Probabilistic Reachability for Parametric Markov Models*

Ernst Moritz Hahn[1], Holger Hermanns[1,2], and Lijun Zhang[1]

[1] Saarland University, Saarbrücken, Germany
[2] INRIA Grenoble – Rhône-Alpes, France

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.

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]. In this context, we might aim, for instance, at determining parametric reachability probabilities, 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 [8] 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

* This work is supported by the NWO-DFG bilateral project VOSS, by the DFG as part of the Transregional Collaborative Research Center SFB/TR 14 AVACS and the Graduiertenkolleg "Leistungsgarantien für Rechnersysteme", and has received funding from the European Community's Seventh Framework Programme under grant agreement n^o 214755.

as a finite automaton. Then, based on the *state elimination* [16] method, the regular expression describing the language of such an automaton is calculated. In a postprocessing step, this regular expression is recursively evaluated resulting in a rational function over the parameters of the model. Recently, Gruber and Johannsen [11] 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 [8] is that instead of postprocessing 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 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 of [8]. 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 (MRMs) 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 $\mathbf{B}$. Intuitively, this amounts to the sum of probabilities of paths leading to $\mathbf{B}$, each weighted with the reward accumulated along this path. A direct extension of the state elimination approach does not work for MRMs, as in the final regular expression the probability of individual paths leading to $\mathbf{B}$ 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 is then submitted to the dedicated algorithm for parametric MCs.

In all settings, we reduce the state space prior to state elimination, by extending standard strong [9] and weak bisimulation [3] lumping techniques (computing the quotient model for further analysis) to parametric MCs and MRMs. 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, including the parametric bisimulation lumping. We illustrate the feasibility of the entire approach on a number of non-trivial parametric MCs, MRMs and MDPs.

Our work has connections to several other recent scientific contributions. In [20], Lanotte *et al.* 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. For parametric continuous-time Markov chains, Han *et al.* [13] have provided approximation algorithms to find valid valuations with respect to time bounded reachability properties. Recently, Damman *et al.* [7] have extended the approach of [8] 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.

Organisation of the paper. In Section 2 we introduce the parametric models used in this paper. Then, in Section 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 Section 4. In Section 5, we compare our method to the original approach of Daws. Finally, Section 6 concludes this paper. Proofs can be found in [12].

2 Parametric Models

In this section we present the parametric models which we will use throughout the paper. Firstly, we introduce some general notations. Let S be a finite set. For a relation $E \subseteq S \times S$, let $E(s) = \{s' \mid (s, s') \in E\}$, and $E^{-1}(s) = \{s' \mid (s', s) \in E\}$. 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 u is total if $Dom(u) = V$. A *polynomial* g over V is a sum of monomials $g(x_1, \ldots, x_n) = \sum_{i_1,\ldots,i_n} a_{i_1,\ldots,i_n} x_1^{i_1} \cdots x_n^{i_n}$ 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 [8,20]. 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 $pre(s) = E^{-1}(s)$ denote the set of predecessors of s, and let $post(s) = E(s)$ 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. *Given a PMC* $\mathcal{D} = (S, s_0, \mathbf{P}, V)$ *and an evaluation* u*. The induced PMC* $\mathcal{D}_u$ *by* u *is defined by* $\mathcal{D}_u = (S, s_0, \mathbf{P}_u, V \setminus Dom(u))$ *where the transition matrix* $\mathbf{P}_u : S \times S \to \mathcal{F}_V$ *is defined 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 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, at the other hand, that the probability of success is 0.

An infinite path is an infinite sequence $\sigma = s_0 s_1 s_2 \dots$ of states, and a finite path is a finite sequence $\sigma = s_0 s_1 s_2 \dots s_n$ of states. A finite path $\sigma = s_0 s_1 s_2 \dots s_n$ 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. Let $Path^{\mathcal{D}}$ denote the set of maximal paths of $\mathcal{D}$, $Path_{\text{fin}}$ 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 [24] 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 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. 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 5. *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, 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 6. *Given a PMDP $\mathcal{M} = (S, s_0, Act, \mathbf{P}, V)$ and an evaluation u, the PMDP induced by u is defined by $\mathcal{M}_u = (S, s_0, Act, \mathbf{P}_u, V \setminus Dom(u))$ where $\mathbf{P}_u : S \times Act \times S \to \mathcal{F}_V$ 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 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: 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 $\text{MD}(\mathcal{M})$ we denote the set of stationary schedulers of $\mathcal{M}$. A stationary scheduler induces a PMC as follows:

Definition 7. *Given a PMDP $\mathcal{M} = (S, s_0, Act, \mathbf{P}, V)$ and a stationary scheduler A, the* induced PMC *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 $A \in \text{MD}(\mathcal{M})$, 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 [18] and weak bisimulation [4] relations for Markov models to our parametric setting in an obvious way.

Definition 8. *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. For PMRMs, strong bisimulation is obtained by additionally requiring that $r(s_1) = r(s_2)$ and $r(s_1, C) = r(s_2, C)$ for all $C \in S/R$ if $s_1 R s_2$. Now we give the notion of weak bisimulation:

Definition 9. *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 Subsection 3.2 and Subsection 3.3 respectively. In Subsection 3.4 we discuss how to minimise parametric models using bisimulation, and we discuss the complexity of our algorithm in Subsection 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 valuations. This is defined by $Pr^{\mathcal{D}}(\{\sigma \mid \sigma[0] = s_0 \land \exists i.\sigma[i] \in \mathbf{B}\})$.

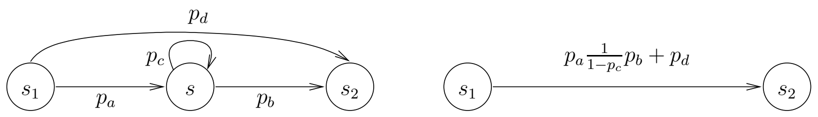
Daws [8] has already solved this problem as follows. First, the PMC is transformed into a finite automaton, with the same initial state, and $\mathbf{B}$ as the final states. Transition probabilities are described by symbols from an alphabet of the automaton of the form $\frac{p}{q}$ or x representing rational numbers, or variables. Afterwards, based on the *state elimination* [16] 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 $n^{\Theta(\log n)}$ [11].

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

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 $reach(s, \mathbf{B})$.
1: for all $s \in S \setminus (\{s_0\} \cup \mathbf{B})$ do
2:     for all $(s_1, s_2) \in pre(s) \times post(s)$ do
3:         $\mathbf{P}(s_1, s_2) = \mathbf{P}(s_1, s_2) + \mathbf{P}(s_1, s)\frac{1}{1-\mathbf{P}(s,s)}\mathbf{P}(s, s_2)$
4:     $eliminate(s)$
5: return $\frac{1}{1-\mathbf{P}(s_0,s_0)}\mathbf{P}(s_0, \mathbf{B})$
```

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 can not 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 the figure above. 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 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 $\frac{p_a p_b}{1-p_c}$ 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.

Now we discuss the correctness of our algorithm. Consider the simple PMC in the figure above. 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 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 can not 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 10. *Assume that the PMC $\mathcal{D}$ and 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 $reach^{\mathcal{D}_u}(s, \mathbf{B})$.*

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 $Pr^{\mathcal{D}_u}(s, \mathbf{B}) = 1$
 1: for all $s \in S \setminus (\{s_0\} \cup \mathbf{B})$ do
 2:   for all $(s_1, s_2) \in pre(s) \times post(s)$ do
 3:     $p_e = \mathbf{P}(s_1, s) \frac{1}{1-\mathbf{P}(s,s)} \mathbf{P}(s, s_2)$
 4:     $r_e = r(s_1, s) + r(s, s_2) + r(s) + \frac{\mathbf{P}(s,s)}{1-\mathbf{P}(s,s)} (r(s, s) + r(s))$
 5:     $r(s_1, s_2) = \frac{p_e r_e + \mathbf{P}(s_1, s_2) r(s_1, s_2)}{p_e + \mathbf{P}(s_1, s_2)}$
 6:     $\mathbf{P}(s_1, s_2) = \mathbf{P}(s_1, s_2) + p_e$
 7:   $eliminate(s)$
 8: return $\sum_{s \in \mathbf{B}} \{ \frac{\mathbf{P}(s_0,s)}{1-\mathbf{P}(s_0,s_0)} (r(s_0) + r(s_0, s)) + \frac{\mathbf{P}(s_0,s_0)\mathbf{P}(s_0,s)}{(1-\mathbf{P}(s_0,s_0))^2} (r(s_0, s_0) + r(s_0)) \}$
```

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 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 [12]. We can handle non-maximal evaluations by reducing it 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* [19] until $\mathbf{B}$, which is denoted by $R^{\mathcal{R}}(s_0, \mathbf{B})$. Formally, $R^{\mathcal{R}}(s_0, \mathbf{B})$ is the *expectation* of the random variable $X : \sigma \in Path^{\mathcal{D}}(s_0) \to \mathbb{R}_{\geq 0}$ which is defined by: $X(\sigma)$ equals 0 if *first*$(\sigma) \in \mathbf{B}$, ∞ if $\sigma[i] \notin \mathbf{B}$ for all i, and equals $\sum_{i=0}^{\min\{j \mid \sigma[j] \in \mathbf{B}\}-1} r(\sigma[i]) + r(\sigma[i], \sigma[i+1])$ otherwise.

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 $R^{\mathcal{R}}(s_0, \mathbf{B}) = \sum_{\sigma} Pr(\sigma) \cdot X(\sigma)$ where σ ranges over all maximal paths of $\mathcal{D}$. The key part of the algorithm is the adaption of the state elimination algorithm

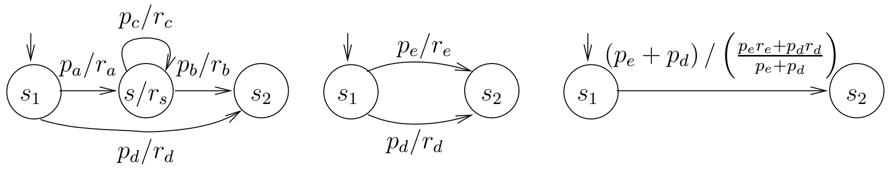


Fig. 1. For MRMs: $p_e = \frac{p_a p_b}{1-p_c}$, $r_e = r_a + r_b + r_s + \frac{p_c}{1-p_c}(r_c + r_s)$

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 Figure 1, 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 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 our definition, the expected accumulated rewards would be

$$\begin{aligned} g(s_1, s_2) &:= \sum_{i=0}^{\infty} (p_a p_c^i p_b) \cdot (r_a + r_s + (r_c + r_s)i + r_b) \\ &= (r_a + r_s + r_b)\frac{p_a p_b}{1-p_c} + p_a p_b (r_c + r_s) \sum_{i=0}^{\infty} i p_c^i \end{aligned}$$

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.

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, $\mathbf{B} \subseteq S$ a set of target states. Our goal of 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 valuation u, with A ranging over all schedulers. For the ordinary MDP case (e.g. $\mathcal{M}_u$ where u is strictly well-defined), as shown in [5], 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 in [8]. For state $s \in S$ with $k = |Act(s)|$ non-deterministic choices, we need to introduce $k-1$ variables.

Definition 11. *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 with respect to a from s. 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 $v_{s,a}$ indicates that $\delta(s)$ is taken. Now we define the encoding of $\mathcal{M}$ with respect to Var_δ.

Definition 12. *Let $\mathcal{M} = (S, s_0, Act, \mathbf{P}, V)$ be a PMDP. The* encoding PMC *with respect to Var_δ is defined as $\mathrm{enc}(\mathcal{M}) = (S, s_0, \mathbf{P}_\delta, V \dot\cup Var_\delta)$ 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 $a \in Act(s) \setminus \{\delta(s)\}$ 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 $a \in Act(s) \setminus \{\delta(s)\}$, 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 $(\mathrm{enc}(\mathcal{M}))_v = \mathcal{M}_A$.*

Due to the fact that 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:*

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

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 $s_1 \approx_{\mathcal{D}_u} s_2$.*

Both strong and weak bisimulation preserve the reachability probability for ordinary MCs [14,3]. 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 MRMs.

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 $\mathrm{enc}(\mathcal{M})$. Since both strong and weak bisimulation preserve reachability for PMCs, by Lemma 3 and Lemma 4, bisimulation minimisation on the encoded PMC $\mathrm{enc}(\mathcal{M})$ 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 left of Figure 2. 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 right part. In this encoding PMC, states s_1, s_2, s_3 are weak bisimilar.

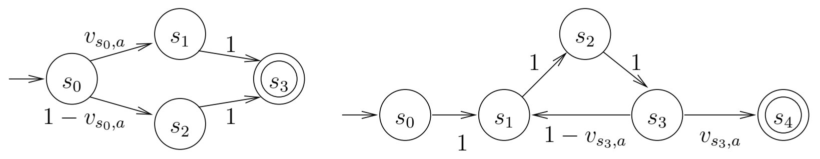


Fig. 2. Bisimulation for PMDPs

Remark: For the right part of Figure 2, we explain below why our results do not hold to obtain 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 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 [10]. 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 [6]. 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 [9] 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, 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 MC, MRM and MDPs. The language is extended from Prism [15] 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** are extracted from the formula. Then, bisimulation minimisation can be applied to reduce the state space. For MCs, both strong and weak bisimulation applies, and for MRMs, currently only strong bisimulation is supported. For PMDP, bisimulation is run for the encoded PMC. We use the computer algebra library CoCoALib[1] for handling arithmetic of rational functions, for example the basic arithmetic operations, comparisons and simplification.

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 AMD Athlon(tm) XP 2600+ processor at 2 GHz equipped with 2GB of RAM.

Crowds Protocol. The intention of the Crowds protocol [22] 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: (i) $B = \frac{M}{M+N}$ is the probability that a Crowd member is untrustworthy, (ii) 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, the following table summarises the time needed for computing the function representing this probability, with and without the weak bisimulation optimisation. 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 the weak bisimulation quotient has the same size for the same R, but different probabilities. The reason for this is that the other parameter N has only an effect on the transition probabilities of the quotient and not its underlying graph.

N	R	no bisimulation				weak bisimulation				Result
		States	Trans.	Time(s)	Mem(MB)	States	Trans.	Time(s)	Mem(MB)	
5	3	1192	2031	6	6	33	62	3	6	0.3129
5	5	8617	14916	73	22	127	257	22	21	0.3840
5	7	37169	64888	1784	84	353	732	234	84	0.4627
10	3	6552	15131	80	18	33	62	16	17	0.2540
10	5	111098	261247	1869	245	127	257	504	245	0.3159
15	3	19192	55911	508	47	33	62	51	47	0.2352

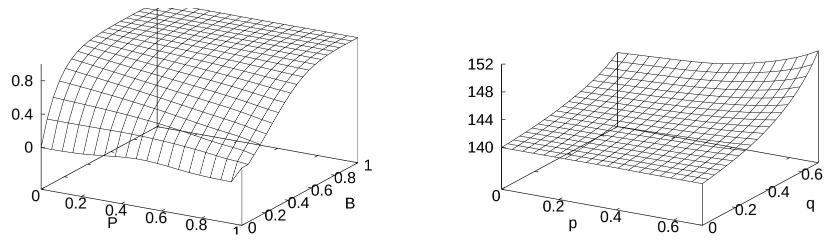


Fig. 3. Left: Crowds Protocol. Right: Zeroconf

In Figure 3 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.

$$\begin{tikzcd}
s_{\mathrm{ok}} & s_{-1} \arrow[l, "1/n-1"'] & s_0 \arrow[l, "1-q/1"'] \arrow[r, "q/1"] & s_1 \arrow[r, "p/1"] \arrow[l, bend left, "1-p/1"] & s_2 \arrow[r, "p/1"] \arrow[ll, bend left, "1-p/1"] & \cdots \arrow[r, "p/1"] & s_n \arrow[r, "p/1"] \arrow[llll, bend left, "1-p/1"] & s_{\mathrm{err}} \\
 & & {} \arrow[u] & & & & &
\end{tikzcd}$$

Zeroconf. Zeroconf 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 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 not answer in case of 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 the figure above. We consider the expected number of tries till either the IP address is selected correctly or erroneously that is, $\mathbf{B} = \{s_{\mathrm{ok}}, s_{\mathrm{err}}\}$. For $n = 140$, the plot of this function is depicted in on the right part of Figure 3. 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 seconds and 50 MB of memory.

Cyclic Polling Server. The cyclic polling server [17] 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, circulating 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. The following table summarises performance for different N. The last column corresponds to the evaluation $\mu = 1, \gamma = 200$.

N	no bisimulation				weak bisimulation				Result
	States	Trans.	Time(s)	Mem(MB)	States	Trans.	Time(s)	Mem(MB)	
4	89	216	1	3	22	55	1	3	0.25
5	225	624	3	3	32	86	1	3	0.20
6	545	1696	10	4	44	124	3	4	0.17
7	1281	4416	32	5	58	169	7	5	0.14
8	2945	11136	180	7	74	221	19	8	0.12

On the left of Figure 4 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 pasts 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 is always increasing. The reason for this is that the arrival rate $\lambda = \frac{\mu}{N}$ is very small, which means also that the token moves faster. 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.

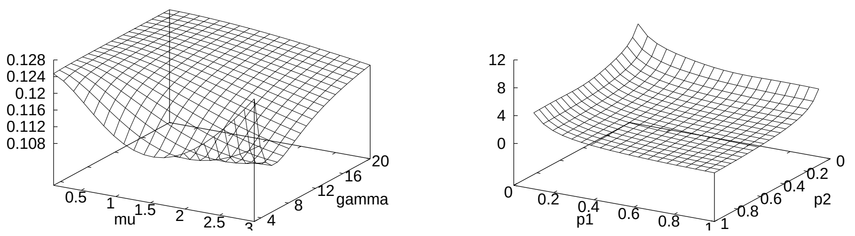


Fig. 4. Left: Cyclic Polling Server. Right: Randomised Mutual Exclusion

Randomised Mutual Exclusion. In the randomised mutual exclusion protocol [21] 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 right part of Figure 4. This number decreases with both p_1 and p_2, because both processes have more chance to enter their critical sections. The computation took 98 seconds, and 5 MB of memory was used. The model consisted of 77 states and 201 non-zero transitions. In the quotient, there were 71 states and 155 non-zero transitions.

Bounded Retransmission Protocol. In the bounded retransmission protocol, 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 the following table 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 one 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.

N	MAX	model			weak bisimulation		Time(s)	Mem(MB)	Result
		States	Trans.	Nd.Vars	States	Trans.			
64	4	8551	11569	137	643	1282	23	16	1.50E-06
64	5	10253	13922	138	771	1538	28	19	4.48E-08
256	4	33511	45361	521	2563	5122	229	63	6.02E-06
256	5	40205	54626	522	3075	6146	371	69	1.79E-07

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.

5 Comparison with Daws' Method

Our algorithm is based on the state elimination approach, inspired by Daws [8], 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 $n^{\Theta(\log n)}$ [11] 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 Section 4. To shed light on the differences between the two methods, we return to the cyclic polling server example:

Number of workstations	4	5	6	7	8
Length of regular expression (Daws' method)	191	645	2294	8463	32011
Number of terms (our method)	7	9	11	13	15
Total degree (our method)	6	8	10	12	14

In the table above, 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 nominator 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 [8] 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 Conclusion

We have presented algorithms for analysing parametric Markov models, possibly extended with rewards or non-determinism. 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 also 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 [25], 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 [23].

Acknowledgements. We are grateful to Björn Wachter (Saarland University) for insightful discussions and for providing us with the parser of PASS.

References

1. J. Abbott. The design of cocoalib. In *ICMS*, pages 205–215, 2006.
2. C. Baier, F. Ciesinski, and M. Größer. Probmela and verification of markov decision processes. *SIGMETRICS Performance Evaluation Review*, 32(4):22–27, 2005.
3. C. Baier and H. Hermanns. Weak Bisimulation for Fully Probabilistic Processes. In *CAV*, pages 119–130, 1997.
4. C. Baier, J.-P. Katoen, H. Hermanns, and V. Wolf. Comparative branching-time semantics for Markov chains. *Inf. Comput.*, 200(2):149–214, 2005.
5. Bianco and de Alfaro. Model Checking of Probabilistic and Nondeterministic Systems. *FSTTCS*, 15, 1995.
6. J. A. Brzozowski and E. Mccluskey. Signal Flow Graph Techniques for Sequential Circuit State Diagrams. *IEEE Trans. on Electronic Computers*, EC-12:67–76, 1963.
7. B. Damman, T. Han, and J.-P. Katoen. Regular Expressions for PCTL Counterexamples. In *QEST*, 2008. to appear.
8. C. Daws. Symbolic and Parametric Model Checking of Discrete-Time Markov Chains. In *ICTAC*, pages 280–294, 2004.
9. S. Derisavi, H. Hermanns, and W. Sanders. Optimal State-Space Lumping in Markov Chains. *Inf. Process. Lett.*, 87(6):309–315, 2003.
10. K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for computer algebra*. Kluwer Academic Publishers, 1992.
11. H. Gruber and J. Johannsen. Optimal Lower Bounds on Regular Expression Size Using Communication Complexity. In *FoSSaCS*, pages 273–286, 2008.
12. E. M. Hahn, H. Hermanns, and L. Zhang. Probabilistic reachability for parametric markov models. Reports of SFB/TR 14 AVACS 50, SFB/TR 14 AVACS, 2009.
13. T. Han, J.-P. Katoen, and A. Mereacre. Approximate Parameter Synthesis for Probabilistic Time-Bounded Reachability. In *RTSS*, pages 173–182, 2008.
14. H. Hansson and B. Jonsson. A Logic for Reasoning about Time and Reliability. *FAC*, 6(5):512–535, 1994.
15. A. Hinton, M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM: A Tool for Automatic Verification of Probabilistic Systems. In *TACAS*, pages 441–444, 2006.
16. J. E. Hopcroft, R. Motwani, and J. D. Ullman. Introduction to automata theory, languages, and computation, 2nd edition. *SIGACT News*, 32(1):60–65, 2001.
17. O. Ibe and K. Trivedi. Stochastic Petri Net Models of Polling Systems. *IEEE Journal on Selected Areas in Communications*, 8(9):1649–1657, 1990.
18. B. Jonsson and K. G. Larsen. Specification and Refinement of Probabilistic Processes. In *LICS*, pages 266–277. IEEE Computer Society, 1991.
19. M. Z. Kwiatkowska, G. Norman, and D. Parker. Stochastic Model Checking. In *SFM*, pages 220–270, 2007.
20. R. Lanotte, A. Maggiolo-Schettini, and A. Troina. Parametric probabilistic transition systems for system design and analysis. *FAC*, 19(1):93–109, 2007.
21. A. Pnueli and L. Zuck. Verification of multiprocess probabilistic protocols. *Distrib. Comput.*, 1(1):53–72, 1986.
22. M. K. Reiter and A. D. Rubin. Crowds: anonymity for Web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, 1998.
23. K. Sen, M. Viswanathan, and G. Agha. Model-checking markov chains in the presence of uncertainties. In *TACAS*, pages 394–410, 2006.
24. W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.
25. R. Wimmer, S. Derisavi, and H. Hermanns. Symbolic partition refinement with dynamic balancing of time and space. In *QEST*, pages 65–74, 2008.

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 $\mathit{Path}^{\mathcal{D}}(s_0, \mathbf{B}) = \{\sigma \in \mathit{Path}^{\mathcal{D}} \mid \mathit{first}(\sigma) = s_0 \wedge \mathit{last}(\sigma) \in \mathbf{B}\}$ denote the set of paths reaching $\mathbf{B}$. The probability of reaching $\mathbf{B}$ can be then expressed by the sum

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

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

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

Now it is sufficient to show that

$$Pr^{(\mathcal{D}_1)_u}(\mathbf{B}) = Pr^{(\mathcal{D}_2)_u}(\mathbf{B}) \tag{2}$$

Let $\sigma \in \mathit{Path}^{\mathcal{D}_1}(s_0, \mathbf{B})$ be an arbitrary path in $\mathcal{D}_1$. Then, we let $\mathit{ind}(\sigma) \in \mathit{Path}^{\mathcal{D}_2}(s_0, \mathbf{B})$ denote the induced path in $\mathcal{D}_2$, which is obtained by eliminating each occurrence of s in σ. Let $\sigma' \in \mathit{Path}^{\mathcal{D}_2}(s_0, \mathbf{B})$ be an arbitrary path in $\mathcal{D}_2$. The path σ' corresponds to a set of paths $\mathit{pre}(\sigma')$ in $\mathit{Path}^{\mathcal{D}_1}(s_0, \mathbf{B})$ defined as follows: $\mathit{pre}(\sigma') = \{\sigma \in \mathit{Path}^{\mathcal{D}_1}(s_0, \mathbf{B}) \mid \mathit{ind}(\sigma) = \sigma'\}$. The relation of σ' and $\mathit{pre}(\sigma')$ is illustrated in Fig. 5. Obviously, it holds that:

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

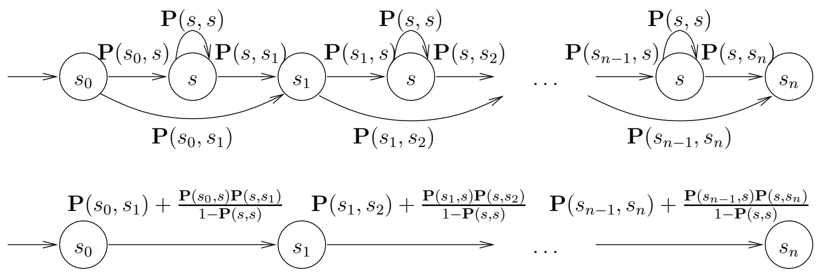
Now to show Equation 2, it is sufficient to show that for each $\sigma' \in \mathit{Path}^{\mathcal{D}_2}(s_0, \mathbf{B})$, it holds that:

$$Pr^{(\mathcal{D}_1)_u}(\mathit{pre}(\sigma')) = Pr^{(\mathcal{D}_2)_u}(\sigma') \tag{3}$$

Without loss of generality, let $\sigma' = s_0, s_1, \ldots, s_n$ with $s_n \in \mathbf{B}$. Before we show Equation 3, we first introduce some notations. For $j = 0, \ldots, n-1$ and $i_j \in \mathbb{N}$, we define $f(s_j, j, s_{j+1})$ by:

$$f(s_j, i_j, s_{j+1}) = \begin{cases} \mathbf{P}_1(s_j, s_{j+1}) & \text{if } i_j = 0 \\ \mathbf{P}_1(s_j, s)\mathbf{P}_1(s, s)^{i_j - 1}\mathbf{P}_1(s, s_{j+1}) & \text{if } i_j > 0 \end{cases} \tag{4}$$

For the definition 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 $s_0 s^{i_0} s_1 s^{i_1} \cdots s^{i_{n-1}} s_n$ has either the form



$\mathbf{P}(s,s)$
$\mathbf{P}(s_0,s)$
$\mathbf{P}(s,s_1)$
$\mathbf{P}(s_1,s)$
$\mathbf{P}(s,s_2)$
$\mathbf{P}(s_{n-1},s)$
$\mathbf{P}(s,s_n)$
s_0
s
s_1
s_n
$\cdots$
$\mathbf{P}(s_0,s_1)$
$\mathbf{P}(s_1,s_2)$
$\mathbf{P}(s_{n-1},s_n)$
$\mathbf{P}(s_0,s_1)+\frac{\mathbf{P}(s_0,s)\mathbf{P}(s,s_1)}{1-\mathbf{P}(s,s)}$
$\mathbf{P}(s_1,s_2)+\frac{\mathbf{P}(s_1,s)\mathbf{P}(s,s_2)}{1-\mathbf{P}(s,s)}$
$\mathbf{P}(s_{n-1},s_n)+\frac{\mathbf{P}(s_{n-1},s)\mathbf{P}(s,s_n)}{1-\mathbf{P}(s,s)}$

Fig. 5. Illustration of state elimination proof

1. $s_j s_{j+1}$, that is, no s occurs in between s_j and s_{j+1}, or
2. $s_j s^{i_j-1} s_{j+1}$ where $i_j > 0$ that is, there are one or several s in between.

For (1), 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. For (2), 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 a probability of $\mathbf{P}_1(s_j, s)\mathbf{P}_1(s,s)^{i_j-1}\mathbf{P}_1(s, s_{j+1})$. If s_j, s_{j+1} is clear from the context, we write simply $f(i_j)$. Thus, it holds:

$$\sum_{i_j=0}^{\infty} f(i_j) = \mathbf{P}_1(s_j, s_{j+1}) + \frac{\mathbf{P}_1(s_j, s)\mathbf{P}_1(s, s_{j+1})}{1 - \mathbf{P}_1(s,s)} = \mathbf{P}_2(s_j, s_{j+1}) \tag{5}$$

Now we show Equation 3:

$$Pr^{(\mathcal{D}_1)_u}(pre(\sigma')) = \sum_{i_0=0}^{\infty}\sum_{i_1=0}^{\infty}\cdots\sum_{i_{n-1}=0}^{\infty} Pr^{(\mathcal{D}_1)_u}\left(s_0 s^{i_0} s_1 s^{i_1}\cdots s^{i_{n-1}} s_n\right)$$

$$= \sum_{i_0=0}^{\infty}\sum_{i_1=0}^{\infty}\cdots\sum_{i_{n-1}=0}^{\infty}\left(\prod_{j=0}^{n-1} f(i_j)\right) = \sum_{i_0=0}^{\infty}\sum_{i_1=0}^{\infty}\cdots\sum_{i_{n-2}=0}^{\infty}\prod_{j=0}^{n-2} f(i_j)\left(\sum_{i_{n-1}=0}^{\infty} f(i_{n-1})\right)$$

$$= \left(\sum_{i_0=0}^{\infty} f(i_0)\right)\left(\sum_{i_1=0}^{\infty} f(i_1)\right)\cdots\left(\sum_{i_{n-1}=0}^{\infty} f(i_{n-1})\right) \overset{(5)}{=} \prod_{i=0}^{n-1}\mathbf{P}_2(s_i, s_{i+1}) = Pr^{(\mathcal{D}_2)_u}(\sigma')$$

Since u is maximal well-defined, $1 - \mathbf{P}_1(s,s) \neq 0$, implying Equation 3. 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 $\mathit{reach}(s_0, s)$ and $\mathit{reach}(s, \mathbf{B})$. 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 a evaluation function u. We let $\mathcal{R}_1 = (\mathcal{D}_1, r_1)$, $\mathcal{D}_1 = (S_1, s_0, \mathbf{P}_1)$ denote the PMC before eliminating of the state $s \in S \setminus \mathbf{B} \cup \{s_0\}$, and let $\mathcal{R}_2 = (\mathcal{D}_2, r_2)$, $\mathcal{D}_2 = (S_2, s_0, \mathbf{P}_2)$ denote the PMC after eliminating of the state s. As for PMCs, it holds that $S_2 = S_1 \setminus \{s\}$, and the matrix $\mathbf{P}_2$ is as defined in Equation 1. For $s, s' \in S_1$, we let $r_1^*(s, s')$ denote the reward $r_1(s) + r_1(s, s')$. Similarly, for $s, s' \in S_2$, we let $r_r^*(s, s')$ denote the reward $r_1(s) + r_1(s, s')$. We now discuss how the reward function r_2 is obtained in the 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

$$p_e(s_1, s_2) := \frac{\mathbf{P}_1(s_1, s)\mathbf{P}_1(s, s_2)}{1 - \mathbf{P}_1(s, s)} \tag{6}$$

$$r_e(s_1, s_2) := r_1(s_1, s) + r_1^*(s, s_2) + \frac{\mathbf{P}_1(s, s)}{1 - \mathbf{P}_1(s, s)} r_1^*(s, s) \tag{7}$$

$$r_2(s_1, s_2) := \frac{p_e(s_1, s_2) r_e(s_1, s_2) + \mathbf{P}_1(s_1, s_2) r_1(s_1, s_2)}{p_e(s_1, s_2) + \mathbf{P}_1(s_1, s_2)} \tag{8}$$

$$\mathbf{P}_2(s_1, s_2) = p_e(s_1, s_2) + \mathbf{P}_1(s_1, s_2) \tag{9}$$

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

$$R^{(\mathcal{R}_1)_u}(s_0, \mathbf{B}) = R^{(\mathcal{R}_2)_u}(s_0, \mathbf{B}) \tag{10}$$

Now with the notation of the proof of lemma 1 to show Equation 10, it is sufficient to show that for each $\sigma' \in \mathit{Path}^{\mathcal{D}_2}(s_0, \mathbf{B})$, it holds that:

$$R^{(\mathcal{R}_1)_u}(\mathit{pre}(\sigma')) = R^{(\mathcal{R}_2)_u}(\sigma') \tag{11}$$

where $R(\sigma) = Pr(\sigma)X(\sigma)$ and $R(C) = \sum_{\sigma \in C} R(\sigma)$. Without loss of generality, let $\sigma' = s_0, s_1, \ldots, s_n$ with $s_n \in \mathbf{B}$. Before we show Equation 11, we introduce some notations. For $j = 0, \ldots, n-1$ and $i_j \in \mathbb{N}$, $f(s_j, j, s_{j+1})$ is as defined in Equation 4. Moreover, $g(s_j, i_j, s_{j+1})$ is defined by:

$$g(s_j, i_j, s_{j+1}) = \begin{cases} r_1^*(s_j, s_{j+1}) & \text{if } i_j = 0 \\ r_1^*(s_j, s) + r_1^*(s, s_{j+1}) + (i_j - 1)(r_1^*(s, s)) & \text{if } i_j > 0 \end{cases} \tag{12}$$

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 11:

$$\begin{aligned}
&R^{(\mathcal{D}_1)_u}(pre(\sigma')) \\
&= \sum_{i_0=0}^{\infty}\sum_{i_1=0}^{\infty}\cdots\sum_{i_{n-1}=0}^{\infty} Pr^{(\mathcal{D}_1)_u}\left(s_0 s^{i_0} s_1 s^{i_1}\cdots s^{i_{n-1}} s_n\right) X\left(s_0 s^{i_0} s_1 s^{i_1}\cdots s^{i_{n-1}} s_n\right) \\
&= \sum_{i_0=0}^{\infty}\sum_{i_1=0}^{\infty}\cdots\sum_{i_{n-1}=0}^{\infty}\left(\prod_{j=0}^{n-1} f(i_j)\sum_{k=0}^{n-1} g(i_k)\right) = \sum_{k=0}^{n-1}\left(\sum_{i_0=0}^{\infty}\sum_{i_1=0}^{\infty}\cdots\sum_{i_{n-1}=0}^{\infty}\prod_{j=0}^{n-1} f(i_j) g(i_k)\right) \\
&= \sum_{k=0}^{n-1}\left(\sum_{i_0=0}^{\infty} f(i_0)\right)\cdots\left(\sum_{i_{k-1}=0}^{\infty} f(i_{k-1})\right)\left(\sum_{i_k=0}^{\infty} f(i_k) g(i_k)\right) \\
&\qquad\qquad\qquad\left(\sum_{i_{k+1}=0}^{\infty} f(i_{k+1})\right)\cdots\left(\sum_{i_{n-1}=0}^{\infty} f(i_{n-1})\right) \\
&\overset{(5)}{=} \sum_{k=0}^{n-1}\left(\prod_{i=0}^{n-1}\mathbf{P}_2(s_i, s_{i+1})\cdot\frac{\sum_{i_k=0}^{\infty} f(i_k) g(i_k)}{\mathbf{P}_2(s_k, s_{k+1})}\right) \\
&= \left(\prod_{i=0}^{n-1}\mathbf{P}_2(s_i, s_{i+1})\right)\left(\sum_{k=0}^{n-1}\frac{\sum_{i_k=0}^{\infty} f(i_k) g(i_k)}{\mathbf{P}_2(s_k, s_{k+1})}\right) \\
&= Pr^{(\mathcal{D}_2)_u}(\sigma')\left(\sum_{k=0}^{n-1}\frac{\sum_{i_k=0}^{\infty} f(i_k) g(i_k)}{\mathbf{P}_2(s_k, s_{k+1})}\right)
\end{aligned}$$

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

$$\sum_{i_k=0}^{\infty} f(i_k) g(i_k) = \mathbf{P}_2(s_k, s_{k+1}) r_2^*(s_k, s_{k+1}) \tag{13}$$

Taking the term for $i_k = 0$ to the right side, it is equivalent to show that:

$$\sum_{i_k=1}^{\infty} f(i_k) g(i_k) = \mathbf{P}_2(s_k, s_{k+1}) r_2^*(s_k, s_{k+1}) - \mathbf{P}_1(s_k, s_{k+1}) r_1^*(s_k, s_{k+1}) \tag{14}$$

According to Equation 12, $\sum_{i_k=1}^{\infty} f(i_k) g(i_k)$ equals to:

$$\left((r_1^*(s_k, s) + r_1^*(s, s_{k+1}))\sum_{i_k=1}^{\infty} f(i_k)\right) + \left(r_1^*(s, s)\sum_{i_k=1}^{\infty} f(i_k)(i_k - 1)\right) \tag{15}$$

By Equation 5 and Equation 6, it holds that:

$$\sum_{i_k=1}^{\infty} f(i_k) = p_e(s_k, s_{k+1}) \tag{16}$$

For $0 \leq p_c < 1$, 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 15:

$$\sum_{i_k=1}^{\infty} f(i_k)(i_k - 1) = p_e(s_k, s_{k+1}) \frac{\mathbf{P}_1(s,s)}{1 - \mathbf{P}_1(s,s)} \tag{17}$$

Now putting Equations 15,16,17 together with Equation 7, we have:

$$\begin{aligned}
&\sum_{i_k=1}^{\infty} f(i_k) g(i_k) = (r_e(s_k, s_{k+1}) + r_1(s_k)) p_e(s_k, s_{k+1}) \\
&\overset{(8)}{=} \mathbf{P}_2(s_k, s_{k+1}) r_2(s_k, s_{k+1}) - \mathbf{P}_1(s_k, s_{k+1}) r_1(s_k, s_{k+1}) + r_1(s_k) p_e(s_k, s_{k+1}) \\
&\overset{(9)}{=} \mathbf{P}_2(s_k, s_{k+1}) r_2^*(s_k, s_{k+1}) - \mathbf{P}_1(s_k, s_{k+1}) r_1^*(s_k, s_{k+1})
\end{aligned}$$

which proves Equation 14. This means that the expected accumulated reward till $\mathbf{B}$ is reached is equal in the old and the new model. After the execution of lines 2-2 the remaining paths with non-zero probabilities from the initial states to $\mathbf{B}$ all have a length of 1. Because of this, in line 2 the expected reward can be obtained directly from the probability matrix $\mathbf{P}$ and reward matrix r. As before, let $r^*(s,s')$ denote $r(s) + r(s,s')$. Then,

$$\begin{aligned}
&R^{(D_2)_u}(s_0, \mathbf{B}) \\
&= \sum_{s \in \mathbf{B}} \sum_{i=0}^{\infty} \mathbf{P}(s_0, s_0)^i \mathbf{P}(s_0, s) \left(r^*(s_0, s) + i \cdot r^*(s_0, s_0)\right) \\
&= \sum_{s \in \mathbf{B}} \left(\mathbf{P}(s_0, s) r^*(s_0, s) \sum_{i=0}^{\infty} \mathbf{P}(s_0, s_0)^i + \mathbf{P}(s_0, s) r^*(s_0, s_0) \sum_{i=0}^{\infty} i \cdot \mathbf{P}(s_0, s_0)^i \right) \\
&= \sum_{s \in \mathbf{B}} \left(\frac{\mathbf{P}(s_0, s) r^*(s_0, s)}{1 - \mathbf{P}(s_0, s_0)} + \frac{\mathbf{P}(s_0, s) r^*(s_0, s_0) \mathbf{P}(s_0, s_0)}{\left(1 - \mathbf{P}(s_0, s_0)\right)^2} \right)
\end{aligned}$$

The proof to show that no divisions by zero occur is analog to the one in A.1

A.3 Proof of lemma 3

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

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

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

A.4 Proof of lemma 4

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

$$Pr^{\mathcal{M}_u, A}(s_0, \mathbf{B}) = \max_{A' \in \mathrm{MD}(\mathcal{M})} Pr^{\mathcal{M}_u, A'}(s_0, \mathbf{B}) \tag{18}$$

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

$$Pr^{\mathcal{M}_u, A}(s, \mathbf{B}) = \max_{A' \in \mathrm{MD}(\mathcal{M})} Pr^{\mathcal{M}_u, A'}(s, \mathbf{B}) \tag{19}$$

for all $s \in S$ [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), we only have to show that $reach^{(\mathrm{enc}(\mathcal{M}))_w}(s, \mathbf{B})$ for $s \in S' = \left\{ s \in S \mid reach^{\mathrm{enc}(\mathcal{M})}(s, \mathbf{B}) \right\}$, because states not in S' will be removed by the preprocessing of the state-elimination algorithm. Because u is strictly well-defined, it is $reach^{(\mathrm{enc}(\mathcal{M}))_u}(s, \mathbf{B})$ if $s \in S'$. That means that there is a v' with $Pr^{((\mathrm{enc}(\mathcal{M}))_u)_{v'}}(s, \mathbf{B}) > 0$. This means that there is a scheduler A' with $Pr^{(\mathcal{M}_{A'})_u}(s, \mathbf{B}) > 0$. Because of 19, it follows that $Pr^{(\mathcal{M}_A)_u}(s, \mathbf{B}) > 0$. Due to the definition of v, this also means $Pr^{((\mathrm{enc}(\mathcal{M}))_u)_v}(s, \mathbf{B}) > 0$ and in turn $Pr^{(\mathrm{enc}(\mathcal{M}))_w}(s, \mathbf{B}) > 0$, which is equivalent to $reach^{(\mathrm{enc}(\mathcal{M}))_w}(s, \mathbf{B})$.

Now we have:

$$\begin{aligned} f[Var_\delta/v][V/u] = f[V \dot\cup Var_\delta / w] &\overset{\text{Lem.1}}{=} Pr^{\mathrm{enc}(\mathcal{M})_w}(s_0, \mathbf{B}) \\ &= Pr^{\left((\mathrm{enc}(\mathcal{M}))_v\right)_u}(s_0, \mathbf{B}) \overset{\text{Lem.3}}{=} Pr^{M_u, A}(s_0, \mathbf{B}) \\ &\overset{(18)}{=} \max_{A' \in \mathrm{MD}(\mathcal{M})} Pr^{\mathcal{M}_u, A'}(s_0, \mathbf{B}) \end{aligned}$$

A.5 Proof of lemma 5

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.

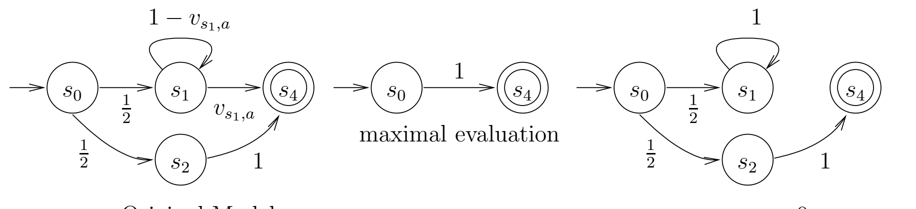
Proof. If $s_1 \sim s_2$ then there exists a strong bisimulation R with $s_1 R s_2$. Obviously R is also a bisimulation in $\mathcal{D}_u$: for $s_1' R s_2'$ and $C \in S/R$, we have: $\mathbf{P}(s_1', C) = \mathbf{P}(s_2', C)$ implies $\mathbf{P}_u(s_1', C) = \mathbf{P}_u(s_2', C)$.

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.

Proof. 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 bisimulation $\mathcal{D}_u$. 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 $\mathcal{G}_{\mathcal{D}_u}$.



Original Model $\qquad$ $v_{s_1,a} = 0$

In the figure above, 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 valuation 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 $u'(v_{s_1,a})$. Obviously u' is not maximal well-defined, as state s_1 can not reach $\mathbf{B}$. For this evaluation no states are weak bisimilar, thus the quotient is the same as the original automaton (depicted on the right).