Accelerating Parametric Probabilistic Verification ⋆

Nils Jansen[1], Florian Corzilius[1], Matthias Volk[1], Ralf Wimmer[2], Erika Ábrahám[1], Joost-Pieter Katoen[1], and Bernd Becker[2]

[1] RWTH Aachen University, Germany
{nils.jansen | corzilius | volk | abraham | katoen}@cs.rwth-aachen.de
[2] Albert-Ludwigs-University Freiburg, Germany
{wimmer | becker}@informatik.uni-freiburg.de

Abstract. We present a novel method for computing reachability probabilities of parametric discrete-time Markov chains whose transition probabilities are fractions of polynomials over a set of parameters. Our algorithm is based on two key ingredients: a graph decomposition into strongly connected subgraphs combined with a novel factorization strategy for polynomials. Experimental evaluations show that combining these approaches leads to a speed-up of up to several orders of magnitude in comparison to existing approaches.

1 Introduction

Discrete-time Markov chains (DTMCs) are a widely used modeling formalism for systems exhibiting probabilistic behavior. Their applicability ranges from distributed computing to security and systems biology. Efficient algorithms exist to compute measures like: "What is the probability that our communication protocol terminates successfully if messages are lost with probability 0.05?". However, often actual system parameters like costs, faultiness, reliability and so on are not given explicitly. For the design of systems incorporating random behavior, this might even not be possible at an early design stage. In model-based performance analysis, the research field of *fitting* [1], where—intuitively—probability distributions are generated from experimental measurements, mirrors the difficulties in obtaining such concrete values.

This calls for treating probabilities as parameters and motivates to consider *parametric* DTMCs, PDTMCs for short, where transition probabilities are (rational) functions in terms of the system's parameters. Using these functions one can, e. g., find appropriate values of the parameters such that certain properties are satisfied or analyze the sensitivity of reachability probabilities to small changes in the parameters. Computing reachability probabilities for standard DTMCs is

⋆ This work was partly supported by the German Research Council (DFG) as part of the research project CEBug (AB 461/1-1), the Research Training Group AlgoSyn (1298), the EU FP7-project MoVeS, the FP7-IRSES project MEALS and by the Excellence Initiative of the German federal and state government.

typically done by solving a linear equation system using iterative methods. This is not feasible for PDTMCs. Instead, approaches based on state elimination have been proposed [2,3]. The idea is to replace a state and its incident transitions with direct transitions from its predecessor to its successor states. In this way, one eliminates all states except for the initial and target states of a system. The result is a rational function describing the probability of reaching a set of target states, depending on the values of the parameters. The efficiency of such elimination methods strongly depends on the order in which states are eliminated and on the representation of the rational functions.

Related work. The idea of constructing a regular expression representing the DTMC originates from Daws [2]. He uses state elimination to generate regular expressions describing the paths to the target states of the system. Hahn *et al.* [3] apply this idea to PDTMCs to obtain rational functions for reachability and expected reward properties. They improve the efficiency of the construction by heuristics for the transformation of finite automata to regular expressions [4] to guide the elimination of states. Additionally, they reduce the polynomials to simplify the rational functions. These ideas have been extended to Markov decision processes [5]. The main problem there is that the reachability probabilities depend on the chosen resolution of the nondeterminism. When maximizing or minimizing these probabilities, the optimal resolution generally depends on the values of the parameters. Their algorithms are implemented in PARAM [6], the—to the best of our knowledge—only available tool for computing reachability probabilities of PDTMCs. This paper can be seen as a continuation of [2,3]. Several authors have considered the related problem of parameter synthesis: for which parameter instances does a given (LTL or PCTL) formula hold? To mention a few, Han *et al.* [7] considered this problem for timed reachability in continuous-time Markov chains, Pugelli *et al.* [8] for Markov decision processes (MDPs), and Benedikt *et al.* [9] for ω-regular properties of interval Markov chains.

Contributions of the paper. In this paper we improve the computation of reachability probabilities for PDTMCs [2,3] in two important ways. We consider a state elimination strategy based on a recursive graph decomposition of the PDTMC into strongly connected subgraphs and give a novel method to efficiently factorize polynomials. Although presented in the context of parametric Markov chains, this constitutes a generic method for representing and manipulating rational functions and is also suited for other applications as well. The experiments show that the combination of both techniques yields a speed-up of more than one order of magnitude compared to [3].

2 Preliminaries

Definition 1 (Discrete-time Markov chain). *A* discrete-time Markov chain (DTMC) *is a tuple $\mathcal{D} = (S, I, P)$ with a non-empty finite set S of states, an initial distribution $I : S \to [0,1] \subseteq \mathbb{R}$ with $\sum_{s \in S} I(s) = 1$, and a transition probability matrix $P : S \times S \to [0,1] \subseteq \mathbb{R}$ with $\sum_{s' \in S} P(s,s') = 1$ for all $s \in S$.*

The states $S_I = \{s_I \in S \mid I(s_I) > 0\}$ are called *initial states*. A *transition* leads from a state $s \in S$ to a state $s' \in S$ iff $P(s, s') > 0$. The set of *successor states of* $s \in S$ is $succ(s) = \{s' \in S \mid P(s, s') > 0\}$. A *path* of $\mathcal{D}$ is a finite sequence $\pi = s_0 s_1 \dots s_n$ of states $s_i \in S$ such that $P(s_i, s_{i+1}) > 0$ for all $i < n$. $Paths^{\mathcal{D}}$ is the set of all paths of $\mathcal{D}$. We denote the set of paths starting at $s \in S$ by $Paths^{\mathcal{D}}(s)$. $Paths^{\mathcal{D}}(s, t)$ denotes the set of paths starting at s and ending at t. We generalize this to sets $S', S'' \subseteq S$ of states by $Paths^{\mathcal{D}}(S', S'') = \bigcup_{s' \in S'} \bigcup_{s'' \in S''} Paths^{\mathcal{D}}(s', s'')$. A state t is *reachable* from s iff $Paths^{\mathcal{D}}(s, t) \neq \emptyset$.

The *probability measure* $Pr^{\mathcal{D}}_{fin}(\pi)$ for paths $\pi \in Paths^{\mathcal{D}}$ is given by $Pr^{\mathcal{D}}_{fin}\big((\pi = s_0 \dots s_n)\big) = I(s_0) \cdot \prod_{i=0}^{n-1} P(s_i, s_{i+1})$. Note that for two paths $\pi_1, \pi_2 \in Paths^{\mathcal{D}}$ it holds that $Pr^{\mathcal{D}}_{fin}(\{\pi_1, \pi_2\}) = Pr^{\mathcal{D}}_{fin}(\pi_1) + Pr^{\mathcal{D}}_{fin}(\pi_2)$ if no path is a prefix of the other one, i. e., if they are *stochastically independent*. For a set $R \subseteq Paths^{\mathcal{D}}$ we define $Pr^{\mathcal{D}}_{fin}(R) = \sum_{\pi \in R'} Pr^{\mathcal{D}}_{fin}(\pi)$ with $R' = \{\pi \in R \mid \forall \pi' \in R.\ \pi' \text{ is not a proper prefix of } \pi\}$. For more details we refer, e. g., to [10].

For a DTMC $\mathcal{D} = (S, I, P)$ and a subset of states $K \subseteq S$ we define the set of *input states* of K by $Inp(K) = \{s \in K \mid I(s) > 0 \vee \exists s' \in S \setminus K.\ P(s', s) > 0\}$, i. e., the states inside K that have an incoming transition from outside K. Analogously, we define the set of *output states* of K by $Out(K) = \{s \in S \setminus K \mid \exists s' \in K.\ P(s', s) > 0\}$, i. e., the states outside K that have an incoming transition from a state inside K. The set of *inner states* of K is given by $K \setminus Inp(K)$.

We call a state set $S' \subseteq S$ *absorbing* iff there is a state $s' \in S'$ from which no state outside S' is reachable in $\mathcal{D}$, i. e., iff $Paths^{\mathcal{D}}(\{s'\}, S \setminus S') = \emptyset$. A state $s \in S$ is absorbing if $\{s\}$ is absorbing.

A set $S' \subseteq S$ induces a *strongly connected subgraph (SCS) of* $\mathcal{D}$ iff for all $s, t \in S'$ there is a path from s to t visiting only states from S'. A *strongly connected component (SCC)* of $\mathcal{D}$ is a maximal (w. r. t. $\subseteq$) SCS of S. If $Out(S') = \emptyset$ holds for an SCC S', S' is called a *bottom SCC*. The probability of eventually reaching a bottom SCC in a finite DTMC is always 1 [10, Chap. 10.1].

We only consider *probabilistic reachability properties*, i. e., the probability to eventually reach a set $T \subseteq S$ of target states, formally: $Pr^{\mathcal{D}}_{fin}\big(Paths^{\mathcal{D}}(S_I, T)\big)$. It is well-known that this suffices for checking arbitrary ω-regular properties, see [10, Chap. 10.3] for the details.

2.1 Parametric Markov Chains

To add parameters to DTMCs, we follow [6] by allowing arbitrary rational functions defining probability distributions.

Definition 2 (Polynomial and rational function). *Let* $V = \{x_1, \dots, x_n\}$ *be a finite set of* variables *with domain* $\mathbb{R}$. *A* polynomial g *over* V *is a sum of* monomials*, which are products of variables in* V *and a coefficient in* $\mathbb{Z}$*:*

$$g = a_1 \cdot x_1^{e_{1,1}} \cdot \ldots \cdot x_n^{e_{1,n}} \ + \ \cdots \ + \ a_m \cdot x_1^{e_{m,1}} \cdot \ldots \cdot x_n^{e_{m,n}},$$

where $e_{i,j} \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$ *and* $a_i \in \mathbb{Z}$ *for all* $1 \leq i \leq m$ *and* $1 \leq j \leq n$. $\mathbb{Z}[x_1, \dots, x_n]$ *denotes the set of polynomials over* $V = \{x_1, \dots, x_n\}$. *A* rational

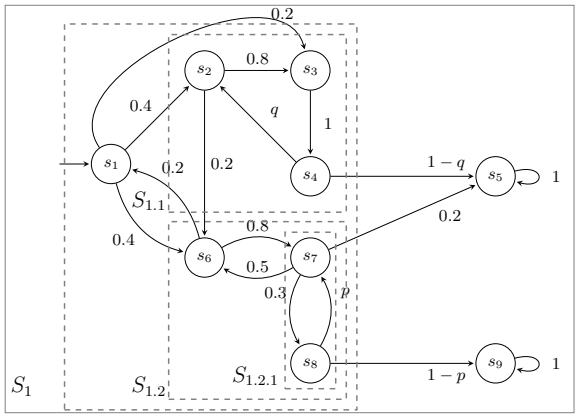


Fig. 1. Example PDTMC and its SCC decomposition

function *over V is a quotient $f = \frac{g_1}{g_2}$ of two polynomials g_1, g_2 over V with $g_2 \neq 0$*[3]*. We use $\mathcal{F}_V = \left\{ \frac{g_1}{g_2} \mid g_1, g_2 \in \mathbb{Z}[x_1, \ldots, x_n] \wedge g_2 \neq 0 \right\}$ to denote the set of rational functions over V.*

Definition 3 (PDTMC). *A* parametric discrete-time Markov chain (PDTMC) *is a tuple $\mathcal{M} = (S, V, I, P)$ with a finite set of states S, a finite set of parameters $V = \{x_1, \ldots, x_n\}$ with domain $\mathbb{R}$, an initial distribution $I : S \to \mathcal{F}_V$, and a parametric transition probability matrix $P : S \times S \to \mathcal{F}_V$.*

As we are applying graph-based algorithms, we need the *underlying graph of a (P)DTMC $\mathcal{M}$*, which is given by $\mathcal{G}_\mathcal{M} = (S, \mathcal{D}_P)$ where $\mathcal{D}_P = \left\{ (s, s') \in S \times S \mid P(s, s') \neq 0 \right\}$. Using an *evaluation*, all or some of the parameters occurring in the rational functions of a PDTMC can be instantiated.

Definition 4 (Evaluated PDTMC). *An* evaluation u of V *is a function $u \colon V \to \mathbb{R}$. The evaluation $g[u]$ of a polynomial $g \in \mathbb{Z}[x_1, \ldots, x_n]$ under $u \colon V \to \mathbb{R}$ substitutes each $x \in V$ by $u(x)$, using the standard semantics for $+$ and $\cdot$. For $f = \frac{g_1}{g_2} \in \mathcal{F}_V$ we define $f[u] = \frac{g_1[u]}{g_2[u]} \in \mathbb{R}$ if $g_2[u] \neq 0$.*

For a PDTMC $\mathcal{M} = (S, V, I, P)$, the evaluated PDTMC *is the DTMC $\mathcal{D} = (S_u, I_u, P_u)$ given by $S_u = S$ and for all $s, s' \in S_u$, $P_u(s, s') = P(s, s')[u]$ and $I_u(s) = I(s)[u]$ if the evaluations are defined and 0 otherwise.*

An evaluation u substitutes each parameter by a real number. This induces a probability measure on the evaluated PDTMC under the following conditions.

Definition 5 (Well-defined evaluation). *An evaluation u is* well-defined *for PDTMC $\mathcal{M} = (S, V, I, P)$ if for the evaluated PDTMC $\mathcal{D} = (S_u, I_u, P_u)$ it holds that*

- *$P_u : S_u \times S_u \to [0, 1]$ with $\forall s \in S_u : \sum_{s' \in S_u} P_u(s, s') = 1$, and*

[3] $g_2 \neq 0$ means that g_2 cannot be simplified to 0.

– $I_u : S_u \to [0,1]$ *with* $\sum_{s \in S_u} I_u(s) = 1$.

A well-defined evaluation u is called graph preserving*, if it holds that*

$$\forall s, s' \in S : P(s,s') \neq 0 \implies P(s,s')[u] > 0.$$

Note that $P(s,s')[u] > 0$ implies that no division by 0 will occur. This will be ensured during the model checking algorithm. Evaluation u is required to be graph preserving, i. e., $\mathcal{G}_{\mathcal{M}} = \mathcal{G}_{\mathcal{M}_u}$. This is necessary as by altering the graph, states might become unreachable which can change reachability probabilities.

Definition 6. *Given a PDTMC $\mathcal{M} = (S, V, I, P)$ and a set of absorbing target states $T \subseteq S$, the* parametric probabilistic model checking problem *is to find for each initial state $s_I \in S_I$ and each $t \in T$ a rational function $f_{s_I,t} \in \mathcal{F}_V$ such that for all graph-preserving evaluations $u : V \to \mathbb{R}$ and the evaluated PDTMC $\mathcal{D} = (S_u, I_u, P_u)$ it holds that $f_{s_I,t}[u] = Pr^{M_u}(Paths(s_I, t))$.*

3 Parametric Model Checking by SCC Decomposition

In this section we present our algorithmic approach to apply model checking to PDTMCs. In the following, we assume every PDTMC $\mathcal{M} = (S, V, I, P)$ to have only bottom SCCs consisting of one state, i. e., absorbing states, which will be the target states. For each initial state $s_I \in S_I$ and each target state $t \in T$ we compute a rational function $f_{s_I,t}$ over the set of parameters V which describes the probability of reaching t from s_I as in [3]. A similar method was introduced in [11] for the non-parametric case.

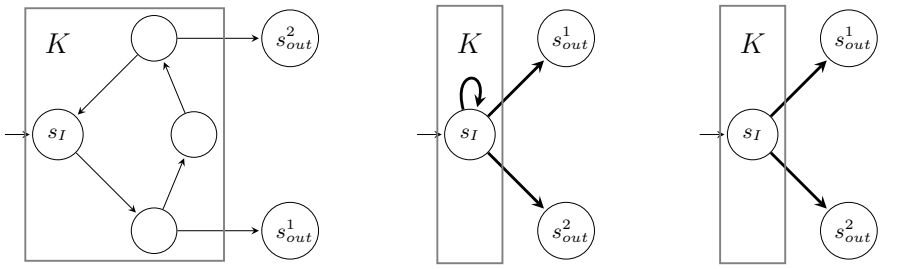
3.1 PDTMC Abstraction

The basic concept of our model checking approach is to replace a non-absorbing subset of states $K \subseteq S$ and its transitions inside a PDTMC $\mathcal{M}$ by transitions directly leading from the input states $Inp(K)$ of K to the output states $Out(K)$ of K. These transitions have the probabilities of all paths visiting only states of K. This concept is illustrated in Figure 2: In Figure 2(a), an arbitrary, non-absorbing set of states K has one input state s_I and two output states s^1_{out}, s^2_{out}. The abstraction in Figure 2(c) hides every state of K except for s_I; all transitions are directly leading to the output states.

As we need a probability measure for arbitrary subsets of states, we first define sub-PDTMCs induced by such subsets.

Definition 7 (Induced PDTMC). *Given a PDTMC $\mathcal{M} = (S, V, I, P)$ and a non-absorbing subset $K \subseteq S$ of states, the* induced PDTMC over K and $\mathcal{M}$ *is given by $\mathcal{M}^K = (S^K, I^K, V^K, P^K)$ with $S^K = K \cup Out(K)$, $V^K = V$, $\forall s \in S^K . I^K(s) \neq 0 \iff s \in Inp(K)$, and*

$$P^K(s,s') = \begin{cases} P(s,s'), & \text{if } s \in K, s' \in S^K \\ 1, & \text{if } s = s' \in Out(K) \\ 0, & \text{otherwise .} \end{cases}$$



(a) Initial PDTMC (b) Abstraction of K with abstract loop (c) Abstraction of K

Fig. 2. Concept of PDTMC abstraction

Intuitively, all incoming and outgoing transitions are preserved for inner states of K while the output states are made absorbing. We allow an arbitrary input distribution I with the only constraint that $I(s)\neq 0$ iff s is an input state of K.

Example 1. Consider the PDTMC $\mathcal{M}$ in Figure 1 and the state set $K = \{s_7, s_8\}$. The induced PDTMC $\mathcal{M}^K = (S^K, I^K, V^K, P^K)$ over K and $\mathcal{M}$ shown in Figure 3(a) has output states $Out(K) = \{s_5, s_6, s_9\}$ and input states $Inp(K) = \{s_7\}$.

For our abstraction we take into account all finite paths that do not intermediately return to the initial state. In Figure 2(b), there are abstract transitions leading to the output states together with a self-loop on the initial state. The outgoing transitions describe all paths that do not visit the input state again, while the self-loop describes all paths that return to the input state. These paths build the set of all paths that add to the probability of finally reaching one of the output states. Note that inside a non-absorbing set of states, the probability of reaching the set of all output states is 1. Figure 2(c) shows the final abstraction where the probability of the self-loop is taken into account in determining the transition probabilities of the outgoing transitions.

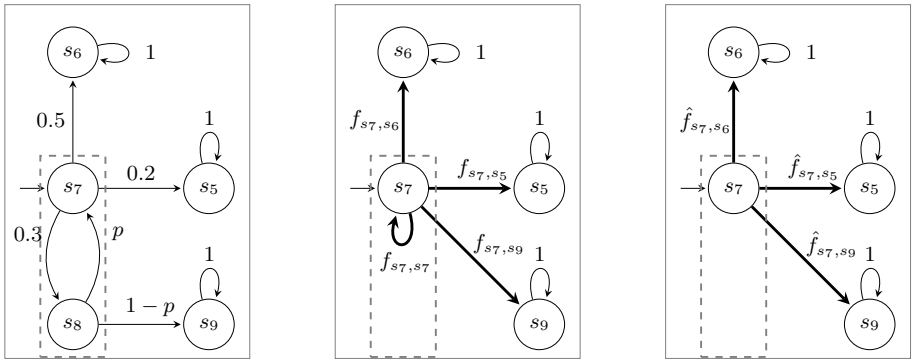
Formally, we define the probability of all finite paths that start in a state s and finally reach a state s' without returning to s beforehand. This includes paths that both start and end in s.

Definition 8. *Assume a PDTMC $\mathcal{M} = (S, V, I, P)$, a non-absorbing state $s \in S$ and a state $s' \in S$. The* path abstraction *of s and s' is given by*

$$p^{\mathcal{M}}_{abs}(s, s') = Pr^{\mathcal{M}}_{fin}\big(\{\pi = s_0 \dots s_n \in Paths^{\mathcal{M}}(s, s') \mid s_i \neq s \wedge s_i \neq s', 0 < i < n\}\big).$$

Using this we are now ready to define the abstraction of a PDTMC $\mathcal{M}$ with respect to initial states and target states. The probabilities are the total reachability probabilities between these states. Let us first consider an example.

Example 2. Consider the PDTMC $\mathcal{M}' = (S', I', P', V')$ of Figure 3(a) and let the set of target states $T' = \{s_5, s_6, s_9\}$ correspond to the absorbing states of $\mathcal{M}'$. The



(a) Induced PDTMC (b) Abstracted PDTMC (c) Scaled functions

Fig. 3. PDTMC Abstraction

abstract PDTMC $\mathcal{M}'_{abs} = (S'_{abs}, I'_{abs}, P'_{abs}, V'_{abs})$ has states $S'_{abs} = \{s_5, s_6, s_7, s_9\}$ and edges from s_7 to all other states. The first abstraction step according to the path abstraction $p^{\mathcal{M}'}_{abs}$ as in Definition 8 is depicted in Figure 3(b). The rational functions describing the probabilities of all finite paths that either leave K without visiting state s_7 again or starting and ending in s_7 are:

$$f_{s_7,s_5} = p^{\mathcal{M}'}_{abs}(s_7, s_5) = 0.2 \qquad f_{s_7,s_6} = p^{\mathcal{M}'}_{abs}(s_7, s_6) = 0.5$$
$$f_{s_7,s_7} = p^{\mathcal{M}'}_{abs}(s_7, s_7) = 0.3 \cdot p \qquad f_{s_7,s_9} = p^{\mathcal{M}'}_{abs}(s_7, s_9) = 0.3 \cdot (1-p)$$

The total probability of reaching the output states is given by paths which first use the loop on s_7 arbitrarily many times (including zero times) and then take a transition to an output state. For example, using the geometric series, the probability of the set of paths leading from s_7 to s_5 is given by

$$\sum_{i=0}^{\infty} (f_{s_7,s_7})^i \cdot f_{s_7,s_5} = \frac{1}{1 - f_{s_7,s_7}} \cdot f_{s_7,s_5}$$

As the probability of finally reaching the set of absorbing states in $\mathcal{M}'$ is 1, we can directly scale the probabilities of the outgoing edges such that their sum is equal to 1. This is achieved by dividing each outgoing probability by the sum of all outgoing probabilities, $f_{out} = 0.2 + 0.5 + 0.3 \cdot (1-p)$. The abstract and scaled PDTMC is depicted in Figure 3(c) with the probabilities given by

$$\hat{f}_{s_7,s_5} = 0.2 \,/ f_{out} \qquad \hat{f}_{s_7,s_6} = 0.5 \,/ f_{out}$$
$$\hat{f}_{s_7,s_9} = (0.3 \cdot (1-p)) \,/ f_{out}$$

We now define the final abstraction formally.

Definition 9 (Abstract PDTMC). *For a PDTMC $\mathcal{M} = (S, V, I, P)$ and a set of absorbing states $T \subseteq S$, the* abstract PDTMC $\mathcal{M}_{abs} = (S_{abs}, V_{abs}, I_{abs}, P_{abs})$

is given by $S_{abs} = \{s \in S \mid I(s) \neq 0 \vee s \in T\}$, $V_{abs} = V$, *and for all* $s, s' \in S_{abs}$ *we define* $I_{abs}(s) = I(s)$ *and*

$$P_{abs}(s,s') = \begin{cases} \frac{p^{\mathcal{M}}_{abs}(s,s')}{\sum_{s'' \in T} p^{\mathcal{M}}_{abs}(s,s'')} & \text{if } I(s) > 0 \wedge s' \in T \\ 1 & \text{if } s = s' \in T \\ 0 & \text{otherwise .} \end{cases}$$

Theorem 1. *For a PDTMC* $\mathcal{M} = (S, V, I, P)$ *and its abstraction* $\mathcal{M}_{abs} = (S_{abs}, I_{abs}, V_{abs}, P_{abs})$ *according to Definition 9 it holds for all initial states* $s_I \in S_I$ *and all absorbing states* $t \in T$ *that*

$$Pr^{\mathcal{M}}_{fin}(Paths^{\mathcal{M}}(s_I, t)) = Pr^{\mathcal{M}_{abs}}_{fin}(Paths^{\mathcal{M}_{abs}}(s_I, t)).$$

The proof of this theorem can be found in the appendix. It remains to define the substitution of subsets of states by their abstractions. Intuitively, a subset of states is replaced by the abstraction as in Definition 9, while the incoming transitions of the initial states of the abstraction as well as the outgoing transitions of the absorbing states of the abstraction are not changed.

Definition 10. *Assume a PDTMC* $\mathcal{M} = (S, V, I, P)$, *a non-absorbing set of states* $K \subseteq S$, *the induced PDTMC* $\mathcal{M}^K = (S^K, V^K, I^K, P^K)$ *and the abstraction* $\mathcal{M}^K_{abs} = (S^K_{abs}, V^K_{abs}, I^K_{abs}, P^K_{abs})$. *The* substitution of $\mathcal{M}^K$ by its abstraction $\mathcal{M}^K_{abs}$ in $\mathcal{M}$ *is given by* $\mathcal{M}_{K \mapsto abs} = (S_{K \mapsto abs}, V_{K \mapsto abs}, I_{K \mapsto abs}, P_{K \mapsto abs})$ *with* $S_{K \mapsto abs} = (S \setminus K) \cup S^K_{abs}$, $V_{K \mapsto abs} = V$ *and for all* $s, s' \in S_{K \mapsto abs}$, $I_{K \mapsto abs}(s) = I(s)$ *and*

$$P_{K \mapsto abs}(s,s') = \begin{cases} P(s,s') & \text{if } s \notin K \\ P^K_{abs}(s,s') & \text{if } s \in K \wedge s' \in Out(K) \\ 0 & \text{otherwise .} \end{cases}$$

Due to Theorem 1, it directly follows that this substitution does not change reachability properties from input states to the absorbing states of a PDTMC.

Corollary 1. *Given a PDTMC* $\mathcal{M}$ *and a non-absorbing subset* $K \subseteq S$ *of states, it holds for all initial states* $s_I \in S_I$ *and absorbing states* $t \in T$ *that*

$$Pr^{\mathcal{M}}_{fin}(Paths^{\mathcal{M}}(s_I, t)) = Pr^{\mathcal{M}_{K \mapsto abs}}_{fin}(Paths^{\mathcal{M}_{K \mapsto abs}}(s_I, t)).$$

3.2 Model Checking Parametric Markov Chains

In the previous section we gave the theoretical background for our model checking algorithm. Now we will describe how to compute the abstractions efficiently.

As a heuristic for forming the sets of states to be abstracted, we choose an SCC-based decomposition of the graph: In Figure 1, the dashed rectangles indicate the decomposition: SCC $S_1 = \{1, 2, 3, 4, 6, 7, 8\}$ and the SCSs $S_{1.1} = \{2, 3, 4\}$, $S_{1.2} = \{6, 7, 8\}$, and $S_{1.2.1} = \{7, 8\}$. Algorithmically, Tarjan's algorithm [12] is used to determine the SCC structure of the graph. Afterwards, for each SCC

Algorithm 1 Model Checking PDTMCs

abstract(PDTMC $\mathcal{M}$)
begin
  for all non-bottom SCCs K in $\mathcal{M}^{S \setminus Inp(\mathcal{M})}$ **do** (1)
    $\mathcal{M}_{abs}^{K} :=$ abstract$(\mathcal{M}^{K})$ (2)
    $\mathcal{M} := \mathcal{M}_{K \mapsto abs}$ (3)
  end for (4)
  $K := \{$*non-absorbing states in* $\mathcal{M}\}$ (5)
  $\mathcal{M} := \mathcal{M}_{K \mapsto abs}$ (6)
  return M_{abs} (7)
end

model_check(PDTMC $\mathcal{M} = (S, V, I, P), T \subseteq S, \lambda \in \mathbb{Q}$)
begin
  $\mathcal{M}_{abs} = (S_{abs}, V_{abs}, I_{abs}, P_{abs}) :=$ abstract$(\mathcal{M})$ (8)
  return $\sum_{s_I \in S_I} I(s_I) \cdot \left(\sum_{t \in T} P_{abs}(s_I, t) \right) \leq \lambda$ (9)
end

K the input states $Inp(K)$ are removed. On the resulting decomposed graph, a new search is performed, which yields a new set of SCCs which are SCSs in the original graph. This is iterated until no SCCs remain. The subset relation forms a partial order on these sets: $S_{1.1} \subset S_1$ and $S_{1.2.1} \subset S_{1.2} \subset S_1$. The smallest sets according to this partial order, $S_{1.1}$ and $S_{1.2.1}$, can only loop via their input state, otherwise there would be other included SCSs. Note that the deletion of the input states is only one possible heuristic for a decomposition of the graph.

The general model checking algorithm is depicted in Algorithm 1. The recursive method abstract(PDTMC $\mathcal{M}$) computes the abstraction $\mathcal{M}_{abs}$ by iterating over all SCCs of the graph induced by removing the input states of $\mathcal{M}$ (line 1). For each SCC K, the abstraction $\mathcal{M}_{abs}^{K}$ of the induced PDTMC $\mathcal{M}^{K}$ is computed by a recursive call of the method (line 2, Definitions 7,9). Afterwards, $\mathcal{M}^{K}$ is substituted by its abstraction inside $\mathcal{M}$ (line 3, Definition 10). Finally, the abstraction $\mathcal{M}_{abs}$ is computed and returned (line 7, Definition 9). This method is called by the model checking method (line 8) which yields the abstract system M_{abs}, in which transitions lead only from the initial states to the absorbing states. All transitions are labeled with a rational function for the reachability probability, as in Definition 6. Then the whole reachability probability is computed by building the sum of these transitions (line 9). This is compared to the given upper probability bound $\lambda \in \mathbb{Q}$ returning a truth-value. Note that this can be adapted for lower or strict probability bounds as well.

What remains to be explained is the computation of the abstract probabilities $p_{abs}^{\mathcal{M}}$. We distinguish the cases where the set K has one or multiple input states.

One input state. We define the set of paths R_{loop} going from s_I to s_I and the set of paths R_{out} going from s_I to some $t \in T$ without revisiting s_I:

$$R_{loop} = \{s_I s_1 \dots s_n s_I \in Paths^{\mathcal{M}} \mid \forall 1 \leq i \leq n.\, s_i \notin \{s_I\} \cup T\}, \tag{1}$$

$$R_{out} = \{s_I s_1 \dots s_n t \in Paths^{\mathcal{M}} \mid t \in T \wedge \forall 1 \leq i \leq n.\, s_i \notin \{s_I\} \cup T\}. \tag{2}$$

Consider a PDTMC $\mathcal{M}^K$ induced by K with one initial state s_I and the set of absorbing states $T = \{t^1, \dots, t^n\}$. We determine the probabilities $p_{abs}^{\mathcal{M}^K}(s_I, t^i)$ for all $1 \leq i \leq n$. As $K \setminus Inp(K)$ has no non-trivial SCSs, the set R_{out} of outgoing paths consists of finitely many loop-free paths. The probability is computed by the following equations for all $s \in S^K$:

$$p_{abs}^{\mathcal{M}^K}(s, t^i) = \begin{cases} 1, & \text{if } s = t^i, \\ \sum\limits_{s' \in (succ(s) \cap K) \setminus Inp(K)} P^K(s, s') \cdot p_{abs}^{\mathcal{M}^K}(s', t^i), & \text{otherwise.} \end{cases} \tag{3}$$

These probabilities can be computed by direct or indirect methods for solving linear equation systems, see, e. g., [13, Chapters 3,4]. Note that also state elimination as in [3] can be applied here.

The probabilities of the abstract PDTMC $\mathcal{M}_{abs}^K = (S_{abs}, I_{abs}, V_{abs}, P_{abs})$ as in Definition 9 can now directly be computed, while an additional constraint is added in order to avoid divisions by zero:

$$P_{abs}^{\mathcal{M}^K}(s_I, t) = \begin{cases} \frac{p_{abs}^{\mathcal{M}^K}(s_I, t)}{\sum_{t' \in T} p_{abs}^{\mathcal{M}^K}(s_I, t')}, & \text{if } \sum_{t' \in T} p_{abs}^{\mathcal{M}^K}(s_I, t') \neq 0 \\ 0, & \textit{otherwise.} \end{cases} \tag{4}$$

In case there is only one absorbing state, i. e., $n = 1$, we have $p_{abs}^{\mathcal{M}^K}(s_I, t^1) = 1$. This is directly exploited without further computations.

Multiple input states. Given a PDTMC $\mathcal{M}^K$ with the set of initial states $S_I = \{s_I^1, \dots, s_I^m\}$ with $I^K(s_I^i) > 0$ for all $1 \leq i \leq m$ and a set of absorbing states $T = \{t^1, \dots, t^n\}$. The intuitive idea would be to maintain a copy of $\mathcal{M}^K$ for each initial state and handle the other initial states as inner states in this copy. Then, the method as described in the previous paragraph can be used. However, this is both very time and memory consuming. Therefore, we first formulate the linear equation system as in Equation (3). All variables $p_{abs}^{\mathcal{M}^K}(s, s')$ with $s' \in K \setminus Inp(K)$ are eliminated from the equation system. For each of the variables $p_{abs}^{\mathcal{M}^K}(s_I, s')$, the equation system is then solved separately by eliminating all other variables.

Algorithm 1 returns the rational functions $P_{abs}^{\mathcal{M}^K}(s_I, t)$ for all $t \in T$ as in Equation (4). To allow only graph-preserving evaluations of the parameters, we perform preprocessing where conditions are collected according to Definition 5 as well as the ones from Equation (4). These constraints can be evaluated by a *SAT modulo theories* solver which can handle non-linear arithmetic over the reals [14]. In case the solver returns an evaluation which satisfies the resulting constraint set, the reachability property is satisfied. Otherwise, the property is violated.

4 Factorization of Polynomials

The procedure introduced in the previous section constructs rational functions representing reachability probabilities. We now present an optimization of the frequently used arithmetic operations of *addition*, *multiplication* and *division* of rational functions. During the algorithm presented in Section 3 as well as the mere state-elimination [3], the rational functions that occur rapidly grow even when canceling these functions in every step. Although this exponential blow-up cannot be prevented in general, our experiments show that optimizing the arithmetic operations leads to remarkable speed ups.

The key of the optimization for the operations on rational functions is to maintain a factorization for each polynomial which occurs as numerator or denominator. A polynomial $g = a_1 \cdot x_1^{e_{1,1}} \cdot \ldots \cdot x_n^{e_{1,n}} + \cdots + a_m \cdot x_1^{e_{m,1}} \cdot \ldots \cdot x_n^{e_{m,n}}$ is *normalized* if $(e_{j,1}, \ldots, e_{j,n}) \neq (e_{k,1}, \ldots, e_{k,n})$ for all $j, k \in \{1, \ldots, m\}$ with $j \neq k$ and the monomials are ordered, e. g., according to the reverse lexicographical ordering. A *factorization* $\mathcal{F}_g = \{g_1^{e_1}, \ldots, g_n^{e_n}\}$ of a polynomial g is a set[4] of *factors* $g_i^{e_i}$, where the bases g_i are normalized and pairwise different polynomials, the exponents are $e_i \in \mathbb{N}$, $n = 0$ if $g = 0$, and $g = \prod_{i=1}^{n} g_i^{e_i}$ otherwise. For polynomials g, h and a factorization $\mathcal{F}_g = \{g_1^{e_1}, \ldots, g_n^{e_n}\}$ of g let $\mathit{bases}(\mathcal{F}_g) = \{g_1, \ldots, g_n\}$ and $\exp(h, \mathcal{F}_g)$ be e_i if $g_i = h$ and 0 if $h \notin \mathit{bases}(\mathcal{F}_g)$. As the bases are not required to be irreducible, factorizations are not unique. We maintain that bases and exponents are non-zero, $\mathcal{F}_0 = \emptyset$, $\mathcal{F}_1 = \{1^1\}$, and $1^k \notin \mathcal{F}_g$ for $g \neq 1$. For $\mathcal{F}_g = \{g_1^{e_1}, \ldots, g_n^{e_n}\}$, this is expressed by the reduction $\mathcal{F}_g^{red} = \{1^1\}$ if $n > 0$ and $g_i = 1$ or $e_i = 0$ for all $1 \leq i \leq n$, and $\mathcal{F}_g^{red} = \mathcal{F}_g \setminus \{g_i^{e_i} \mid g_i = 1 \vee e_i = 0\}$ otherwise.

Instead of applying arithmetic operations on two polynomials g_1 and g_2 directly, we operate on their factorizations $\mathcal{F}_{g_1}$ and $\mathcal{F}_{g_2}$. We use the following operations on factorizations: $\mathcal{F}_{g_1} \cup_{\mathcal{F}} \mathcal{F}_{g_2}$ factorizes a (not necessarily least) common multiple of g_1 and g_2, $\mathcal{F}_{g_1} \cap_{\mathcal{F}} \mathcal{F}_{g_2}$ a (not necessarily greatest) common divisor, whereas the binary operations $\cdot_{\mathcal{F}}, /_{\mathcal{F}}, +_{\mathcal{F}}$ correspond to multiplication, division[5] and addition, respectively. Due to space limitations, we omit in the remaining of this paper the trivial cases involving $\mathcal{F}_0$. Therefore we define

$$
\begin{aligned}
\mathcal{F}_{g_1} \cup_{\mathcal{F}} \mathcal{F}_{g_2} &= \{h^{\max(\exp(h,\mathcal{F}_{g_1}),\exp(h,\mathcal{F}_{g_2}))} \mid h \in \mathit{bases}(\mathcal{F}_{g_1}) \cup \mathit{bases}(\mathcal{F}_{g_2})\}^{red} \\
\mathcal{F}_{g_1} \cap_{\mathcal{F}} \mathcal{F}_{g_2} &= \{h^{\min(\exp(h,\mathcal{F}_{g_1}),\exp(h,\mathcal{F}_{g_2}))} \mid h{=}1 \vee h{\in}\mathit{bases}(\mathcal{F}_{g_1}){\cap}\mathit{bases}(\mathcal{F}_{g_2})\}^{red} \\
\mathcal{F}_{g_1} \cdot_{\mathcal{F}} \mathcal{F}_{g_2} &= \{h^{\exp(h,\mathcal{F}_{g_1})+\exp(h,\mathcal{F}_{g_2})} \mid h \in \mathit{bases}(\mathcal{F}_{g_1}) \cup \mathit{bases}(\mathcal{F}_{g_2})\}^{red} \\
\mathcal{F}_{g_1} /_{\mathcal{F}} \mathcal{F}_{g_2} &= \{h^{\max(0,e-\exp(h,\mathcal{F}_{g_2}))} \mid h^e \in \mathcal{F}_{g_1}\}^{red} \\
\mathcal{F}_{g_1} +_{\mathcal{F}} \mathcal{F}_{g_2} &= D \cdot_{\mathcal{F}} \{(\Pi_{g_1' \in \mathcal{F}_{g_1}/_{\mathcal{F}} D}\ g_1') + (\Pi_{g_2' \in \mathcal{F}_{g_2}/_{\mathcal{F}} D}\ g_2')\}^{red}
\end{aligned}
$$

where $D = \mathcal{F}_{g_1} \cap_{\mathcal{F}} \mathcal{F}_{g_2}$ and $\max(a, b)$ ($\min(a, b)$) equals a if $a \geq b$ ($a \leq b$) and b otherwise. Example 3 illustrates the application of the above operations.

[4] We represent a factorization of a polynomial as a set; however, in the implementation we use a more efficient binary search tree instead.

[5] $\mathcal{F}_{g_1} /_{\mathcal{F}} \mathcal{F}_{g_2}$ is a factorization of g_1/g_2 only if $\mathcal{F}_{g_1}$ and $\mathcal{F}_{g_2}$ are sufficiently refined and g_2 divides g_1.

Algorithm 2 gcd computation with factorization refinement

gcd(factorization $\mathcal{F}_{g_1}$, factorization $\mathcal{F}_{g_2}$)
begin
$G := (\mathcal{F}_{g_1} \cap_{\mathcal{F}} \mathcal{F}_{g_2})$ (1)
$F_i := \mathcal{F}_{g_i} /_{\mathcal{F}} G$ and $F'_i := \{1^1\}$ for $i = 1, 2$ (2)
while exists $r_1^{e_1} \in F_1$ with $r_1 \neq 1$ **do** (3)
$F_1 := F_1 \setminus \{r_1^{e_1}\}$ (4)
while $r_1 \neq 1$ and exists $r_2^{e_2} \in F_2$ with $r_2 \neq 1$ **do** (5)
$F_2 := F_2 \setminus \{r_2^{e_2}\}$ (6)
if $\neg$irreducible$(r_1) \vee \neg$irreducible(r_2) **then** $g :=$ common_gcd(r_1, r_2) (7)
else $g := 1$ (8)
if $g = 1$ **then** (9)
$F'_2 := F'_2 \cdot_{\mathcal{F}} \{r_2^{e_2}\}$ (10)
else (11)
$r_1 := \frac{r_1}{g}$ (12)
$F_i := F_i \cdot_{\mathcal{F}} \{g^{e_i - \min(e_1, e_2)}\}$ for $i = 1, 2$ (13)
$F'_2 := F'_2 \cdot_{\mathcal{F}} \{(\frac{r_2}{g})^{e_2}\}$ (14)
$G := G \cdot_{\mathcal{F}} \{g^{\min(e_1, e_2)}\}$ (15)
end if (16)
end while (17)
$F'_1 := F'_1 \cdot_{\mathcal{F}} \{r_1^{e_1}\}$ (18)
$F_2 := \mathcal{F}_{g'_2}$ (19)
$F'_2 := \{1^1\}$ (20)
end while (21)
return (F'_1, F_2, G) (22)
end

For rational functions $\frac{g}{h}$ we maintain separate factorizations $\mathcal{F}_g$ and $\mathcal{F}_h$ for the numerator g and the denominator h, respectively. For multiplication $\frac{g}{h} = \frac{g_1}{h_1} \cdot \frac{g_2}{h_2}$, we compute $\mathcal{F}_g = \mathcal{F}_{g_1} \cdot_{\mathcal{F}} \mathcal{F}_{g_2}$ and $\mathcal{F}_h = \mathcal{F}_{h_1} \cdot_{\mathcal{F}} \mathcal{F}_{h_2}$. For division we use the multiplication due to $\frac{g_1}{h_1} : \frac{g_2}{h_2} = \frac{g_1}{h_1} \cdot \frac{h_2}{g_2}$.

For the addition $\frac{g}{h} = \frac{g_1}{h_1} + \frac{g_2}{h_2}$, we compute h as a common multiple of h_1 and h_2 factorized by $\mathcal{F}_h = \mathcal{F}_{h_1} \cup_{\mathcal{F}} \mathcal{F}_{h_2}$, such that $h = h_i \cdot h'_i$ with $\mathcal{F}_{h'_i} = \mathcal{F}_h /_{\mathcal{F}} \mathcal{F}_{h_i}$ for $i = 1, 2$. For the numerator g we first determine a common divisor d of g_1 and g_2 by $\mathcal{F}_d = \mathcal{F}_{g_1} \cap_{\mathcal{F}} \mathcal{F}_{g_2}$, such that $g_i = d \cdot g'_i$ with $\mathcal{F}_{g'_i} = \mathcal{F}_{g_i} /_{\mathcal{F}} \mathcal{F}_d$ for $i = 1, 2$. The numerator g is $d \cdot (g'_1 \cdot h'_1 + g'_2 \cdot h'_2)$ with factorization $\mathcal{F}_{d \cdot \mathcal{F}} (\mathcal{F}_{g'_1} \cdot_{\mathcal{F}} \mathcal{F}_{h'_1} +_{\mathcal{F}} \mathcal{F}_{g'_2} \cdot_{\mathcal{F}} \mathcal{F}_{h'_2})$.

The rational function $\frac{g}{h}$ resulting from the addition is further simplified by cancelation, i. e., dividing g and h by their greatest common divisor (gcd) g'. Given the factorizations $\mathcal{F}_g$ and $\mathcal{F}_h$, Algorithm 2 calculates the factorizations $\mathcal{F}_{g'}$, $\mathcal{F}_{\frac{g_1}{g'}}$, and $\mathcal{F}_{\frac{g_2}{g'}}$ by reusing $\mathcal{F}_g$ and $\mathcal{F}_h$ as much as possible. Initially, a factorization G of a common divisor of g_1 and g_2 is set to $\mathcal{F}_{g_1} \cap_{\mathcal{F}} \mathcal{F}_{g_2}$. The factors of g_1 and g_2, which are not part of the factorization, are stored in F_1 resp. F_2. Moreover, F'_1 and F'_2 contain those factors of g_1 resp. g_2, such that for all $(f'_1,\ f'_2) \in F'_1 \times F'_2$ f'_1 and f'_2 have no common divisors. The algorithm now iteratively adds further common divisors of g_1 and g_2 to G until it is a factorization of their gcd. For this purpose,

we consider for each factor in F_1 all factors in F_2 and calculate the gcd of their bases which we compute by a common approach for calculating gcds. Note that the main concern of Algorithm 2 is to avoid this rather expensive operation or else call it on preferably simple polynomials. Where the latter is entailed by the idea of using factorizations, the former can be achieved by excluding pairs of factors for which we can cheaply decide that both are *irreducible*, i. e., they have no non-trivial divisors. If factors $r_1^{e_1} \in F_1$ and $r_2^{e_2} \in F_2$ with $g := \mathit{common\_gcd}(r_1, r_2) = 1$ are found, we just shift $r_2^{e_2}$ from F_2 to F_2'. Otherwise, we can add $g^{\min(e_1,e_2)}$, which is the gcd of $r_1^{e_1}$ and $r_2^{e_2}$, to G and extend the factors F_1 resp. F_2, which could still contain common divisors, by $g^{e_1-\min(e_1,e_2)}$ resp. $g^{e_2-\min(e_1,e_2)}$. Furthermore, F_2' obtains the new factor $(\frac{r_2}{g})^{e_2}$, which has certainly no common divisor with any factor in F_1'. Finally, we set the basis r_1 to $\frac{r_1}{g}$, excluding the just found common divisor. If all factors in F_2 have been considered for common divisors with r_1, we can add it to F_1' and continue with the next factor in F_1, for which we must reconsider all factors in F_2' and, therefore, shift them to F_2. The algorithm terminates, if the last factor of F_1 has been processed, returning the factorizations $\mathcal{F}_{g'}$, $\mathcal{F}_{\frac{g_1}{g'}}$ and $\mathcal{F}_{\frac{g_2}{g'}}$, which we can use to refine the factorizations of g_1 and g_2 via $\mathcal{F}_{g_1} := \mathcal{F}_{\frac{g_1}{g'}} \cdot_{\mathcal{F}} G$ and $\mathcal{F}_{g_2} := \mathcal{F}_{\frac{g_2}{g'}} \cdot_{\mathcal{F}} G$.

Example 3. Assume we want to apply Algorithm 2 to the factorizations $\mathcal{F}_{xyz} = \{(xyz)^1\}$ and $\mathcal{F}_{xy} = \{(x)^1,\ (y)^1\}$. We initialize $G = F_1' = F_2' = \{(1)^1\}$, $F_1 = \mathcal{F}_{xyz}$ and $F_2 = \mathcal{F}_{xy}$. First, we choose the factors $(r_1)^{e_1} = (xyz)^1$ and $(x)^1$ and remove them from F_1 resp. F_2. The gcd of their bases is x, hence we only update r_1 to $(yz)^1$ and G to $\{(x)^1\}$. Then we remove the next and last element $(y)^1$ from F_2. Its basis and r_1 have the gcd y and we therefore update r_1 to $(z)^1$ and G to $\{(x)^1,\ (y)^1\}$. Finally, we add $(z)^1$ to F_1' and return the expected result $(\{(z)^1\},\ \{(1)^1\},\ \{(x)^1,\ (y)^1\})$. Furthermore, we can update $\mathcal{F}_{xyz} = F_1' \cdot_{\mathcal{F}} G = \{(x)^1,\ (y)^1,\ (z)^1\}$ afterwards.

Theorem 2. *Let p_1 and p_2 be polynomials with factorizations $\mathcal{F}_{p_1}$ resp. $\mathcal{F}_{p_2}$ as before. Applying Algorithm 2 to these factorizations results in $\mathit{gcd}(\mathcal{F}_{p_1},\ \mathcal{F}_{p_2}) = (\mathcal{F}_{r_1},\ \mathcal{F}_{r_2},\ G)$ with G being a factorization of the greatest common divisor g of p_1 and p_2 and $\mathcal{F}_{r_1}$ and $\mathcal{F}_{r_2}$ being factorizations of $\frac{p_1}{g}$ resp. $\frac{p_2}{g}$.*

The proof of this theorem can be found in the appendix.

5 Experiments

We developed a C++ prototype implementation of our approach using the arithmetic library `GiNaC` [15]. The prototype is available for testing on the project homepage[6]. Moreover, we implemented the state-elimination approach that is used by PARAM [6] using our optimized factorization approach to provide a more distinct comparison. All experiments were run on an Intel Core 2 Quad CPU 2.66 GHz with 4 GB of memory. We defined a timeout (TO) of 3600 seconds and a

[6] `http://goo.gl/nS378q`

memory bound (*MO*) of 4 GB. We report on three case studies; a more distinct description and the specific instances we used are available at our homepage.

The *bounded retransmission protocol* (BRP) [16] models the sending of files via an unreliable network, manifested in two lossy channels for sending and acknowledging the reception. This model is parametrized in the probability of reliability of those channels. The *crowds protocol* (CROWDS) [17] is designed for anonymous network communication using random routing, parametrized in how many members are "good" or "bad" and if a good member delivers a message or randomly routes it to another member. *NAND multiplexing* (NAND) [18] models how reliable computations are obtained using unreliable hardware by having a certain number of copies of a NAND unit all doing the same job. Parameters are the probabilities of faultiness of the units and of erroneous inputs. The experimental setting includes our SCC-based approach as described in Section 3 using the optimized factorization of polynomials as in Section 4 (SCC MC), the state elimination as in PARAM but also using the approach of Section 4 (STATE ELIM) and PARAM itself.[7] For all instances we list the number of states and transitions; for each tool we give the running time in seconds and the memory consumption in MB; the best time is boldfaced. Moreover, for our approaches we mention the number of polynomials which are intermediately stored.

Model	Graph		SCC MC			STATE ELIM			PARAM	
—	States	Trans.	Time	Poly	Mem	Time	Poly	Mem	Time	Mem
BRP	2695	3459	**1.86**	2319	16.04	1.97	6647	40.04	36.49	17.38
BRP	5383	6915	14.81	4623	47.00	**12.19**	13367	161.80	356.04	63.21
BRP	10378	13827	TO	—	—	**63.47**	21805	328.94	3203.20	431.40
BRP	10759	13827	147.31	9231	176.89	**85.54**	26807	682.24	3511.96	304.07
CROWDS	8655	14953	4.15	8747	13.21	**3.24**	2943	11.96	139.70	10.44
CROWDS	37293	65011	**16.69**	33549	40.23	21.72	8148	30.61	1977.95	35.39
CROWDS	198201	348349	**80.05**	155000	175.40	271.04	27344	133.82	TO	—
CROWDS	726379	1283297	**262.88**	500048	668.94	1742.42	73702	477.26	TO	—
CROWDS	2888763	5127151	**1030.57**	1707776	2705.35	TO	—	—	TO	—
NAND	14323	21567	39.71	25504	366.79	59.60	405069	926.33	**15.26**	16.89
NAND	28183	42287	208.41	44799	1405.16	218.85	925324	3708.27	**50.45**	30.47
NAND	35113	52647	352.09	54445	2047.66	364.09	1184848	3696.39	**78.19**	40.51
NAND	78334	121512	**639.29**	184799	3785.11	TO	—	—	1127.83	113.32

For BRP, STATE ELIM outperforms PARAM by up to two orders of magnitude while SCC MC runs into a timeout for one particular instance. This is due to the graph structure. In contrast, the crowds protocol induces a nested SCC structure, which is very hard for PARAM since many divisions of polynomials have to be carried out. On larger benchmarks, it is therefore outperformed by more than three orders of magnitude while SCC MC performs best. The NAND graphs consist of single paths, which induces a high number of polynomials we are keeping for the factorization. Our implementation offers the possibility to

[7] Note that no bisimulation reduction was applied to any of the input models, which would improve the feasibility of all approaches likewise.

bound this pool of polynomials which highly decreases the memory consumption for the sake of loosing information about the factorizations.

6 Conclusion and Future Work

We presented a new approach to verify parametric Markov chains together with an improved factorization of polynomials. We were able to highly improve the scalability in comparison to existing approaches. Future work will be dedicated to the actual parameter synthesis. First, we want to incorporate interval constraint propagation [19] in order to provide reasonable intervals for the parameters where properties are satisfied or violated. Moreover, we are going to investigate the possibility of extending our approaches to models with costs.

References

1. Su, G., Rosenblum, D.S.: Asymptotic bounds for quantitative verification of perturbed probabilistic systems. In: Proc. of ICFEM. Volume 8144 of LNCS, Springer (2013) 297–312
2. Daws, C.: Symbolic and parametric model checking of discrete-time Markov chains. In: Proc. of ICTAC. Volume 3407 of LNCS, Springer (2004) 280–294
3. Hahn, E.M., Hermanns, H., Zhang, L.: Probabilistic reachability for parametric Markov models. Software Tools for Technology Transfer **13**(1) (2010) 3–19
4. Gruber, H., Johannsen, J.: Optimal lower bounds on regular expression size using communication complexity. In: Proc. of FOSSACS. Volume 4962 of LNCS, Springer (2008) 273–286
5. Hahn, E.M., Han, T., Zhang, L.: Synthesis for PCTL in parametric Markov decision processes. In: Proc. of NFM. Volume 6617 of LNCS, Springer (2011) 146–161
6. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: PARAM: A model checker for parametric markov models. In: Proc. of CAV. Volume 6174 of LNCS, Springer (2010) 660–664
7. Han, T., Katoen, J.P., Mereacre, A.: Approximate parameter synthesis for probabilistic time-bounded reachability. In: Proc. of RTSS, IEEE CS (2008) 173–182
8. Puggelli, A., Li, W., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Polynomial-time verification of PCTL properties of MDPs with convex uncertainties. In: Proc. of CAV. Volume 8044 of LNCS, Springer (2013) 527–542
9. Benedikt, M., Lenhardt, R., Worrell, J.: LTL model checking of interval Markov chains. In: Proc. of TACAS. Volume 7795 of LNCS, Springer (2013) 32–46
10. Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press (2008)
11. Ábrahám, E., Jansen, N., Wimmer, R., Katoen, J.P., Becker, B.: DTMC model checking by SCC reduction. In: Proc. of QEST, IEEE CS (2010) 37–46
12. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM Journal on Computing **1**(2) (1972) 146–160
13. Quarteroni, A., Sacco, R., Saleri, F.: Numerical Mathematics. Springer (2000)
14. Jovanovic, D., de Moura, L.M.: Solving non-linear arithmetic. In: Proc. of IJCAR. Volume 7364 of LNCS, Springer (2012) 339–354
15. Bauer, C., Frink, A., Kreckel, R.: Introduction to the GiNaC framework for symbolic computation within the C++ programming language. J. Symb. Comput. **33**(1) (2002) 1–12

16. Helmink, L., Sellink, M., Vaandrager, F.: Proof-checking a data link protocol. In: Proc. of TYPES. Volume 806 of LNCS, Springer (1994) 127–165
17. Reiter, M.K., Rubin, A.D.: Crowds: Anonymity for web transactions. ACM Trans. on Information and System Security **1**(1) (1998) 66–92
18. Han, J., Jonker, P.: A system architecture solution for unreliable nanoelectronic devices. IEEE Transactions on Nanotechnology **1** (2002) 201–208
19. Fränzle, M., Herde, C., Teige, T., Ratschan, S., Schubert, T.: Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. Journal on Satisfiability, Boolean Modeling, and Computation **1**(3-4) (2007) 209–236

Appendix

Here, we show the correctness of the PDTMC abstraction as in Definition 9. First, let us recall Theorem 1

Theorem 3. *For a PDTMC $\mathcal{M} = (S, V, I, P)$ and its abstraction $\mathcal{M}_{abs} = (S_{abs}, I_{abs}, V_{abs}, P_{abs})$ according to Definition 9 it holds for all initial states $s_I \in S_I$ and all absorbing states $t \in T$ that*

$$Pr^{\mathcal{M}}_{fin}(Paths^{\mathcal{M}}(s_I, t)) = Pr^{\mathcal{M}_{abs}}_{fin}(Paths^{\mathcal{M}_{abs}}(s_I, t)).$$

Proof. As the bottom SCCs are exactly the absorbing target states in T, the probability of reaching a state of T is 1. The probability $p^{\mathcal{M}}_{abs}(s_I, s_I)$ can therefore be expressed w. r. t. the probabilities of reaching an absorbing state without revisiting s_I:

$$p^{\mathcal{M}}_{abs}(s_I, s_I) = 1 - \sum_{t \in T} p^{\mathcal{M}}_{abs}(s_I, t). \tag{5}$$

To reduce notation, we define the set of paths R_{loop} looping on s_I and the set of paths R_{out} going to some $t \in T$ without revisiting s_I.

$$R_{loop} = \{s_I s_1 \dots s_n s_I \in Paths^{\mathcal{M}} \mid s_i \notin \{s_I\} \cup T, 1 \leq i \leq n\} \tag{6}$$

$$R_{out} = \{s_I s_1 \dots s_n t \in Paths^{\mathcal{M}} \mid s_i \notin \{s_I\} \cup T, 1 \leq i \leq n, t \in T\} \tag{7}$$

As the self-loop in s_I represents the paths of R_{loop}, it holds that

$$p^{\mathcal{M}}_{abs}(s_I, s_I) = Pr_{fin}(R_{loop}). \tag{8}$$

We now have:

$$\begin{aligned}
& Pr^{\mathcal{M}}_{fin}(Paths^{\mathcal{M}}(s_I, t)) \\
= \; & Pr^{\mathcal{M}}_{fin}(\bigcup_{i=0}^{\infty} \{\pi_1 \cdot \dots \cdot \pi_i \cdot \pi_{out} \mid \pi_j \in R_{loop}, 1 \leq j \leq i;\ \pi_{out} \in R_{out}\}) \\
= \; & \sum_{i=0}^{\infty} Pr^{\mathcal{M}}_{fin}(\{\pi_1 \cdot \dots \cdot \pi_i \cdot \pi_{out} \mid \pi_j \in R_{loop}, 1 \leq j \leq i;\ \pi_{out} \in R_{out}\}) \\
= \; & \sum_{i=0}^{\infty} (Pr^{\mathcal{M}}_{fin}(R_{loop}))^i \cdot Pr^{\mathcal{M}}_{fin}(R_{out}) \\
= \; & \sum_{i=0}^{\infty} (p^{\mathcal{M}}_{abs}(s_I, s_I))^i \cdot Pr^{\mathcal{M}}_{fin}(R_{out}) \text{ (Equation (8))} \\
= \; & \frac{1}{1 - p^{\mathcal{M}}_{abs}(s_I, s_I)} \cdot Pr^{\mathcal{M}}_{fin}(R_{out}) \text{ (Geometric Series)} \\
= \; & \frac{1}{\sum_{s_{out} \in T} p^{\mathcal{M}}_{abs}(s_I, s_{out})} \cdot Pr^{\mathcal{M}}_{fin}(R_{out}) \text{ (Equation (5))}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{\sum_{s_{out} \in T} p^{\mathcal{M}}_{abs}(s_I, s_{out})} \cdot p^{\mathcal{M}}_{abs}(s_I, t) \text{ (Definition 8)} \\
&= P_{abs}(s_I, t) \text{ (Definition 9)} \\
&= Pr^{\mathcal{M}_{abs}}_{fin}(Paths^{\mathcal{M}_{abs}}(s_I, t))
\end{aligned}
$$

As the probabilities of reaching the absorbing states from initial states coincide in $\mathcal{M}$ and $\mathcal{M}_{abs}$, our abstraction is valid.

We show the correctness and completeness of Algorithm 2 by proving the following theorem.

Theorem 4. *Let p_1 and p_2 be polynomials with factorizations $\mathcal{F}_{p_1}$ resp. $\mathcal{F}_{p_2}$ as before. Applying Algorithm 2 to these factorizations results in $gcd(\mathcal{F}_{p_1}, \mathcal{F}_{p_2}) = (\mathcal{F}_{r_1}, \mathcal{F}_{r_2}, G)$ with G being a factorization of the greatest common divisor g of p_1 and p_2 and $\mathcal{F}_{r_1}$ and $\mathcal{F}_{r_2}$ being factorizations of $\frac{p_1}{g}$ resp. $\frac{p_2}{g}$.*

Proof. We define the product of a factorization $\mathcal{F}_p$ by $\mathcal{P}(\mathcal{F}_p) = \prod_{q^e \in \mathcal{F}_p} q^e$ and the common gcd by gcd_c. We first prove that

$$
\begin{aligned}
F_1 \cdot_{\mathcal{F}} F_1' \cdot_{\mathcal{F}} \{(r_1)^{e_1}\} \cdot_{\mathcal{F}} G &= \mathcal{F}_{p_1}, && (9) \\
F_2 \cdot_{\mathcal{F}} F_2' \cdot_{\mathcal{F}} G &= \mathcal{F}_{p_2}, && (10) \\
gcd_c(\mathcal{P}(F_1' \cdot_{\mathcal{F}} \{(r_1)^{e_1}\}),\ \mathcal{P}(F_2')) &= 1 && (11)
\end{aligned}
$$

hold after the ith pass through the inner while-loop (line 16) of Algorithm 2, for all $i \in \mathbb{N} \cup \{0\}$, where we can assume without loss of generality that $(r_1)^{e_1}$ is initialized by $(1)^1$ in the beginning of Algorithm 2. In the following we denote by $F_1^{(i)}$, $F_1'^{(i)}$, $F_2^{(i)}$, $F_2'^{(i)}$, $r_1^{(i)}$, $G^{(i)}$ the according datastructures in the ith pass through the inner while-loop.

Basis ($i = 0$): Before entering the outer while-loop and, hence, before entering the inner while-loop (line 2) Equation 9 and Equation 10 hold as a consequence of the definiton of $\cdot_{\mathcal{F}}$. Equation 11 is trivially implied from $F_1'^{(0)} = F_2'^{(0)} = \{(1)^1\}$.

Inductive step ($i > 0$): If $g = 1$ (line 9) we only shift $r_2^{e_2}$ from F_2 to F_2', thus the left-hand sides of Equation 9 and Equation 10 have the same result as in the previous pass through the inner while-loop ($i-1$) and both equations follow from the inductive hypothesis. Concerning Equation 11 either $F_1' = \{(1)^1\}$ or all elements it contains have been added directly after exiting the inner while-loop and therefore the inductive hypothesis still holds. After resetting F_2 in line 19 it holds that

$$
\begin{aligned}
& gcd_c(\mathcal{P}(F_1' \cdot_{\mathcal{F}} \{(r_1)^{e_1}\}),\ \mathcal{P}(F_2)) \\
= {} & gcd_c(\mathcal{P}(F_1' \cdot_{\mathcal{F}} \{(r_1)^{e_1}\}),\ \mathcal{P}(F_2')) \\
\overset{IH}{=} {} & 1
\end{aligned}
$$

As we do not add elements to F_2 nor F_1' anywhere else, $\mathcal{P}(F_2)$ and $\mathcal{P}(F_1')$ are always *coprime* (*), which means that they have no common divisors. It follows that

$$
\begin{aligned}
& gcd_c(\mathcal{P}(F_1'^{(i)} \cdot_{\mathcal{F}} \{(r_{1,i})^{e_1}\}),\ \mathcal{P}(F_2'^{(i)})) \\
&= gcd_c(\mathcal{P}(F_1'^{(i-1)} \cdot_{\mathcal{F}} \{(r_{1,i})^{e_1}\}),\ \mathcal{P}(F_2'^{(i-1)} \cdot_{\mathcal{F}} \{(r_2)^{e_2}\})) \\
&= gcd_c(\mathcal{P}(F_1'^{(i-1)}) \cdot (r_{1,i})^{e_1},\ \mathcal{P}(F_2'^{(i-1)}) \cdot r_2^{e_2}) \\
&\leq gcd_c(\mathcal{P}(F_1'^{(i-1)}) \cdot (r_{1,i})^{e_1},\ \mathcal{P}(F_2'^{(i-1)})) \cdot gcd_c(\mathcal{P}(F_1'^{(i-1)}) \cdot (r_{1,i})^{e_1},\ r_2^{e_2}) \\
&\overset{IH}{=} gcd_c(\mathcal{P}(F_1'^{(i-1)}) \cdot (r_{1,i})^{e_1},\ r_2^{e_2}) \\
&\leq gcd_c(\mathcal{P}(F_1'^{(i-1)}),\ r_2^{e_2}) \cdot gcd_c((r_{1,i})^{e_1},\ r_2^{e_2}) \\
&\overset{(*)}{=} gcd_c((r_{1,i})^{e_1},\ r_2^{e_2}) \\
&\overset{g=1}{=} 1
\end{aligned}
$$

and therefore Equation 11 holds.

If $g \neq 1$, then Equation 9 holds as

$$
\begin{aligned}
& F_1^{(i)} \cdot_{\mathcal{F}} F_1'^{(i)} \cdot_{\mathcal{F}} \{(r_{1,i})^{e_1}\} \cdot_{\mathcal{F}} G^{(i)} \\
&= F_1^{(i-1)} \cdot_{\mathcal{F}} \{g^{e_1 - \min(e_1,e_2)}\} \cdot_{\mathcal{F}} F_1'^{(i-1)} \cdot_{\mathcal{F}} \{(\frac{r_{1,i-1}}{g})^{e_1}\} \\
&\quad \cdot_{\mathcal{F}} G^{(i-1)} \cdot_{\mathcal{F}} \{g^{\min(e_1,e_2)}\} \\
&= F_1^{(i-1)} \cdot_{\mathcal{F}} F_1'^{(i-1)} \cdot_{\mathcal{F}} \{(r_{1,i-1})^{e_1}\} \cdot_{\mathcal{F}} G^{(i-1)} \\
&\overset{IH}{=} \mathcal{F}_{p_1}
\end{aligned}
$$

and Equation 10 holds because of

$$
\begin{aligned}
& F_2^{(i)} \cdot_{\mathcal{F}} F_2'^{(i)} \cdot_{\mathcal{F}} G^{(i)} \\
&= (F_2^{(i-1)} /_{\mathcal{F}} \{r_2^{e_2}\}) \cdot_{\mathcal{F}} \{g^{e_2 - \min(e_1,e_2)}\} \cdot_{\mathcal{F}} F_2'^{(i-1)} \cdot_{\mathcal{F}} \{(\frac{r_2}{g})^{e_2}\} \\
&\quad \cdot_{\mathcal{F}} G^{(i-1)} \cdot_{\mathcal{F}} \{g^{\min(e_1,e_2)}\} \\
&= F_2^{(i-1)} \cdot_{\mathcal{F}} F_2'^{(i-1)} \cdot_{\mathcal{F}} G^{(i-1)} \\
&\overset{IH}{=} \mathcal{F}_{p_2}.
\end{aligned}
$$

Furthermore, Equation 11 holds as a consequence of

$$
\begin{aligned}
& gcd_c(\mathcal{P}(F_1'^{(i)} \cdot_{\mathcal{F}} \{(r_{1,i})^{e_1}\}),\ \mathcal{P}(F_2'^{(i)})) \\
= \ & gcd_c(\ \mathcal{P}(F_1'^{(i-1)} \cdot_{\mathcal{F}} \{(\frac{r_{1,i-1}}{g})^{e_1}\}),\ \mathcal{P}(F_2'^{(i-1)} \cdot_{\mathcal{F}} \{(\frac{r_2}{g})^{e_2}\})\ ) \\
\leq \ & gcd_c(\mathcal{P}(F_1'^{(i)} \cdot_{\mathcal{F}} \{(r_{1,i-1})^{e_1}\}),\ \mathcal{P}(F_2'^{(i-1)})) \\
& \cdot\ gcd_c(\mathcal{P}(F_1'^{(i-1)}),\ (\frac{r_2}{g})^{e_2})\ \cdot\ gcd_c((\frac{r_{1,i-1}}{g})^{e_1},\ (\frac{r_2}{g})^{e_2}) \\
\overset{IH}{=} \ & gcd_c(\mathcal{P}(F_1'^{(i-1)}),\ (\frac{r_2}{g})^{e_2})\ \cdot\ gcd_c((\frac{r_{1,i-1}}{g})^{e_1},\ (\frac{r_2}{g})^{e_2}) \\
\overset{g \text{ } gcd}{\underset{of\ r_1\ r_2}{=}} \ & gcd_c(\mathcal{P}(F_1'^{(i-1)}),\ (\frac{r_2}{g})^{e_2}) \\
\overset{(*)}{=} \ & 1.
\end{aligned}
$$

Now we can prove the completeness and correctness of Algorithm 2.

Completeness: When passing the outer while-loop it holds that

$$
\frac{\mathcal{P}(F_1'^{(i)})}{\mathcal{P}(F_1'^{(i-1)})}\ \geq\ 1, \quad \frac{\mathcal{P}(F_2'^{(i)})}{\mathcal{P}(F_2'^{(i-1)})}\ \geq\ 1 \quad \text{and} \quad \frac{\mathcal{P}(G^{(i)})}{\mathcal{P}(G^{(i-1)})} \geq\ 1,
$$

and for at least one of these inequalities the relation is even strict ($>$). From Equation 9 and Equation 10, the fact that $\mathcal{F}_{p_1}$ and $\mathcal{F}_{p_2}$ are constant and we only consider factorizations of polynomials ≥ 1 (in particular F_1 and F_2) follows the completeness of Algorithm 2.

Correctness: When Algorithm 2 leaves the outer while-loop it holds that $F_1 = F_2' = \{(1)^1\}$. This and the fact that Equation 9 and Equation 10 are valid in the end of the inner while-loop implies:

$$
\begin{aligned}
F_1' \cdot_{\mathcal{F}} G\ &= \mathcal{F}_{p_1} \\
F_2 \cdot_{\mathcal{F}} G\ &= \mathcal{F}_{p_2}
\end{aligned}
$$

As also Equation 11 holds, Algorithm 2 is correct.