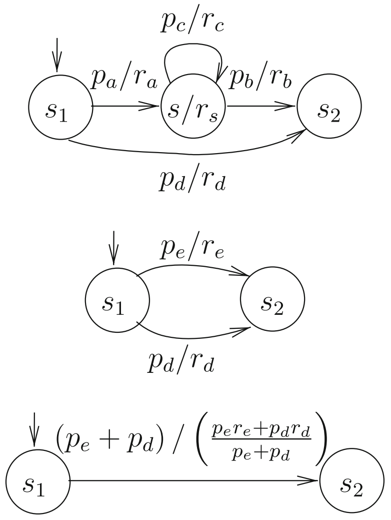

Fig. 2 For parametric MRMs in which we have $p_e = \frac{p_a p_b}{1-p_c}$, and $r_e = r_a + r_b + r_s + \frac{p_c}{1-p_c}(r_c + r_s)$

step we assume that $\mathbf{P}(s_1, s_2) = 0$ ($p_d = 0$ in the figure). As for PMCs, after removing s, the probability of moving from s_1 to s_2 is the infinite sum $f(s_1, s_2) := \sum_{i=0}^{\infty} p_a p_c^i p_b = \frac{p_a p_b}{1-p_c}$. Strictly according to our definition, the expected accumulated rewards would be

$$g(s_1, s_2) := \sum_{i=0}^{\infty} (p_a p_c^i p_b) \cdot (r_a + r_s + (r_c + r_s)i + r_b)$$

$$= (r_a + r_s + r_b) \frac{p_a p_b}{1 - p_c} + p_a p_b (r_c + r_s) \sum_{i=0}^{\infty} i p_c^i$$

The sum $\sum_{i=0}^{\infty} i p_c^i$ can be simplified to $\frac{p_c}{(1-p_c)^2}$. Then, we would take the function $r_e := \frac{g(s_1,s_2)}{f(s_1,s_2)}$ for the new reward from (s_1, s_2). It can be simplified to

$$r_e = r_a + r_b + r_s + \frac{p_c}{1 - p_c}(r_c + r_s).$$

This reward can be understood as follows: the sum $r_a + r_b + r_s$ corresponds to the rewards via visiting s and taking transitions (s_1, s) and (s, s_2). The term $\frac{p_c}{1-p_c}$ can be interpreted as the expected number of times that the self-loop of s is taken; thus the second part is obtained by multiplying it with the rewards $r_c + r_s$ of a single loop.

Now we take account of the case $\mathbf{P}(s_1, s_2) > 0$. The probability becomes then $p_e + p_d$ where $p_e = \frac{p_a p_b}{1-p_c}$ and $p_d = \mathbf{P}(s_1, s_2)$. A similar analysis as above allows us to get the reward $\frac{p_e r_e + p_d r_d}{p_e + p_d}$. Now we give the correctness of the algorithm for the expected reward, the proof of which can be found in Appendix A.2.

Lemma 2 *Assume that the PMRM $\mathcal{R} = (\mathcal{D}, r)$ and $\mathbf{B}$ satisfy the requirement of Algorithm 2. Assume that the algorithm returns $f \in \mathcal{F}_V$. Let u be a maximal well-defined evaluation. Then, it holds that $R^{\mathcal{R}_u}(s_0, \mathbf{B}) = f[V/u]$.*

3.3 Parametric MDPs

Let $\mathcal{M} = (S, s_0, Act, \mathbf{P}, V)$ be a PMDP and let $\mathbf{B} \subseteq S$ be a set of target states. Our goal in this section is to compute the maximum reachability probability of $\mathbf{B}$ in $\mathcal{M}$ with respect to all schedulers. Formally, we want to compute the maximum $\max_A Pr^{\mathcal{M}_u, A}(s_0, \mathbf{B})$ for each strictly well-defined evaluation u, with A ranging over all schedulers. For the ordinary MDP case (e.g. $\mathcal{M}_u$ where u is strictly well defined), it has been shown [5] that the class of stationary schedulers is sufficient to achieve this maximum reachability probability. For PMDPs, different stationary schedulers are needed for different evaluations:

Example 1 Consider the PMDP

$$\mathcal{M} = (\{s_0, s_1, s_2\}, s_0, \{a, b\}, \mathbf{P}, \{x\})$$

where $\mathbf{P}$ is defined by: $\mathbf{P}(s_0, a, s_1) = \mathbf{P}(s_0, a, s_2) = \frac{1}{2}$, $\mathbf{P}(s_0, b, s_1) = x$, $\mathbf{P}(s_0, b, s_2) = 1 - x$. Let $\mathbf{B} = \{s_1\}$. Obviously, for $x \leq \frac{1}{2}$ taking decision a we get the maximum reachability probability $\frac{1}{2}$. Moreover, for $x \geq \frac{1}{2}$ we get the maximum reachability probability x with decision b.

We introduce binary variables to encode non-deterministic choices in PMDPs, as anticipated by Daws [10]. For state $s \in S$ with a number of $k = |Act(s)|$ non-deterministic choices, we need to introduce $k - 1$ variables.

Definition 12 Let $s \in S$ with $|Act(s)| > 1$. Let $\delta(s) \in Act(s)$ be an arbitrary selected action. Then, for each $a \in Act(s)$ and $a \neq \delta(s)$, we introduce a binary variable $v_{s,a}$, denoted by $\mathrm{enc}(s, a)$, to encode the transition from s when choosing a. The transition with respect to $\delta(s)$ is encoded via

$$\mathrm{enc}(s, \delta(s)) := 1 - \sum_{b \in Act(s), b \neq \delta(s)} v_{s,b}.$$

In the following, we fix δ as defined above and let Var_δ denote the set of these variables, all of which have domain $\{0, 1\}$. Intuitively, $v_{s,a} = 1$ indicates that the transition labelled with a is taken from s, whereas $v_{s,a} = 0$ for all $v_{s,a}$ with $a \neq \delta$ indicates that $\delta(s)$ is taken. Now we define the encoding of $\mathcal{M}$ with respect to Var_δ.

Definition 13 Let $\mathcal{M} = (S, s_0, Act, \mathbf{P}, V)$ be a PMDP. The *encoding PMC* with respect to Var_δ is defined as $\mathrm{enc}(\mathcal{M}) = \big(S, s_0, \mathbf{P}_\delta, V \dot{\cup} Var_\delta\big)$ where

$$\mathbf{P}_\delta(s, s') = \sum_{a \in Act} \mathbf{P}(s, a, s') \cdot \mathrm{enc}(s, a).$$

To avoid confusion, we use $v : Var_\delta \to \{0, 1\}$ to denote a total evaluation function for Var_δ. We say v is *stationary*, if for each s with $|Act(s)| > 1$, there exists at most one $a \in Act(s) \backslash \{\delta(s)\}$ with $v(v_{s,a}) = 1$. We let SE_X denote the set of stationary evaluations v with domain $Dom(v) = X$, and let $SE := SE_{Var_\delta}$. Observe that if $v(v_{s,a}) = 0$ for all $a \in Act(s) \backslash \{\delta(s)\}$, the transition labelled with $\delta(s)$ is selected.

We can apply Algorithm 1 on the encoding PMC to compute the parametric reachability probability. In the following, we discuss how to transform the result achieved this way back to the maximum reachability probability for the original PMDPs. The following lemma states that each stationary scheduler corresponds to a stationary evaluation with respect to Var_δ:

Lemma 3 *Let $\mathcal{M} = (S, s_0, Act, \mathbf{P}, V)$ be a PMDP. Then for each stationary scheduler A there is a stationary evaluation $v \in SE$ such that $\mathcal{M}_A = (\mathrm{enc}(\mathcal{M}))_v$. Moreover, for each stationary evaluation $v \in SE$ there exists a stationary scheduler A such that $(\mathrm{enc}(\mathcal{M}))_v = \mathcal{M}_A$.*

Proof Let $\mathcal{M} = (S, s_0, Act, \mathbf{P}, V)$ be a PMDP, and let $A : S \to Act$ be a stationary scheduler. We define a stationary evaluation v with

$$v(v_{s,a}) = \begin{cases} 0, & A(s) \neq a \\ 1, & A(s) = a \end{cases}$$

Then we have $\mathcal{M}_A = (\mathrm{enc}(\mathcal{M}))_v$. If, on the other hand, we start with a stationary evaluation v, we can define a stationary scheduler A by $A(s) = a$ iff

- either $v(v_{s,a}) = 1$, or
- $\delta(s) = a$ and for all $v_{s,b}$ it is $v(v_{s,b}) = 0$.

Then again we have $\mathcal{M}_A = (\mathrm{enc}(\mathcal{M}))_v$. □

Because stationary schedulers are sufficient for maximum reachability probabilities, the above lemma suggests that for a strictly well-defined evaluation u of $\mathcal{M}$, it holds that

$$\max_{A \in \mathrm{MD}(\mathcal{M})} Pr^{\mathcal{M}_u, A}(s_0, \mathbf{B}) = \max_{v \in SE} Pr^{(\mathrm{enc}(\mathcal{M}_u))_v}(s_0, \mathbf{B}).$$

Together with Lemma 1, the following lemma discusses the computation of this maximum:

Lemma 4 *Let $\mathcal{M} = (S, s_0, Act, \mathbf{P}, V)$ be a PMDP and let f be the function obtained by applying Algorithm 1 on $\mathrm{enc}(\mathcal{M})$. Let Var_f denote the set of variables occurring in f. Then for each strictly well-defined evaluation u of $\mathcal{M}$, it holds that*

$$\max_{A \in \mathrm{MD}(\mathcal{M})} Pr^{\mathcal{M}_u, A}(s_0, \mathbf{B}) = \max_{v \in SE_{Var_\delta \cap Var_f}} f[Var_\delta / v][V/u].$$

Proof Let $\mathcal{M} = (S, s_0, Act, \mathbf{P}, V)$ be a PMDP, let u be a strictly well-defined evaluation for $\mathcal{M}$ and let A be a stationary scheduler with

$$Pr^{\mathcal{M}_u, A}(s_0, \mathbf{B}) = \max_{A' \in \mathrm{MD}(\mathcal{M})} Pr^{\mathcal{M}_u, A'}(s_0, \mathbf{B}) \tag{1}$$

Without loss of generality, we can assume that the chosen A in the above equation satisfies the following constraint:

$$Pr^{\mathcal{M}_u, A}(s, \mathbf{B}) = \max_{A' \in \mathrm{MD}(\mathcal{M})} Pr^{\mathcal{M}_u, A'}(s, \mathbf{B}) \tag{2}$$

for all $s \in S$ instead of just the initial state [5]. Let f be the function returned from applying Algorithm 1 on $\mathrm{enc}(\mathcal{M})$. Let $v : Var_\delta \to \{0, 1\}$ be the evaluation from Lemma 3. Then v is the evaluation needed. Applying first u and then v is equivalent to applying $w : V \dot{\cup} Var_\delta \to \mathbb{R}$ with

$$w(a) = \begin{cases} u(a) & \text{if } a \in V \\ v(a) & \text{if } a \in Var_\delta \end{cases}$$

as V and Var_δ are disjunctive sets. So, $f[V \dot{\cup} Var_\delta / w] = f[Var_\delta / v][V/u]$. We show that w is maximal well defined in $\mathrm{enc}(\mathcal{M})$ that is

1. w is well defined, and
2. $reach^{(\mathrm{enc}(\mathcal{M}))_w}(s, \mathbf{B})$ for all $s \in S$.

For (1), this is clear. For (2), let

$$S' = \Big\{ s \in S \mid reach^{\mathrm{enc}(\mathcal{M})}(s, \mathbf{B}) \Big\}.$$

We only have to show that $reach^{(\mathrm{enc}(\mathcal{M}))_w}(s, \mathbf{B})$ for state $s \in S'$, because states not in S' will be removed by the preprocessing of the state-elimination algorithm. As u is strictly well defined, $reach^{(\mathrm{enc}(\mathcal{M}))_u}(s, \mathbf{B})$ for $s \in S'$. Thus, there is a v' with $Pr^{((\mathrm{enc}(\mathcal{M}))_u)_{v'}}(s, \mathbf{B}) > 0$. This means there is a scheduler A' with $Pr^{(\mathcal{M}_{A'})_u}(s, \mathbf{B}) > 0$. Because of Eq. 2, it follows that $Pr^{(\mathcal{M}_A)_u}(s, \mathbf{B}) > 0$. Due to the definition of v, this also means $Pr^{((\text{enc}(\mathcal{M}))_u)_v}(s, \mathbf{B}) > 0$ and in turn $Pr^{(\text{enc}(\mathcal{M}))_w}(s, \mathbf{B}) > 0$, which is equivalent to $reach^{(\text{enc}(\mathcal{M}))_w}(s, \mathbf{B})$.

Now we have

$$\begin{aligned} f[Var_\delta/v][V/u] &= f[V \dot{\cup} Var_\delta/w] \overset{\text{Lem.1}}{=} Pr^{\text{enc}(\mathcal{M})_w}(s_0, \mathbf{B}) \\ &= Pr^{((\text{enc}(\mathcal{M}))_v)_u}(s_0, \mathbf{B}) \overset{\text{Lem.3}}{=} Pr^{M_u, A}(s_0, \mathbf{B}) \\ &\overset{Eq.\ (1)}{=} \max_{A' \in \text{MD}(\mathcal{M})} Pr^{\mathcal{M}_u, A'}(s_0, \mathbf{B}). \end{aligned}$$

□

In worst case, we have $SE_{Var_\delta \cap Var_f} = SE$. The size $|SE| = \prod_{s \in S} |Act(s)|$ grows exponential in the number of states s with $|Act(s) > 1|$.

3.4 Bisimulation minimisation for parametric models

We discuss how to apply bisimulation strategy to reduce the state space before our main algorithm. For PMCs, both strong and weak bisimulation can be applied, while for PMRMs only strong bisimulation is used. The most interesting part is for PMDPs, for which we minimise the encoded PMC instead of the original one. The following lemma shows that strong (weak) bisimilar states in $\mathcal{D}$ are also strong (weak) bisimilar in $\mathcal{D}_u$ for each maximal well-defined evaluation:

Lemma 5 *Let $\mathcal{D} = (S, s_0, \mathbf{P}, V)$ be a PMC with $s_1, s_2 \in S$. Let $\mathbf{B}$ be a set of target states. Then, for all maximal well-defined evaluation u, $s_1 \sim_{\mathcal{D}} s_2$ implies that $s_1 \sim_{\mathcal{D}_u} s_2$, and $s_1 \approx_{\mathcal{D}} s_2$ implies that $s_1 \approx_{\mathcal{D}_u} s_2$.*

Proof First we prove $s_1 \sim_{\mathcal{D}} s_2 \Rightarrow s_1 \sim_{\mathcal{D}_u} s_2$ for all well-defined evaluations u:

If $s_1 \sim s_2$, then there exists a strong bisimulation R with $s_1 R s_2$. Obviously R is also a bisimulation in $\mathcal{D}_u$: for $s_1' R s_2'$ and $C \in S/R$, we have $\mathbf{P}(s_1', C) = \mathbf{P}(s_2', C)$ implies $\mathbf{P}_u(s_1', C) = \mathbf{P}_u(s_2', C)$. Whether states are contained in $\mathbf{B}$ is not changed by the model.

Now we prove $s_1 \approx_{\mathcal{D}} s_2 \Rightarrow s_1 \approx_{\mathcal{D}_u} s_2$ for all maximal well-defined evaluations u:

If $s_1 \approx s_2$, then there exists a weak bisimulation R with $s_1 R s_2$. Moreover, for $s_1' R s_2'$ it holds

1. $s_1' \in \mathbf{B}$ iff $s_2' \in \mathbf{B}$
2. if $\mathbf{P}(s_i', [s_i']_R) \neq 1$ for $i = 1, 2$ then it is $\frac{\mathbf{P}(s_1', C)}{1 - \mathbf{P}(s_1', [s_1']_R)} = \frac{\mathbf{P}(s_2', C)}{1 - \mathbf{P}(s_2', [s_2']_R)}$ for all $C \in S/R$ if $s_1' R s_2'$, and
3. s_1' can reach a state outside $[s_1']$ iff s_2' can also in $\mathcal{G}_{\mathcal{D}}$.

We show that R is also a weak bisimulation in $\mathcal{D}_u$. For (1), this is clear. For (2), assume that $\mathbf{P}_u(s_i', [s_i']_R) < 1$ for $i = 1, 2$. In this case, we must have $\mathbf{P}(s_i', [s_i']_R) \neq 1$ and thus $\frac{\mathbf{P}(s_1', C)}{1 - \mathbf{P}(s_1', [s_1']_R)} = \frac{\mathbf{P}(s_2', C)}{1 - \mathbf{P}(s_2', [s_2']_R)}$ which implies $\frac{\mathbf{P}_u(s_1', C)}{1 - \mathbf{P}_u(s_1', [s_1']_R)} = \frac{\mathbf{P}_u(s_2', C)}{1 - \mathbf{P}_u(s_2', [s_2']_R)}$. For (3) we notice that in maximal well-defined evaluations we always can reach $\mathbf{B}$ from states not in $\mathbf{B}$ in $\mathcal{G}_{\mathcal{D}_u}$.

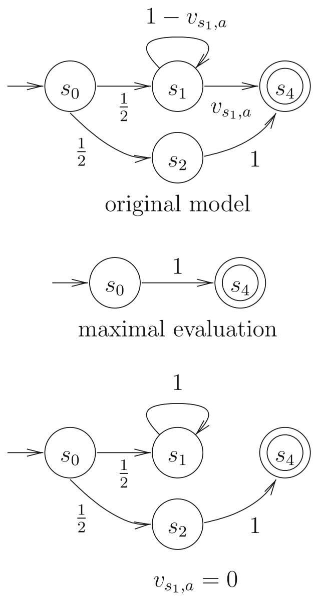

Fig. 3 Weak bisimulation does not maintain minimal reachability in PMDPs

In Fig. 3, it is illustrated why weak bisimulation is only valid for maximal evaluations. The computed partitioning with respect to weak bisimulation is $S/R = \{\{s_0, s_1, s_2\}, \{s_4\}\}$. This is correct with respect to any maximal well-defined evaluation u: since $u(v_{s_1,a}) > 0$ implying both s_1 and s_2 can reach $\mathbf{B}$. The quotient automaton is depicted in the middle of the figure.

Now consider the evaluation function u' with $u'(v_{s_1,a}) = 0$. Obviously u' is not maximal well defined, as state s_1 cannot reach $\mathbf{B}$. For this evaluation no states are weak bisimilar; thus the quotient is the same as the original automaton (depicted on the lower part). □

Both strong and weak bisimulation preserve the reachability probability for ordinary MCs [3,17]. By the above lemma, for PMCs, both strong and weak bisimulation preserve reachability probability for all maximal well-defined evaluations. A similar result holds for PMRMs: if two states s_1, s_2 of $\mathcal{R} = (\mathcal{D}, r)$ are strong bisimilar, i.e. $s_1 \sim_{\mathcal{R}} s_2$, then for all maximal well-defined evaluations u, we have $s_1 \sim_{\mathcal{R}_u} s_2$. As a consequence, strong bisimulation preserves expected accumulated rewards for all well-defined evaluations for PMRMs.

Now we discuss how to minimise PMDPs. Instead of computing the bisimulation quotient of the original PMDPs $\mathcal{M}$, we apply the bisimulation minimisation algorithms on the encoded PMCs $\text{enc}(\mathcal{M})$. Since both strong and weak bisim-

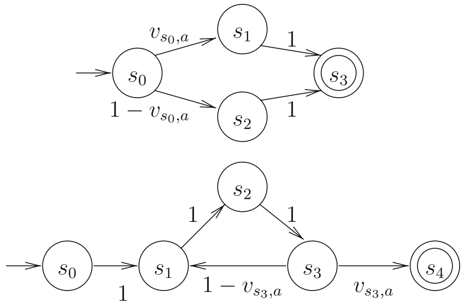

Fig. 4 Bisimulation for PMDPs

ulation preserve reachability for PMCs, by Lemmas 3 and 4, bisimulation minimisation on the encoded PMC enc($\mathcal{M}$) also preserves the maximum reachability probability on $\mathcal{M}$ with respect to strictly well-defined evaluations. Thus, we can apply the efficient strong and weak bisimulation algorithm for the encoding PMC directly. The following example illustrates the use of strong and weak simulations for PMDPs:

Example 2 Consider the encoding PMC on the upper part of Fig. 4. States s_1, s_2 are obviously strong bisimilar. Moreover, in the quotient, we have that the probability of going to the equivalence class $\{s_1, s_2\}$ from s_0 is 1. Because of this, the variable $v_{s,a}$ disappears in the quotient. Now consider the lower part. In this encoding PMC, states s_1, s_2, s_3 are weak bisimilar.

Remark For the lower part of Fig. 4, we explain below why our results do not hold when computing minimum reachability probabilities. Using the state elimination algorithm, we obtain that the probability of reaching s_4 from s_0 is 1, independently of the variable $v_{s,a}$. However, the minimum reachability probability is actually 0 instead. Moreover, states s_0, s_1, s_2 and s_3 are bisimilar; thus, in the quotient we have the probability 1 of reaching the target state directly. Thus, the information about minimum reachability probability is also lost during the state elimination and also during the weak bisimulation lumping of the encoding PMC.

3.5 Complexity

Since our algorithm is dealing with rational functions, we first discuss briefly the complexity of arithmetic for polynomials and rational functions. For more detail we refer to the literature [13]. For a polynomial f, let $mon(f)$ denote the number of monomials. Addition and subtraction of two polynomials f and g are performed by adding or subtracting coefficients of like monomials, which takes time $mon(f)+mon(g)$. Multiplication is performed by cross-multiplying each monomials, which takes $\mathcal{O}(mon(f) \cdot mon(g))$. Division of two polynomials results a rational function, which is then simplified by shortening the *greatest common divisor* (GCD), which can be obtained efficiently using a variation of the Euclid's algorithm. Arithmetic for rational functions reduces to manipulation of polynomials, for example, $\frac{f_1}{f_2} + \frac{g_1}{g_2} = \frac{f_1 g_2 + f_2 g_1}{f_2 g_2}$. Checking whether two rational functions $\frac{f_1}{f_2}$ and $\frac{g_1}{g_2}$ are equal is equivalent to checking whether $f_1 g_2 - f_2 g_1$ is a zero polynomial.

We now discuss the complexity of our algorithms. In each elimination step, we have to update the transition functions (or rewards for PMRMs) which takes $\mathcal{O}(n^2)$ polynomial operations in worst case. Thus, altogether $\mathcal{O}(n^3)$ many operations are needed to get the final function, which is the same as in the state elimination algorithm [7]. The complexity of arithmetic for polynomials depends on the degrees. The size of the final rational function is in worst case $n^{\mathcal{O}(\log n)}$.

For PMDPs, we first encode the non-deterministic choices via new binary variables. Then, the encoding PMC is submitted to the dedicated algorithm for parametric MCs. The final function can thus contain both variables from the input model and variables encoding the non-determinism. As shown in Lemma 4, the evaluation is of exponential size in the number of variables encoding the non-determinism occurring in the final rational function.

We also discuss briefly the complexity of the bisimulation minimisation algorithms. For ordinary MCs, strong bisimulation can be computed [11] in $\mathcal{O}(m \log n)$ where n, m denote the number of states and transitions, respectively. The complexity of deciding weak bisimulation [3] is $\mathcal{O}(mn)$. These algorithms can be extended to PMCs directly, with the support of operations on functions. The complexity is then $\mathcal{O}(m \log n)$ and $\mathcal{O}(mn)$ many operations on rational functions for strong and weak bisimulation, respectively.

4 Case studies

We have built the tool Param [15], which implements our new algorithms, including both the state-elimination algorithm as well as the bisimulation minimisation algorithm. Param allows a *guarded-commands* based input language supporting PMC, PMRM and PMDPs. The language is extended from Prism [19] with unknown parameters. Properties are specified by PCTL formulae without nesting.

The sparse matrices are constructed from the model, and then the set of target states **B** is extracted from the formula. Then, bisimulation minimisation can be applied to reduce the state space. For PMCs, both strong and weak bisimulation applies, and for PMRMs, currently only strong bisimulation is supported. For PMDP, bisimulation is run for the encoded PMC. We implemented methods and data structures to handle

Table 1 Performance statistics for crowds protocol

N	R	Build(s)	No bisimulation				Weak bisimulation					Result
			States	Trans.	Elim.(s)	Mem(MB)	States	Trans.	Lump(s)	Elim.(s)	Mem(MB)	
5	3	3	1192	1982	4	3	33	62	0	0	2	0.3129
5	5	3	8617	14701	57	5	127	257	0	5	6	0.3840
5	7	5	37169	64219	1878	13	353	732	1	181	15	0.4627
10	3	3	6552	14857	132	4	33	62	0	0	5	0.2540
10	5	7	111098	258441	1670	40	127	257	2	6	51	0.3159
10	7	47	989309	2332513	–	–	367	763	26	259	441	0.4062
15	3	4	19192	55132	504	10	33	62	0	0	12	0.2352
15	5	31	591455	1739356	–	–	127	257	14	7	305	0.2933
20	3	8	42298	146807	2044	22	33	63	1	0	26	0.2260

rational functions, for example, the basic arithmetic operations, comparisons and simplification. The computer algebra library CoCoALib [1] is used for handling the cancellation part. For details of the tool, we refer to the corresponding tool paper [15].

We consider a selection of case studies to illustrate the practicality of our approach. All of the models are extended from the corresponding Prism models. All experiments were run on a Linux machine with an Intel Core 2 Duo (tm) processor at 2.16 GHz equipped with 2 GB of RAM. We updated the performance figures for the case studies and also give more details about the time the different parts of the analysis need.

4.1 Crowds protocol

The intention of the Crowds protocol [29] is to protect the anonymity of Internet users. The protocol hides each user's communications via random routing. Assume that we have N honest Crowd members, and M dishonest members. Moreover, assume that there are R different path reformulates. The model is a PMC with two parameters of the model:

1. $B = \frac{M}{M+N}$ is the probability that a Crowd member is untrustworthy,
2. P is the probability that a member forwards the package to a random selected receiver.

With probability $1 - P$ it delivers the message to the receiver directly. We consider the probability that the actual sender was observed more than any other one by the untrustworthy members. For various N and R values, Table 1 summarises the time needed for computing the function representing this probability, with and without the weak bisimulation optimisation. With "Build(s)" we denote the time needed to construct the model for the analysis from the high-level Prism model. The column "Elim.(s)" states the time needed for the state elimination part of the analysis. "Mem(MB)" gives the memory usage. When applying bisimulation minimisation, "Lump(s)" denotes the time needed for computing the quotient. For entries marked by "–", the analysis did not terminate within 1 h. In the last column we evaluate the probability for $M = \frac{N}{5}$ (thus $B = \frac{1}{6}$) and $P = 0.8$. An interesting observation is that for several table entries the weak bisimulation quotient has the same size for the same R, but different probabilities. We also checked that the graph structure was the same in these cases. The reason for this is that then the other parameter N has only an effect on the transition probabilities of the quotient and not its underlying graph.

From Table 1 we also see that for this case study the usage of bisimulation helped to speed up the analysis very much. For the N and R considered, the speedup was in the range of about 10 for $N = 5$, $R = 7$ to factor such that the time for the state elimination was negligible, as for $N = 15$, $R = 3$. Also, the time for the bisimulation minimisation itself did not take longer than 26 s. Using bisimulation minimisation, we are able to handle models with several hundred thousands of states.

In Fig. 5 we give the plot for $N = 5$, $R = 7$. Observe that this probability increases with the number of dishonest members M, which is due to the fact that the dishonest members share their local information. On the contrary, this probability decreases with P. The reason is that each router forwards the message randomly with probability P. Thus, with increasing P the probability that the untrustworthy member can identify the real sender is then decreased.

4.2 Zeroconf

Zeroconf [6] allows the installation and operation of a network in the most simple way. When a new host joins the network, it randomly selects an address among the $K = 65024$ possible ones. With m hosts in the network, the collision probability is $q = \frac{m}{K}$. The host asks other hosts whether they

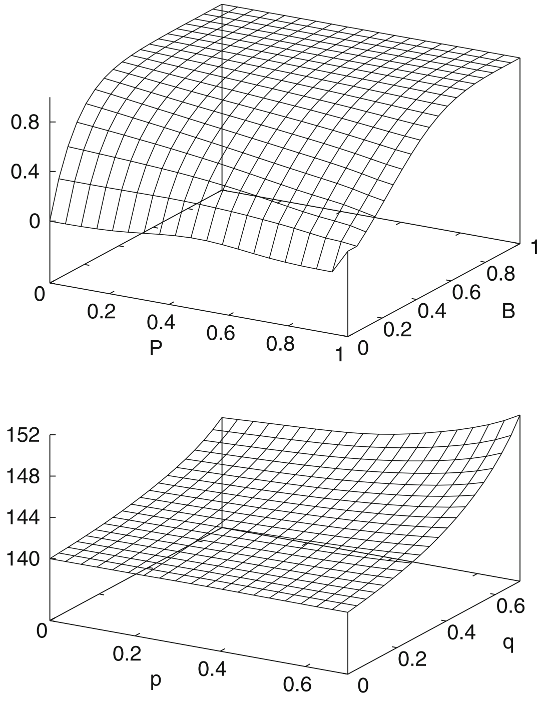

Fig. 5 *Upper* crowds protocol. *Lower* Zeroconf

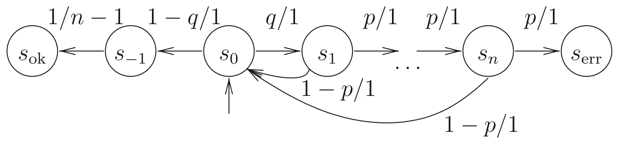

Fig. 6 Zeroconf collision detection

are using this address. If a collision occurs, the host tries to detect this by waiting for an answer. The probability that the host gets no answer in case of a collision is p, in which case he repeats the question. If after n tries the host got no answer, the host will erroneously consider the chosen address as valid. A sketch of the model is depicted in Fig. 6. We consider the expected number of tries till either the IP address is selected correctly or erroneously, that is, $\mathbf{B} = \{s_{\text{ok}}, s_{\text{err}}\}$. For $n = 140$, the plot of this function is depicted in on the lower part of Fig. 5. The expected number of tests till termination increases with both the collision probability as well as the probability that a collision is not detected. Bisimulation optimisation was not of any use, as the quotient equals the original model. For $n = 140$, the analysis took 64 s and 50 MB of memory.

4.3 Cyclic polling server

The cyclic polling server model [22] consists of a number of N stations which are handled by the polling server. Process i is allowed to send a job to the server if he owns the token, which circulates around the stations in a round robin manner. This model is a parametric continuous-time Markov chain, but we can apply our algorithm on the embedded discrete-time PMC, which has the same reachability probability. We have two parameters: the service rate μ and γ is the rate to move the token. Both are assumed to be exponentially distributed. Each station generates a new request with rate $\lambda = \frac{\mu}{N}$. Initially the token is at state 1. We consider the probability p that station 1 will be served before any other one. Table 2 summarises performance for different N. The last column corresponds to the evaluation $\mu = 1, \gamma = 200$.

On the upper part of Fig. 7 a plot for $N = 8$ is given. We have several interesting observations. If μ is greater than approximately 1.5, p first decreases and then increases with γ. The mean time of the token staying in state 1 is $\frac{1}{\gamma}$. With increasing γ, it is more probable that the token passes to the next station before station 1 sends the request. At some point, however (approximated $\gamma = 6$), p increases again as the token moves faster around the stations. For small μ the probability p always increases with increasing γ. The reason for this is that the arrival rate $\lambda = \frac{\mu}{N}$ is very small; it means also that the token moves so fast that the chance for station 1 to be scheduled at first point is rather small. Thus, by increasing the speed with which the token moves around, we give station 1 more chances to catch it. Now we fix γ to be greater than 6. Then, p decreases with μ, as increasing μ implies also a larger λ, which means that all other states become more competitive. However, for small γ we observe that μ increases later again: in this case, station 1 has a higher probability of catching the token initially at this station. Also in this model, bisimulation has quite a large effect on the time needed for the analysis.

4.4 Randomised mutual exclusion

In the randomised mutual exclusion protocol [28] several processes try to enter a critical section. We consider the protocol with two processes $i = 1, 2$. Process i tries to enter the critical section with probability p_i, and with probability $1 - p_i$, it waits until the next possibility to enter and tries again. The model is a PMRM with parameters p_i. A reward with value 1 is assigned to each transition corresponding to the probabilistic branching p_i and $1 - p_i$. We consider the expected number of coin tosses until one of the processes enters the critical section the first time. A plot of the expected number is given on the lower part of Fig. 7. This number decreases with both p_1 and p_2, because both processes have more chances to enter their critical sections. The computation took 98 s, and 5 MB of memory was used. The model consisted of 77 states and 201 non-zero transitions. Converting the transition rewards to state rewards and subsequent strong bisimulation minimisation lead only to a minimal reduction in state and transition numbers and did not reduce the analysis time.

Table 2 Performance statistics for cyclic polling server

N	Build(s)	No bisimulation				Weak bisimulation					Result
		States	Trans.	Elim.(s)	Mem(MB)	States	Trans.	Lump(s)	Elim.(s)	Mem(MB)	
4	1	66	192	1	2	13	36	0	0	3	0.2500
5	1	162	560	2	2	16	46	0	0	3	0.2000
6	1	386	1536	10	2	19	58	0	0	2	0.1667
7	1	898	4032	41	3	22	68	0	0	3	0.1429
8	1	2050	10240	150	4	25	79	0	1	4	0.1250
9	2	4610	25344	733	6	28	91	0	1	6	0.1111

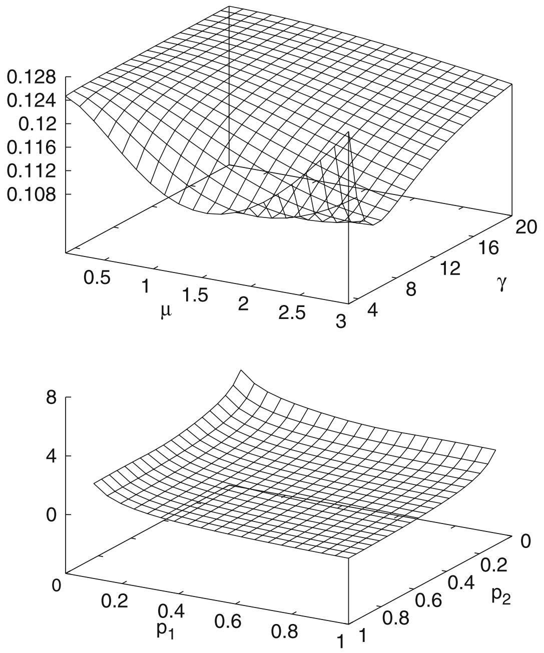

Fig. 7 *Upper* cyclic polling server. *Lower* randomised mutual exclusion

4.5 Bounded retransmission protocol

In the bounded retransmission protocol [18], a file to be sent is divided into a number of N chunks. For each of them, the number of retransmissions allowed is bounded by MAX. There are two lossy channels K and L for sending data and acknowledgements, respectively. The model is a PMDP with two parameters pK, pL denoting the reliability of the channels K and L, respectively. We consider the property "The maximum reachability probability that eventually the sender does not report a successful transmission". In Table 3 we give statistics for several different instantiations of N and MAX. The column "Nd.Vars" gives the number of variables introduced additionally to encode the non-deterministic choices. We give only running time if the optimisation is used. Otherwise, the algorithm does not terminate within 1 hour. The last column gives the probability for $pK = 0.98$ and $pL = 0.99$, as the one in the PRISM model. We observe that for all instances of N and MAX, with an increasing reliability of channel K the probability that the sender does not finally report a successful transmission decreases.

Notably, we encode the non-deterministic choices via additional variables, and apply the algorithm for the resulting parametric MCs. This approach may suffer from exponential enumerations in the number of these additional variables in the final rational function. In this case study, however, the method works quite well. This is partly owed to the fact, that after strong and weak bisimulation on the encoding PMC, the additional variables vanish as illustrated in Example 2. We are well aware, however, that still much work needs to be done to handle general non-deterministic models.

Table 3 Performance statistics for bounded retransmission protocol

N	MAX	Build(s)	Model			Weak bisimulation		Lump(s)	Elim.(s)	Mem(MB)	Result
			States	Trans.	Nd. Vars	States	Trans.				
64	4	2	8551	11564	137	643	1282	3	0	6	1.50E-06
64	5	2	10253	13916	138	771	1538	3	0	6	4.48E-08
256	4	4	33511	45356	521	2563	5122	48	6	19	6.02E-06
256	5	4	40205	54620	522	3075	6146	58	8	22	1.79E-07
512	4	12	66792	90412	1035	5123	10243	230	35	52	1.20E-05
512	5	10	80142	108892	1036	6147	12291	292	52	56	3.59E-07

Table 4 Performance comparisons with regular expressions method

Number of workstations	4	5	6	7	8
Length of regular expression (Daws' method)	191	645	2,294	8,463	32,011
Number of terms (our method)	7	9	11	13	15
Total degree (our method)	6	8	10	12	14

5 Comparison with Daws' method

Our algorithm is based on the state elimination approach, inspired by Daws [10], who treats the concrete probabilities as an alphabet, and converts the MC into a finite automaton. Then, a regular expression is computed and evaluated into functions afterwards (albeit lacking any implementation). The length of the resulting regular expression, however, has size $n^{\Theta(\log n)}$ [14] where n denotes the number of states of the automaton. Our method instead intertwines the steps of state elimination and evaluation. The size of the resulting function is in *worst case* still in $n^{\mathcal{O}(\log n)}$; thus there is no theoretical gain, pessimistically speaking.

The differences of our and Daws' method are thus on the practical side, where they indeed have dramatic implications. Our method simplifies the rational functions in each intermediate step. The worst case for our algorithm can occur only in case no rational function can be simplified during the entire process. In essence, this is the case for models where each edge of the input model has a distinguished parameter. We consider this a pathological construction. In all of the interesting models we have seen, only very few parameters appear in the input model, and it seems natural that a model designer does not deal with more than a handful of model parameters in one go. For those models, the intermediate rational functions can be simplified, leading to a space (and time) advantage. This is the reason why our method does not suffer from a blow up in the case studies considered in Sect. 4. To shed light on the differences between the two methods, we return to the cyclic polling server example:

In Table 4, we compare the two methods in terms of the observed size requirements. For each number of workstations from 4 to 8, we give the length of the regular expression arising in Daws' method. On the other hand, we give the number of terms and the total degree of the numerator and denominator polynomials of the rational function resulting from our method. The numbers for Daws' method are obtained by eliminating the states in the same order as we did for our method, namely by removing states with a lower distance to the target set first. For the length of regular expressions, we counted each occurrence of a probability as having the length 1, as well as each occurrence of the choice operator ("+") and the Kleene star ("*"). We counted braces as well as concatenation ("·") as having length zero.

As can be seen from the table, the size of the regular expression grows very fast, thus materializing the theoretical complexity. This makes the nice idea of Daws [10] infeasible in a direct implementation. For our method, both the number of terms as well as the total degree grow only linearly with the number of workstations.

6 Related work and discussions

Our work has connections to several other recent scientific contributions. Lanotte et al. [27] considered parametric MCs, showing that the problem whether there exists a well-defined evaluation is in PSPACE and whether such an evaluation induces a given reachability probability c is, however, undecidable. Recently, Damman et al. [9] have extended the approach of Daws [10] to generate counterexamples for MC. The regular expressions generated can be seen as a more compact and structured representation of counterexamples than providing a set of paths.

A closely related work is done by Han et al. [16], in which they have studied the parametric synthesis for parametric continuous-time Markov chains (PCTMCs). Instead of abstracting time by steps in discrete-time Markov chains, transitions in PCTMCs have exponential distributions. CTMCs are widely useful for modelling failure rates, produce or service rates. For time-bounded properties (probability of reaching a set of goal state within a fixed number of steps), Han et al. [16] have attacked the problem of finding the set of parameter values (forming the synthesis region) such that the property holds in the induced CTMC. While this problem is in general undecidable, an approximative method is used to solve it anyhow, in the majority of cases. This is done by discretising the continuous region and obtaining a finite number of sample points to approximate the original region. Their work is orthogonal to ours: while they study how to form the synthesis region, we give efficient algorithm for computing the rational function representing the *unbounded* reachability probability. Thus, their algorithms can be directly applied in our setting to solve the synthesis problem for unbounded reachability for PMCs. They have not yet considered reward properties. One possible future work would be to combine our approaches to consider continuous-time Markov reward models.

Another closed model in the literature is interval Markov chains [8,12,25,30]. In this model, each transition probability (or rate, for continuous-time models) is given as an interval, such that it has a minimal and a maximal value.

The Markov chains represented by an interval Markov chain are those concrete models in which all probabilities lie within the restricting intervals, and the probabilities sum up to one (for discrete-time models). This way, they can be used to represent abstractions of ordinary Markov chains [24]. Different from PMCs, results obtained are usually extremal probabilities or other values of the model, instead of a function representing all possible such values, as for PMCs.

There is also research in parametric models in the area of timed automata. For instance, timed automata are considered in which clock constraints are not fixed but given as a set of parameters [21]. Instead of checking whether the automaton fulfills the property, the paper targets at computing parameter regions of the model for which the property is fulfilled.

7 Conclusion

In this paper, we have presented algorithms for analysing parametric Markov models, possibly extended with rewards or non-determinism, together with an implementation in the tool PARAM. The PARAM source code is published under the GPL license. The tool, the case studies and additional material can be found at

http://depend.cs.uni-sb.de/tools/param.

As future work, we are investigating general improvements of the implementation with respect to memory usage and speed, especially for the setting with non-determinism. We are also investigating heuristics for the optimal order in which states are to be eliminated, referring to memory and time usage. Additionally, we plan to look into continuous time models—with clocks—, and PMDPs with rewards. Other possible directions include the use of symbolic model representations, such as MTBDD-based techniques, symbolic bisimulation minimisation [32] and also a symbolic variant of the state elimination algorithm. We would also like to explore whether our algorithm can be used for model checking interval Markov chains [30].

Acknowledgments This work is supported by the NWO-DFG bilateral project ROCKS, by the DFG as part of the Transregional Collaborative Research Center SFB/TR 14 AVACS and the Graduiertenkolleg "Leistungsgarantien für Rechnersysteme", by the European Community's Seventh Framework Programme under grant agreement n^o 214755, and is supported in part by MT-LAB, a VKR Centre of Excellence. We are grateful to Björn Wachter (Saarland University) for insightful discussions and for providing us with the parser of PASS.

A Proofs

A.1 Proof of Lemma 1

Let $\mathcal{D}$ be a PMC and $\mathbf{B}$ be a set of target states. Assume that the PMC $\mathcal{D}$ and set of states $\mathbf{B}$ satisfy the requirement of Algorithm 1, i.e., state $s \in \mathbf{B}$ is absorbing. For all $s \in S$, it holds $\mathit{reach}(s_0, s)$ and $\mathit{reach}(s, \mathbf{B})$. We show that the execution of one loop iteration in lines 1–1 in which a state s is eliminated does not change the probability of reaching $\mathbf{B}$ under any maximal well-defined evaluation function u. Let $\mathit{Path}^{\mathcal{D}}_{\mathit{fin}}(s_0, \mathbf{B}) = \{\sigma \in \mathit{Path}^{\mathcal{D}}_{\mathrm{fin}} \mid \mathit{first}(\sigma) = s_0 \wedge \mathit{last}(\sigma) \in \mathbf{B}\}$ denote the set of paths reaching $\mathbf{B}$. The probability of reaching $\mathbf{B}$ can be then expressed by the sum

$$Pr^{\mathcal{D}_u}(\mathbf{B}) = \sum_{\sigma \in \mathit{Path}^{\mathcal{D}}_{\mathit{fin}}(s_0,\mathbf{B})} Pr^{\mathcal{D}_u}(\sigma)$$

We fix an evaluation function u. We let $\mathcal{D}_1 = (S_1, s_0, \mathbf{P}_1)$ denote the PMC before elimination of the state $s \in S \backslash \mathbf{B} \cup \{s_0\}$, and let $\mathcal{D}_2 = (S_2, s_0, \mathbf{P}_2)$ denote the PMC after eliminating state s. Assume that u is maximal well defined for $\mathcal{D}_1$. By construction of the algorithm, it holds that $S_2 = S_1 \backslash \{s\}$, and that

$$\mathbf{P}_2(s_1, s_2) = \mathbf{P}_1(s_1, s_2) + \frac{\mathbf{P}_1(s_1, s)\mathbf{P}_1(s, s_2)}{1 - \mathbf{P}_1(s, s)} \tag{3}$$

Now it is sufficient to show that

$$Pr^{(\mathcal{D}_1)_u}(\mathbf{B}) = Pr^{(\mathcal{D}_2)_u}(\mathbf{B}) \tag{4}$$

For path $\sigma \in \mathit{Path}^{\mathcal{D}_1}_{\mathit{fin}}(s_0, \mathbf{B})$ in $\mathcal{D}_1$, we define the induced path in $\mathcal{D}_2$ by $\mathit{ind}(\sigma) \in \mathit{Path}^{\mathcal{D}_2}_{\mathit{fin}}(s_0, \mathbf{B})$, which is obtained by eliminating each occurrence of s in σ. For an arbitrary path $\sigma' \in \mathit{Path}^{\mathcal{D}_2}_{\mathit{fin}}(s_0, \mathbf{B})$ in $\mathcal{D}_2$, we define $\mathit{pre}(\sigma') = \{\sigma \in \mathit{Path}^{\mathcal{D}_1}_{\mathit{fin}}(s_0, \mathbf{B}) \mid \mathit{ind}(\sigma) = \sigma'\}$. The relation of σ' and $\mathit{pre}(\sigma')$ is illustrated in Fig. 8. Obviously, it holds that

$$\mathit{Path}^{\mathcal{D}_1}_{\mathit{fin}}(s_0, \mathbf{B}) = \bigcup_{\sigma' \in \mathit{Path}^{\mathcal{D}_2}_{\mathit{fin}}(s_0,\mathbf{B})} \mathit{pre}(\sigma')$$

Now to show Eq. 4, it is sufficient to show that for each $\sigma' \in \mathit{Path}^{\mathcal{D}_2}_{\mathit{fin}}(s_0, \mathbf{B})$, it holds that

$$Pr^{(\mathcal{D}_1)_u}(\mathit{pre}(\sigma')) = Pr^{(\mathcal{D}_2)_u}(\sigma') \tag{5}$$

Without loss of generality, let $\sigma' = s_0, s_1, \ldots, s_n$ with $s_n \in \mathbf{B}$. Before we show Eq. 5, we first introduce some notations. For $j = 0, \ldots, n-1$ and $i_j \in \mathbb{N}$, we define $f(s_j, i_j, s_{j+1})$ by

$$f(s_j, i_j, s_{j+1}) = \begin{cases} \mathbf{P}_1(s_j, s_{j+1}) & \text{if } i_j = 0 \\ \mathbf{P}_1(s_j, s)\mathbf{P}_1(s, s)^{i_j - 1}\mathbf{P}_1(s, s_{j+1}) & \text{if } i_j > 0 \end{cases} \tag{6}$$

For an interpretation of $f(s_j, i_j, s_{j+1})$, consider each part $\ldots s_j s^{i_j} s_{j+1} \ldots$ of the path $s_0 s^{i_0} s_1 s^{i_1} \cdots s^{i_{n-1}} s_n$:

1. $s_j s_{j+1}$, that is, no s occurs in between s_j and s_{j+1}. Then the transition probability is $\mathbf{P}_1(s_j, s_{j+1})$, as we have a direct transition from s_j to s_{j+1} at this point.

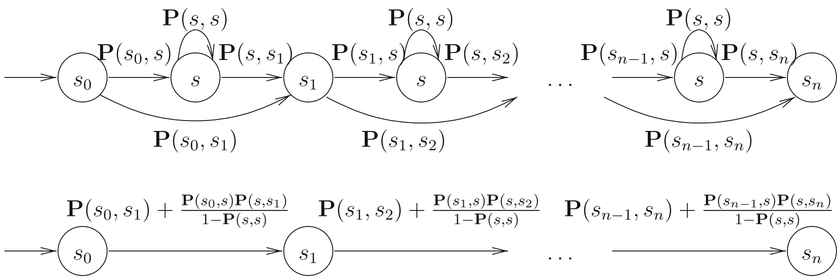
P(s, s)
P(s_0, s)
P(s, s_1)
P(s_1, s)
P(s, s_2)
P(s_{n−1}, s)
P(s, s_n)
s_0
s
s_1
s_n
...
P(s_0, s_1)
P(s_1, s_2)
P(s_{n−1}, s_n)
P(s_0, s_1) + P(s_0,s)P(s,s_1) / 1−P(s,s)
P(s_1, s_2) + P(s_1,s)P(s,s_2) / 1−P(s,s)
P(s_{n−1}, s_n) + P(s_{n−1},s)P(s,s_n) / 1−P(s,s)

Fig. 8 Illustration of state elimination proof

2. $s_j s^{i_j-1} s_{j+1}$ where $i_j > 0$, that is, there are one or several s in between. We first have a transition from s_j to s, then a number of $i_j - 1$ self-loops in s and then a transition from s to s_{j+1}, leading to the probability $\mathbf{P}_1(s_j, s)\mathbf{P}_1(s, s)^{i_j-1}\mathbf{P}_1(s, s_{j+1})$.

If s_j, s_{j+1} is clear from the context, we write simply $f(i_j)$. Thus, it holds

$$\sum_{i_j=0}^{\infty} f(i_j) = \mathbf{P}_1(s_j, s_{j+1}) + \frac{\mathbf{P}_1(s_j, s)\mathbf{P}_1(s, s_{j+1})}{1 - \mathbf{P}_1(s, s)}$$
$$= \mathbf{P}_2(s_j, s_{j+1}) \qquad (7)$$

Now we show Eq. 5:

$$\begin{aligned} &Pr^{(\mathcal{D}_1)_u}(pre(\sigma')) \\ &= \sum_{i_0=0}^{\infty}\sum_{i_1=0}^{\infty}\cdots\sum_{i_{n-1}=0}^{\infty} Pr^{(\mathcal{D}_1)u}\left(s_0 s^{i_0} s_1 s^{i_1} \ldots s^{i_{n-1}} s_n\right) \\ &= \sum_{i_0=0}^{\infty}\sum_{i_1=0}^{\infty}\cdots\sum_{i_{n-1}=0}^{\infty}\left(\prod_{j=0}^{n-1} f(i_j)\right) \\ &= \left(\sum_{i_0=0}^{\infty} f(i_0)\right)\left(\sum_{i_1=0}^{\infty} f(i_1)\right)\cdots\left(\sum_{i_{n-1}=0}^{\infty} f(i_{n-1})\right) \\ &\overset{(7)}{=} \prod_{i=0}^{n-1}\mathbf{P}_2(s_i, s_{i+1}) = Pr^{(\mathcal{D}_2)_u}(\sigma') \end{aligned}$$

Since u is maximal well defined, $1 - \mathbf{P}_1(s, s) \neq 0$, implying Eq. 5. Observe that u is also maximal well defined for $\mathcal{D}_2$, since $reach^{(\mathcal{D}_1)_u}(s_1, \mathbf{B})$ implying also

$$reach^{(\mathcal{D}_2)_u}(s_1, \mathbf{B})$$

for all $s_1 \notin \mathbf{B}$.

After the execution of lines 1–1 the model consists only of the initial state s_0 and the set of target states $\mathbf{B}$. Now we can directly compute the reachability probability:

$$Pr(s_0, \mathbf{B}) = \sum_{i=0}^{\infty}\sum_{s\in\mathbf{B}} \mathbf{P}(s_0, s_0)^i \mathbf{P}(s_0, s)$$
$$= \frac{1}{1 - \mathbf{P}(s_0, s_0)}\mathbf{P}(s_0, \mathbf{B})$$

□

A.2 Proof of Lemma 2

Let $\mathcal{R} = (\mathcal{D}, r)$ be a PMRM and $\mathbf{B}$ be a set of target states. Assume that the PMRM $\mathcal{R}$ and set of states $\mathbf{B}$ satisfy the requirement of Algorithm 2, i.e., state $s \in \mathbf{B}$ is absorbing. For all $s \in S$, it holds $reach(s_0, s)$ and $reach(s, \mathbf{B})$. We show that the execution of one loop iteration in lines 2–2 in which a state s is eliminated does not change the expected reward until $\mathbf{B}$ is reached under any maximal well-defined evaluation function u.

We fix an evaluation function u. We let $\mathcal{R}_1 = (\mathcal{D}_1, r_1)$ with $\mathcal{D}_1 = (S_1, s_0, \mathbf{P}_1)$ denote the PMRM before eliminating state $s \in S\backslash\mathbf{B} \cup \{s_0\}$, and let $\mathcal{R}_2 = (\mathcal{D}_2, r_2)$ with $\mathcal{D}_2 = (S_2, s_0, \mathbf{P}_2)$ denote the PMRM after eliminating state s. As for PMCs, it holds that $S_2 = S_1\backslash\{s\}$, and the matrix $\mathbf{P}_2$ is as defined in Eq. 3. For $s, s' \in S_1$, we let $r_1^*(s, s')$ denote the reward $r_1(s) + r_1(s, s')$. The value $r_2^*(s, s')$ is defined likewise. We now discuss how the reward function r_2 is obtained in Algorithm 2. The reward of a state $s' \in S_2$ does not change: $r_2(s') = r_1(s')$. For $s_1, s_2 \in S_2$, let

$$p_e(s_1, s_2) := \frac{\mathbf{P}_1(s_1, s)\mathbf{P}_1(s, s_2)}{1 - \mathbf{P}_1(s, s)} \qquad (8)$$

$$r_e(s_1, s_2) := r_1(s_1, s) + r_1^*(s, s_2) + \frac{\mathbf{P}_1(s, s)}{1 - \mathbf{P}_1(s, s)} r_1^*(s, s) \qquad (9)$$

$$r_2(s_1, s_2) := \frac{p_e(s_1, s_2) r_e(s_1, s_2) + \mathbf{P}_1(s_1, s_2) r_1(s_1, s_2)}{p_e(s_1, s_2) + \mathbf{P}_1(s_1, s_2)} \qquad (10)$$

$$\mathbf{P}_2(s_1, s_2) = p_e(s_1, s_2) + \mathbf{P}_1(s_1, s_2) \qquad (11)$$

as defined in the algorithm. Note that in case $\mathbf{P}_1(s_1, s) = 0$, we have $p_e(s_1, s_2) = 0$ implying that $r_2(s_1, s_2) = r_1(s_1, s_2)$. Assume that u is maximal well defined for $\mathcal{D}_1$. Now it is sufficient to show that

$$R^{(\mathcal{R}_1)_u}(s_0, \mathbf{B}) = R^{(\mathcal{R}_2)_u}(s_0, \mathbf{B}) \qquad (12)$$

Now with the notation of the proof of Lemma 1, we prove that for each $\sigma' \in Path_{fin}^{\mathcal{D}_2}(s_0, \mathbf{B})$, it holds that

$$R^{(\mathcal{R}_1)_u}(pre(\sigma')) = R^{(\mathcal{R}_2)_u}(\sigma') \qquad (13)$$

where $R^{\mathcal{R}}(\sigma) = Pr^{\mathcal{D}}(\sigma)X^{\mathcal{R}}(\sigma)$ for $\mathcal{R}$, and $R^{\mathcal{R}}(C) = \sum_{\sigma\in C} R^{\mathcal{R}}(\sigma)$, for which we extended $X^{\mathcal{R}}$ to finite paths ending in **B** in an obvious way. Without loss of generality, let $\sigma' = s_0, s_1, \ldots, s_n$ with $s_n \in \mathbf{B}$. For $j = 0, \ldots, n-1$ and $i_j \in \mathbb{N}$, the function $f(s_j, i_j, s_{j+1})$ is as defined in Eq. 6. Moreover, we define $g(s_j, i_j, s_{j+1})$ by

$$g(s_j, i_j, s_{j+1}) = \begin{cases} r_1^*(s_j, s_{j+1}) & \text{if } i_j = 0 \\ r_1^*(s_j, s) + r_1^*(s, s_{j+1}) + (i_j - 1)(r_1^*(s, s)) & \text{if } i_j > 0 \end{cases} \tag{14}$$

If s_j and s_{j+1} are clear from the context, we write $f(i_j)$ and $g(i_j)$ instead. Similar to $f(i_j)$, $g(i_j)$ denotes the rewards gained via visiting the path segment $\ldots s_j s_{j+1} \ldots$ for the case $i_j = 0$, or $\ldots s_j s^{i_j} s_{j+1} \ldots$ for the case $i_j > 0$. Now we show Equation 13:

$$\begin{aligned}
&R^{(\mathcal{R}_1)_u}(pre(\sigma')) \\
&= \sum_{i_0=0}^{\infty} \sum_{i_1=0}^{\infty} \cdots \sum_{i_{n-1}=0}^{\infty} Pr^{(\mathcal{D}_1)_u}\left(s_0 s^{i_0} s_1 s^{i_1} \ldots s^{i_{n-1}} s_n\right) \\
&\quad \cdot X^{(\mathcal{R}_1)_u}\left(s_0 s^{i_0} s_1 s^{i_1} \ldots s^{i_{n-1}} s_n\right) \\
&= \sum_{i_0=0}^{\infty} \sum_{i_1=0}^{\infty} \cdots \sum_{i_{n-1}=0}^{\infty} \left(\prod_{j=0}^{n-1} f(i_j) \sum_{k=0}^{n-1} g(i_k) \right) \\
&= \sum_{k=0}^{n-1} \left(\sum_{i_0=0}^{\infty} \sum_{i_1=0}^{\infty} \cdots \sum_{i_{n-1}=0}^{\infty} \prod_{j=0}^{n-1} f(i_j) g(i_k) \right) \\
&= \sum_{k=0}^{n-1} \left(\sum_{i_0=0}^{\infty} f(i_0) \right) \cdots \left(\sum_{i_{k-1}=0}^{\infty} f(i_{k-1}) \right) \left(\sum_{i_k=0}^{\infty} f(i_k) g(i_k) \right) \\
&\quad \cdot \left(\sum_{i_{k+1}=0}^{\infty} f(i_{k+1}) \right) \cdots \left(\sum_{i_{n-1}=0}^{\infty} f(i_{n-1}) \right) \\
&\overset{(7)}{=} \sum_{k=0}^{n-1} \left(\prod_{i=0}^{n-1} \mathbf{P}_2(s_i, s_{i+1}) \cdot \frac{\sum_{i_k=0}^{\infty} f(i_k) g(i_k)}{\mathbf{P}_2(s_k, s_{k+1})} \right) \\
&= \left(\prod_{i=0}^{n-1} \mathbf{P}_2(s_i, s_{i+1}) \right) \left(\sum_{k=0}^{n-1} \frac{\sum_{i_k=0}^{\infty} f(i_k) g(i_k)}{\mathbf{P}_2(s_k, s_{k+1})} \right) \\
&= Pr^{(\mathcal{D}_2)_u}(\sigma') \left(\sum_{k=0}^{n-1} \frac{\sum_{i_k=0}^{\infty} f(i_k) g(i_k)}{\mathbf{P}_2(s_k, s_{k+1})} \right)
\end{aligned}$$

By definition, we have $R^{(\mathcal{R}_2)}(\sigma') = Pr^{(\mathcal{D}_2)_u}(\sigma') X^{(\mathcal{R}_2)_u}(\sigma')$. Recall for a path σ', it holds $X^{\mathcal{R}_2}(\sigma') = \sum_{k=0}^{n-1} r_2^*(s_k, s_{k+1})$; thus, it is now sufficient to show that for each $k = 0, \ldots, n-1$, it holds that

$$\sum_{i_k=0}^{\infty} f(i_k) g(i_k) = \mathbf{P}_2(s_k, s_{k+1}) r_2^*(s_k, s_{k+1}) \tag{15}$$

According to Eq. 14, $\sum_{i_k=1}^{\infty} f(i_k) g(i_k)$ equals

$$\begin{aligned}
&\left(\left(r_1^*(s_k, s) + r_1^*(s, s_{k+1}) \right) \sum_{i_k=1}^{\infty} f(i_k) \right) \\
&\quad + \left(r_1^*(s, s) \sum_{i_k=1}^{\infty} f(i_k)(i_k - 1) \right)
\end{aligned} \tag{16}$$

By Eqs. 7 and 8, it holds that

$$\sum_{i_k=1}^{\infty} f(i_k) = p_e(s_k, s_{k+1}) \tag{17}$$

For $0 \le p_c < 1$, the sum $\sum_{i=0}^{\infty} i p_c^i$ can be simplified to $\frac{p_c}{(1-p_c)^2}$. Using this, we can simplify the second sum of Equation 16:

$$\sum_{i_k=1}^{\infty} f(i_k)(i_k - 1) = p_e(s_k, s_{k+1}) \frac{\mathbf{P}_1(s, s)}{1 - \mathbf{P}_1(s, s)} \tag{18}$$

Now putting Eqs. 16,17,18 together with Eq. 9, we have

$$\begin{aligned}
&\sum_{i_k=1}^{\infty} f(i_k) g(i_k) \\
&\quad = (r_e(s_k, s_{k+1}) + r_1(s_k)) p_e(s_k, s_{k+1}) \\
&\quad \overset{(10)}{=} \mathbf{P}_2(s_k, s_{k+1}) r_2(s_k, s_{k+1}) \\
&\qquad - \mathbf{P}_1(s_k, s_{k+1}) r_1(s_k, s_{k+1}) + r_1(s_k) p_e(s_k, s_{k+1}) \\
&\quad \overset{(11)}{=} \mathbf{P}_2(s_k, s_{k+1}) r_2^*(s_k, s_{k+1}) - \mathbf{P}_1(s_k, s_{k+1}) r_1^*(s_k, s_{k+1})
\end{aligned}$$

which proves Eq. 15. This means that the expected accumulated reward till **B** is reached is equal in the old and the new models. After the execution of lines 2–2 the remaining paths with non-zero probabilities from the initial states to **B** all have a length of 1. Because of this, in line 2 the expected reward can be obtained directly from the probability matrix **P** and reward matrix r. As before, let $r^*(s, s')$ denote $r(s) + r(s, s')$. Then,

$$\begin{aligned}
&R^{(R_2)_u}(s_0, \mathbf{B}) \\
&\quad = \sum_{s \in \mathbf{B}} \sum_{i=0}^{\infty} \mathbf{P}(s_0, s_0)^i \mathbf{P}(s_0, s) \left(r^*(s_0, s) + i \cdot r^*(s_0, s_0) \right) \\
&\quad = \sum_{s \in \mathbf{B}} \left(\mathbf{P}(s_0, s) r^*(s_0, s) \sum_{i=0}^{\infty} \mathbf{P}(s_0, s_0)^i \right. \\
&\qquad \left. + \mathbf{P}(s_0, s) r^*(s_0, s_0) \sum_{i=0}^{\infty} i \cdot \mathbf{P}(s_0, s_0)^i \right) \\
&\quad = \sum_{s \in \mathbf{B}} \left(\frac{\mathbf{P}(s_0, s) r^*(s_0, s)}{1 - \mathbf{P}(s_0, s_0)} + \frac{\mathbf{P}(s_0, s) r^*(s_0, s_0) \mathbf{P}(s_0, s_0)}{(1 - \mathbf{P}(s_0, s_0))^2} \right)
\end{aligned}$$

The proof to show that no divisions by zero occur and that the evaluation function u is still maximal well defined is analogous to the one in A.1 □

References

1. Abbott, J.: The design of CoCoALib. In: ICMS, pp. 205–215 (2006)
2. Baier, C., Ciesinski, F., Größer, M.: ProbMela and verification of Markov decision processes. SIGMETRICS **32**(4), 22–27 (2005)
3. Baier, C., Hermanns, H.: Weak bisimulation for fully probabilistic processes. In: CAV, pp. 119–130 (1997)
4. Baier, C., Katoen, J.-P., Hermanns. H., Wolf, V.: Comparative branching-time semantics for Markov chains. Inf. Comput. **200**(2), 149–214 (2005)
5. Bianco, A., de Alfaro, L.: Model checking of probabilistic and nondeterministic systems. FSTTCS **15**, 499–513 (1995)
6. Bohnenkamp, H.C., van der Stok, P., Hermanns, H., Vaandrager, F.W.: Cost-optimization of the IPv4 zeroconf protocol. In: DSN, pp. 531–540 (2003)
7. Brzozowski, J.A., Mccluskey, E.J.: Signal flow graph techniques for sequential circuit state diagrams. IEEE Trans. Electron. Comp. EC **12**, 67–76 (1963)
8. Chatterjee, K., Henzinger, T., Sen, K.: Model-checking omega-regular properties of interval Markov chains. In: FoSSaCS, pp. 302–317 (2008)
9. Damman, B., Han, T., Katoen, J.-P.: Regular expressions for PCTL counterexamples. In: QEST (2008)
10. Daws, C.: Symbolic and parametric model checking of discrete-time Markov Chains. In: ICTAC, pp. 280–294 (2004)
11. Derisavi, S., Hermanns, H., Sanders, W.H.: Optimal state-space lumping in Markov chains. Inf. Process. Lett. **87**(6), 309–315 (2003)
12. Fecher, H., Leucker, M., Wolf, V.: Don't know in probabilistic Systems. In: SPIN, pp. 71–88 (2006)
13. Geddes, K.O., Czapor, S.R., Labahn, G.: Algorithms for Computer Algebra. Kluwer, Dordrecht (1992)
14. Gruber, H., Johannsen, J.: Optimal lower bounds on regular expression size using communication complexity. In: FoSSaCS, pp. 273–286 (2008)
15. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: PARAM: a model checker for parametric Markov models. In: CAV, 2010 (to appear)
16. Han, T., Katoen, J.-P., Mereacre, A.: Approximate parameter synthesis for probabilistic time-bounded reachability. In: RTSS, pp. 173–182 (2008)
17. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. FAC **6**(5), 512–535 (1994)
18. Helmink, L., Sellink, A., Vaandrager, F.W.: Proof-checking a data link protocol. In: TYPES, vol. 806, pp. 127–165. Springer, Heidelberg (1994)
19. Hinton, A., Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM: a tool for automatic verification of probabilistic systems. In: TACAS, pp. 441–444 (2006)
20. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to automata theory, languages, and computation, 2nd edn. SIGACT News **32**(1), 60–65 (2001)
21. Hune, T., Romijn, J., Stoelinga, M., Vaandrager, F.W.: Linear parametric model checking of timed automata. J. Log. Algebra Program. **52**(53), 183–220 (2002)
22. Ibe, O.C., Trivedi, K.S.: Stochastic petri net models of polling systems. IEEE J. Selected Areas Commun. **8**(9), 1649–1657 (1990)
23. Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: LICS, pp. 266–277. IEEE Computer Society, New York (1991)
24. Katoen, J.-P., Klink, D., Leucker, M., Wolf, V.: Three-valued abstraction for continuous-time markov chains. In: CAV, vol. 4590, pp. 311–324. Springer, Heidelberg (2007)
25. Kozine, I., Utkin, L.V.: Interval-valued finite Markov chains. Reliable Comput. **8**(2), 97–113 (2002)
26. Kwiatkowska, M.Z., Norman, G., Parker, D.: Stochastic model checking. In: SFM, pp. 220–270 (2007)
27. Lanotte, R., Maggiolo-Schettini, A., Troina, A.: Parametric probabilistic transition systems for system design and analysis. FAC **19**(1), 93–109 (2007)
28. Pnueli, A., Zuck, L.: Verification of multiprocess probabilistic protocols. Distrib. Comput. **1**(1), 53–72 (1986)
29. Reiter, M.K., Rubin, A.D.: Crowds: anonymity for web transactions. ACM Trans. Inf. Syst. Secur. **1**(1), 66–92 (1998)
30. Sen, K., Viswanathan, M., Agha, G.: Model-checking Markov chains in the presence of uncertainties. In: TACAS, pp. 394–410 (2006)
31. Stewart, W.J.: Introduction to the Numerical Solution of Markov Chains. Princeton University Press, Princeton (1994)
32. Wimmer, R., Derisavi, S., Hermanns, H.: Symbolic partition refinement with dynamic balancing of time and space. In: QEST, pp. 65–74 (2008)