| | |
|---|---|
| **Project no.:** | **PIRSES-GA-2011-295261** |
| **Project full title:** | **Mobility between Europe and Argentina applying Logics to Systems** |
| **Project Acronym:** | **MEALS** |
| **Deliverable no.:** | **1.4 / 2** |
| **Title of Deliverable:** | **Accelerating Parametric Probabilistic Verification** |

| | |
|---|---|
| **Contractual Date of Delivery to the CEC:** | **30-Sep-2015** |
| **Actual Date of Delivery to the CEC:** | **30-Sep-2015** |
| **Organisation name of lead contractor for this deliverable:** | **UNC** |
| **Author(s):** | **Nils Jansen, Florian Corzilius, Matthias Volk, Ralf Wimmer, Erika Ábrahám, Joost-Pieter Katoen, and Bernd Becker** |
| **Participants(s):** | **RTW** |
| **Work package contributing to the deliverable:** | **WP1** |
| **Nature:** | **R** |
| **Dissemination Level:** | **Public** |
| **Total number of pages:** | **22** |
| **Start date of project:** | 1 Oct. 2011 **Duration:** 48 month |

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 these approaches can lead to a speed-up of up to several orders of magnitude in comparison to existing approaches.

Note:

This deliverable has been published in *Quantitative Evaluation of Systems*, LNCS 8657, pp 404-420. Sprnger, 2014.

*This project has received funding from the European Union Seventh Framework Programme (FP7 2007-2013) under Grant Agreement Nr. 295261.*

# Contents

# 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), 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 DTMCs is typically done by solving linear equation systems. This is not feasible for PDTMCs, since the resulting equation system is non-linear. Instead, approaches based on *state elimination* have been proposed [2, 3]. The idea is to replace states and their incident transitions by direct transitions from the predecessors to the successors. Eliminating states iteratively leads to a model having only initial and absorbing states, where transitions between these states carry—as rational functions over the model parameters—the probability of reaching the absorbing states from the initial states. The efficiency of such methods strongly depends on the order in which states are eliminated and on the representation of rational functions.

**Related work** The idea of constructing a regular expression representing a DTMC's behavior originates from Daws [2]. He uses state elimination to generate regular expressions describing the paths from the initial states to the absorbing states of a DTMC. 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 common heuristics for the generation of regular expressions [4] to guide the elimination of states. Additionally, they simplify the rational functions. These ideas have been extended to Markov decision processes [5]. The main problem is that the reachability probabilities depend on the chosen scheduler to resolve the nondeterminism. When maximizing or minimizing these probabilities, the optimal scheduler generally depends on the values of the parameters. These concepts are implemented in PARAM [6] and recently also in PRISM [7], which are—to the best of our knowledge—the only available tools for computing reachability probabilities of PDTMCs.

Several authors have considered the related problem of parameter synthesis: for which parameter instances does a given (LTL or PCTL) formula hold? For instance, Han *et al.* [8] considered this problem for timed reachability in continuous-time Markov chains, Pugelli *et al.* [9] for Markov decision processes, and Benedikt *et al.* [10] for $\omega$-regular properties of interval Markov chains.

**Contributions of this paper** In this paper we improve the computation of reachability probabilities for PDTMCs [2, 3] in two important ways. First, we introduce a state elimination strategy based on a *recursive graph decomposition* of the PDTMC into strongly connected subgraphs. Each (sub-)SCC is replaced by abstract transitions that lead from its ingoing states to its outgoing states. The resulting rational functions describe exactly the probability of entering the SCC and leaving it eventually. Secondly, we give a novel method to perform arithmetic operations directly on a *factorization of polynomials*. As many benchmarks have a symmetric structure, identical polynomials occur very often; therefore a maintenance of partial factorizations often speeds up the cancelation of rational functions. Although presented in the context of parametric Markov chains, this constitutes a generic method for representing and manipulating polynomials and rational functions or is well-suited for other applications as well. The experiments show that using our techniques yields a speed-up of up to three orders of magnitude compared to [3] on many benchmarks.

An extended version of this paper including all proofs can be found in [11].

## 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 $\mathrm{succ}(s) = \{s' \in S \mid P(s, s') > 0\}$. 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 $0 \le i < n$. The set $\mathrm{Paths}^{\mathcal{D}}$ contains all paths of $\mathcal{D}$, $\mathrm{Paths}^{\mathcal{D}}(s)$ those starting in $s \in S$, and $\mathrm{Paths}^{\mathcal{D}}(s, t)$ those starting in $s$ and ending in $t$. We generalize this to sets $S', S'' \subseteq S$ of states by $\mathrm{Paths}^{\mathcal{D}}(S', S'') = \bigcup_{s'\in S'} \bigcup_{s''\in S''} \mathrm{Paths}^{\mathcal{D}}(s', s'')$. A state $t$ is *reachable* from $s$ iff $\mathrm{Paths}^{\mathcal{D}}(s, t) \neq \emptyset$.

The *probability measure* $\mathrm{Pr}^{\mathcal{D}}$ for paths satisfies

$$\mathrm{Pr}^{\mathcal{D}}(s_0 \dots s_n) = \prod_{i=0}^{n-1} P(s_i, s_{i+1})$$

and $\mathrm{Pr}^{\mathcal{D}}(\{\pi_1, \pi_2\}) = \mathrm{Pr}^{\mathcal{D}}(\pi_1) + \mathrm{Pr}^{\mathcal{D}}(\pi_2)$ for all $\pi_1, \pi_2 \in \mathrm{Paths}^{\mathcal{D}}$ not being the prefix of each other. In general, for $R \subseteq \mathrm{Paths}^{\mathcal{D}}$ we have $\mathrm{Pr}^{\mathcal{D}}(R) = \sum_{\pi\in R'} \mathrm{Pr}^{\mathcal{D}}(\pi)$ with $R' = \{\pi \in R \mid \forall \pi' \in R.\ \pi'$ is not a proper prefix of $\pi\}$. We often omit the superscript $\mathcal{D}$ if it is clear from the context. For more details see, e. g., [12].

For a DTMC $\mathcal{D} = (S, I, P)$ and some $K \subseteq S$ we define the set of *input states* of $K$ by $\mathrm{Inp}(K) = \{s \in K \mid I(s) > 0 \vee \exists s' \in S \setminus K.\ P(s', s) > 0\}$, 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 $\mathrm{Out}(K) = \{s \in S \setminus K \mid \exists s' \in K.\ P(s', s) > 0\}$, 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 $K \setminus \mathrm{Inp}(K)$.

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 $\mathrm{Paths}^{\mathcal{D}}(\{s'\}, S \setminus S') = \emptyset$. 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 $S$. An SCC $S'$ is called *bottom* if $\mathrm{Out}(S') = \emptyset$ holds. The probability of eventually reaching a bottom SCC in a finite DTMC is always 1 [12, Chap. 10.1].

We consider *probabilistic reachability properties*, putting bounds on the probability $\sum_{s_I \in S_I} I(s_I) \cdot \mathrm{Pr}^{\mathcal{D}}(\mathrm{Paths}^{\mathcal{D}}(s_I, T))$ to eventually reach a set $T \subseteq S$ of states from the initial states. It is well-known that this suffices for checking arbitrary $\omega$-regular properties, see [12, Chap. 10.3] for the details. The probability of reaching a bottom SCC equals the probability of reaching one of its input states. Therefore, we can make all input states of bottom SCCs absorbing, without loss of information. Furthermore, if we are interested in the probability to reach a given state, also this state can be made absorbing without modifying the reachability probability of interest. Therefore, in the following we consider only models whose bottom SCCs are single absorbing states forming the set $T$ of *target* states, whose reachability probabilities are of interest.

## 2.1 Parametric Markov Chains

To add parameters to DTMCs, we allow arbitrary rational functions in the definition of probability distributions [6].

**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 $1 \le i \le m$ and $1 \le j \le n$. $\mathbb{Z}[x_1, \dots, x_n]$ denotes the set of polynomials over $V = \{x_1, \dots, x_n\}$. A *rational function* over $V$ is a quotient $f = \frac{g_1}{g_2}$ of two polynomials $g_1, g_2$ over $V$ with $g_2 \neq 0$[1]. We use $\mathcal{F}_V = \{\frac{g_1}{g_2} \mid g_1, g_2 \in \mathbb{Z}[x_1, \dots, x_n] \wedge g_2 \neq 0\}$ 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 $V = \{x_1, \dots, x_n\}$ with domain $\mathbb{R}$, an initial distribution $I : S \to \mathcal{F}_V$, and a parametric transition probability matrix $P : S \times S \to \mathcal{F}_V$.

The *underlying graph* $\mathcal{G}_{\mathcal{M}} = (S, \mathcal{D}_P)$ of a (P)DTMC $\mathcal{M} = (S, V, I, P)$ is given by $\mathcal{D}_P = \{(s, s') \in S \times S \mid P(s, s') \neq 0\}$. As for DTMCs, we assume that all bottom SCCs of considered PDTMCs are single absorbing states.

**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 $g \in \mathbb{Z}[x_1, \dots, x_n]$ 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 $g_2[u] \neq 0$.

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

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

An evaluation $u$ substitutes the parameters by real numbers. Well-defined probability measures are induced under the following conditions:

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

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

An evaluation $u$ is called *graph preserving* if is well-defined and 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, which will be ensured during the model checking algorithm by requiring a graph preserving evaluation $u$, i. e., $\mathcal{G}_{\mathcal{M}} = \mathcal{G}_{\mathcal{M}_u}$. This is necessary, otherwise altering the graph could make reachable states unreachable, thereby changing reachability probabilities.

**Definition 6.** Given a PDTMC $\mathcal{M} = (S, V, I, P)$ with absorbing 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 $f_{s_I,t}[u] = \Pr^{\mathcal{M}_u}(\mathrm{Paths}^{\mathcal{M}_u}(s_I, t))$.

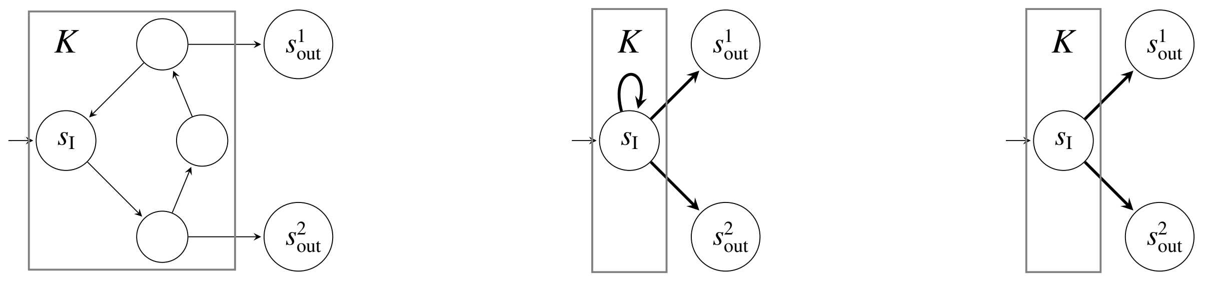
Given the functions $f_{s_I,t}$ for $s_I \in S_I$ and $t \in T$, the probability of reaching a state in $T$ from an initial state is $\sum_{s_I \in S_I} I(s_I) \cdot \left(\sum_{t \in T} f_{s_I,t}\right)$.

## 3 Parametric Model Checking by SCC Decomposition

In this section we present our algorithmic approach to apply model checking to PDTMCs. Let $\mathcal{M} = (S, V, I, P)$ be a PDTMC with absorbing state set $T \subseteq S$. 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 parameters $V$ which describes the probability of reaching $t$ from $s_I$. We do this using *hierarchical graph decomposition*, inspired by a method for computing reachability probabilities in the non-parametric case [13].

### 3.1 PDTMC Abstraction

The basic concept of our model checking approach is to replace a non-absorbing subset $K \subseteq S$ of states and all transitions between them by transitions directly leading from the input states $\mathrm{Inp}(K)$ of $K$ to the output states $\mathrm{Out}(K)$ of $K$, carrying the accumulated probabilities of all paths between the given input and output states in $K$. This concept is illustrated in Figure 1: In 1(a), $K$ has one



(a) Initial PDTMC

(b) Abstraction of $K$ with abstract loop

(c) Abstraction of $K$

Figure 1: The concept of PDTMC abstraction

input state $s_{\text{I}}$ and two output states $s^1_{\text{out}}$, $s^2_{\text{out}}$. The abstraction in 1(c) hides every state of $K$ except for $s_{\text{I}}$; 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 *PDTMC induced by* $\mathcal{M}$ *and* $K$ is given by $\mathcal{M}^K = (S^K, V^K, I^K, P^K)$ with $S^K = K \cup \text{Out}(K)$, $V^K = V$, and for all $s, s' \in S^K$, $I^K(s) \neq 0 \iff s \in \text{Inp}(K)$ and

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

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 $I^K$ with the only constraint that $I^K(s) \neq 0$ iff $s$ is an input state of $K$.

*Example* 1. Consider the PDTMC $\mathcal{M}$ in Figure 2 and the state set $K = \{s_7, s_8\}$ with input states $\text{Inp}(K) = \{s_7\}$ and output states $\text{Out}(K) = \{s_5, s_6, s_9\}$. The PDTMC $\mathcal{M}^K = (S^K, V^K, I^K, P^K)$ induced by $\mathcal{M}$ and $K$ is shown in Figure 3(a).

Note that, since $K$ is non-absorbing, the probability of eventually reaching one of the output states is 1. The probability of reaching an output state $t$ from an input state $s$ is determined by the accumulated probability of all paths $\text{Paths}(s, t)$ from $s$ to $t$. Those paths are composed by a (possibly empty) prefix looping on $s$ and a postfix leading from $s$ to $t$ without returning back to $s$. In our abstraction this is reflected by representing the prefixes by an abstract self-loop on $s$ with probability $f_{s,s}$ and the postfixes by abstract transitions from the input states $s$ to the output states $t$ with probability $f_{s,t}$ (see Figure 1(b)). If all loops in $K$ are loops on $s$ then $f_{s,t}$ can be easily computed as the sum of the probabilities of all loop-free paths from $s$ to $t$. In the final abstraction shown in Figure 1(c), we make use of the fact that all paths from $s$ to $t$ can be extended with the same loops on $s$ as a prefix. Therefore we do not need to compute the probability of looping on $s$, but can scale the probabilities $f_{s,t}$ such that they sum up to 1.

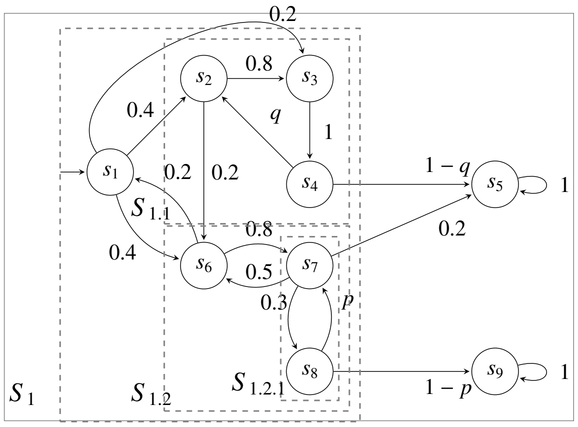
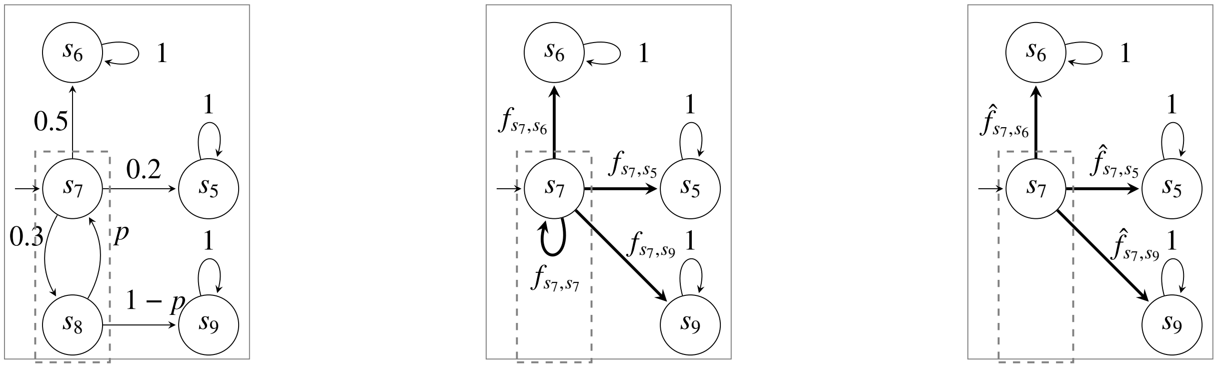


Figure 2: Example PDTMC and its SCC decomposition



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

Figure 3: PDTMC Abstraction

**Definition 8** (Abstract PDTMC)**.** Let $\mathcal{M} = (S, V, I, P)$ be a PDTMC with absorbing states $T \subseteq S$. The *abstract PDTMC* $\mathcal{M}_{\text{abs}} = (S_{\text{abs}}, V_{\text{abs}}, I_{\text{abs}}, P_{\text{abs}})$ is given by $S_{\text{abs}} = \{s \in S \mid I(s) \neq 0 \vee s \in T\}$, $V_{\text{abs}} = V$, and for all $s, s' \in S_{\text{abs}}$ we define $I_{\text{abs}}(s) = I(s)$ and

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

with

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

*Example* 2. Consider the PDTMC $\mathcal{M}' = (S', V', I', P')$ of Figure 3(a) with initial state $s_7$ and target states $T' = \{s_5, s_6, s_9\}$. The first abstraction step for the probabilities $p^{\mathcal{M}}_{\text{abs}}(s, s')$ is depicted in Figure 3(b) with the following probabilities:

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

The total probabilities of reaching the output states in $\mathcal{M}'_{\text{abs}}$ are 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: We divide each of these probabilities by the sum of all probabilities of outgoing edges, $f_{\text{out}} = 0.2 + 0.5 + 0.3 \cdot (1 - p) = 1 - 0.3p$.

Thus the abstract PDTMC $\mathcal{M}'_{\text{abs}} = (S'_{\text{abs}}, V'_{\text{abs}}, I'_{\text{abs}}, P'_{\text{abs}})$ depicted in Figure 3(c) has the states $S'_{\text{abs}} = \{s_5, s_6, s_7, s_9\}$ and edges from $s_7$ to all other states with the following probabilities:

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

**Theorem 1.** *Assume a PDTMC $\mathcal{M} = (S, V, I, P)$ with absorbing states $T \subseteq S$, and let $\mathcal{M}_{\text{abs}}$ be the abstraction of $\mathcal{M}$. Then for all $s_{\text{I}} \in S_{\text{I}}$ and $t \in T$ it holds that*

$$\text{Pr}^{\mathcal{M}}(\text{Paths}^{\mathcal{M}}(s_{\text{I}}, t)) = \text{Pr}^{\mathcal{M}_{\text{abs}}}(\text{Paths}^{\mathcal{M}_{\text{abs}}}(s_{\text{I}}, t)) .$$

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 8, while incoming transitions of the initial states of the abstraction as well as outgoing transitions of the absorbing states of the abstraction remain unmodified.

**Definition 9** (Substitution)**.** Assume a PDTMC $\mathcal{M} = (S, V, I, P)$, a non-absorbing set $K \subseteq S$ of states, the induced PDTMC $\mathcal{M}^K = (S^K, V^K, I^K, P^K)$ and the abstraction $\mathcal{M}^K_{\text{abs}} = (S^K_{\text{abs}}, V^K_{\text{abs}}, I^K_{\text{abs}}, P^K_{\text{abs}})$. The *substitution of $\mathcal{M}^K$ by its abstraction $\mathcal{M}^K_{\text{abs}}$ in $\mathcal{M}$* is given by $\mathcal{M}_{K\mapsto\text{abs}} = (S_{K\mapsto\text{abs}}, V_{K\mapsto\text{abs}}, I_{K\mapsto\text{abs}}, P_{K\mapsto\text{abs}})$ with $S_{K\mapsto\text{abs}} = (S \setminus K) \cup S^K_{\text{abs}}$, $V_{K\mapsto\text{abs}} = V$ and for all $s, s' \in S_{K\mapsto\text{abs}}$, $I_{K\mapsto\text{abs}}(s) = I(s)$ and

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

Due to Theorem 1, it directly follows that this substitution does not change reachability properties from the initial 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 $s_{\text{I}} \in S_{\text{I}}$ and absorbing states $t \in T$ that*

$$\text{Pr}^{\mathcal{M}}(\text{Paths}^{\mathcal{M}}(s_{\text{I}}, t)) = \text{Pr}^{\mathcal{M}_{K\mapsto\text{abs}}}(\text{Paths}^{\mathcal{M}_{K\mapsto\text{abs}}}(s_{\text{I}}, t)) .$$

**Algorithm 1** Model Checking PDTMCs

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

model_check(PDTMC $\mathcal{M} = (S, V, I, P)$, $T \subseteq \{t \in S \mid P(t,t) = 1\}$)
begin
    $\mathcal{M}_{\text{abs}} = (S_{\text{abs}}, V_{\text{abs}}, I_{\text{abs}}, P_{\text{abs}})$ := abstract($\mathcal{M}$)    (8)
    return $\sum_{s_I \in S_I} I(s_I) \cdot \Big(\sum_{t\in T} P_{\text{abs}}(s_I, t)\Big)$    (9)
end
```

## 3.2 Model Checking Parametric Markov Chains

In the previous section we gave the theoretical background for our model checking algorithm. Now we 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. Algorithmically, Tarjan's algorithm [14] is used to determine the SCC structure of the graph while we do not consider bottom SCCs. Sub-SCCs inside the SCCs without their input states are determined hierarchically, until no non-trivial sub-SCCs remain.

*Example* 3. In Figure 2, the dashed rectangles indicate the decomposition into the SCC $S_1 = \{1, 2, 3, 4, 6, 7, 8\}$ and the sub-SCSs $S_{1.1} = \{2, 3, 4\}$, $S_{1.2} = \{6, 7, 8\}$, and $S_{1.2.1} = \{7, 8\}$ with $S_{1.1} \subset S_1$ and $S_{1.2.1} \subset S_{1.2} \subset S_1$.

The general model checking algorithm is depicted in Algorithm 1. The recursive method *abstract*(PDTMC $\mathcal{M}$) computes the abstraction $\mathcal{M}_{\text{abs}}$ by iterating over all SCCs of the graph without the input states of $\mathcal{M}$ (Line 1). For each SCC $K$, the abstraction $\mathcal{M}^K_{\text{abs}}$ of the induced PDTMC $\mathcal{M}^K$ is computed by a recursive call (Line 2, Definitions 7,8). Afterwards, $\mathcal{M}^K$ is substituted by its abstraction in $\mathcal{M}$ (Line 3, Definition 9). Finally, the abstraction $\mathcal{M}_{\text{abs}}$ is computed and returned (Line 7, Definition 8). The method *abstract* is called by *model_check* (Line 8) which yields the abstract system $\mathcal{M}_{\text{abs}}$ where 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. The total probability is computed by building the sum of these transitions (Line 9).

For the computation of the abstract probabilities $p^{\mathcal{M}}_{\text{abs}}$, we distinguish the cases where the set $K$ has one or multiple input states.

**One input state** Consider a PDTMC $\mathcal{M}^K$ induced by $K$ with one initial state $s_\mathrm{I}$ and the set of absorbing states $T = \{t^1, \ldots, t^n\}$, such that $K \setminus \{s_\mathrm{I}\}$ has no non-trivial SCCs. If there is only one absorbing state, i. e., $n = 1$, we trivially have $p_{\mathrm{abs}}^{\mathcal{M}^K}(s_\mathrm{I}, t^1) = 1$. Otherwise we determine the probabilities $p_{\mathrm{abs}}^{\mathcal{M}^K}(s_\mathrm{I}, t^i)$ for all $1 \leq i \leq n$. As $K \setminus \{s_\mathrm{I}\}$ has no non-trivial SCSs, the set of those paths from $s_\mathrm{I}$ to $t^i$ that do not return to $s_\mathrm{I}$ consists of finitely many loop-free paths. The probability is computed recursively for all $s \in S^K$ by:

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

These probabilities can also be computed by direct or indirect methods for solving linear equation systems[2], see [15, Chapters 3,4], or state elimination as in [3].

The probabilities of the abstract PDTMC $\mathcal{M}_{\mathrm{abs}}^K = (S_{\mathrm{abs}}, V_{\mathrm{abs}}, I_{\mathrm{abs}}, P_{\mathrm{abs}})$ as in Definition 8 can now directly be computed, while an additional constraint is added in order to avoid divisions by zero:

$$P_{\mathrm{abs}}^{\mathcal{M}^K}(s_\mathrm{I}, t^i) = \begin{cases} \frac{p_{\mathrm{abs}}^{\mathcal{M}^K}(s_\mathrm{I}, t^i)}{\sum_{j=1}^n p_{\mathrm{abs}}^{\mathcal{M}^K}(s_\mathrm{I}, t^j)}, & \text{if } \sum_{j=1}^n p_{\mathrm{abs}}^{\mathcal{M}^K}(s_\mathrm{I}, t^j) \neq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

**Multiple input states** Given a PDTMC $\mathcal{M}^K$ with initial states $S_I = \{s_\mathrm{I}^1, \ldots, s_\mathrm{I}^m\}$ such that $I^K(s_\mathrm{I}^i) > 0$ for all $1 \leq i \leq m$ and absorbing states $T = \{t^1, \ldots, t^n\}$. The idea is 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 would be expensive in terms of both time and memory. Therefore, we first formulate the linear equation system as in Equation (1). All variables $p_{\mathrm{abs}}^{\mathcal{M}^K}(s, t^i)$ with $s \in K \setminus \mathrm{Inp}(K)$ are eliminated from the equation system. Then for each initial state $s_\mathrm{I}^i$ the equation system is solved separately by eliminating all variables $p_{\mathrm{abs}}^{\mathcal{M}^K}(s_\mathrm{I}^j, t^k)$, $j \neq i$.

Algorithm 1 returns the rational functions $P_{\mathrm{abs}}^{\mathcal{M}^K}(s_\mathrm{I}, t)$ for all $s_\mathrm{I} \in S_I$ and $t \in T$ as in Equation (2). 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 (2). These constraints can be evaluated by a *SAT-modulo- theories* (*SMT*) solver for non-linear real arithmetic [16]. 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

Both the SCC-based procedure as introduced in the last section as well as mere state-elimination [3] build rational functions representing reachability probabilities. These rational functions might

[2]Note that these equation systems are solved by keeping the parameters as constants.

grow rapidly in both algorithms and thereby form one of the major bottlenecks of this methodology. As already argued in [3], the best way to stem this blow-up is the cancellation of the rational functions in every computation step, which involves—apart from *addition*, *multiplication*, and *division* of rational functions as illustrated in Example 2—the rather expensive calculation of the *greatest common divisor* (gcd) of two polynomials.

In this section we present a new way of handling this problem: Additional maintenance and storage of (partial) polynomial factorizations can lead to remarkable speed-ups in the gcd computation, especially when dealing with symmetrically structured benchmarks where many similar polynomials occur. We present an optimized algorithm called gcd which *operates on the (partial) factorizations* of the polynomials to compute their gcd. During the calculations, the factorizations are also refined. On this account we reformulate the arithmetic operations on rational functions such that they preserve their numerator's and denominator's factorizations, if it is possible with reasonable effort.

**Factorizations.** In the following we assume that polynomials are *normalized*, that is they are of the form $g = a_1 \cdot x_1^{e_{1,1}} \cdot \ldots \cdot x_n^{e_{1,n}} + \cdots + a_m \cdot x_1^{e_{m,1}} \cdot \ldots \cdot x_n^{e_{m,n}}$ with $(e_{j,1}, \ldots, e_{j,n}) \neq (e_{k,1}, \ldots, e_{k,n})$ 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.

**Definition 10** (Factorization)**.** A *factorization* $\mathcal{F}_g = \{g_1^{e_1}, \ldots, g_n^{e_n}\}$ of a polynomial $g \neq 0$ is a non-empty set[3] of *factors* $g_i^{e_i}$, where the bases $g_i$ are pairwise different polynomials and the exponents are $e_i \in \mathbb{N}$ such that $g = \prod_{i=1}^{n} g_i^{e_i}$. We additionally set $\mathcal{F}_0 = \emptyset$.

For polynomials $g, h$ and a factorization $\mathcal{F}_g = \{g_1^{e_1}, \ldots, g_n^{e_n}\}$ of $g$ let $\text{bases}(\mathcal{F}_g) = \{g_1, \ldots, g_n\}$ and $\exp(h, \mathcal{F}_g)$ be $e_i$ if $g_i = h$ and 0 if $h \notin \text{bases}(\mathcal{F}_g)$. As the bases are not required to be irreducible, factorizations are not unique.

We assume that bases and exponents are non-zero, $\mathcal{F}_1 = \{1^1\}$, 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 $\mathcal{F}_g^{\text{red}} = \{1^1\}$ if $n > 0$ and $g_i = 1$ or $e_i = 0$ for all $1 \leq i \leq n$, and $\mathcal{F}_g^{\text{red}} = \mathcal{F}_g \setminus \{g_i^{e_i} \mid g_i = 1 \vee e_i = 0\}$ otherwise.

**Operations on factorizations.** 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 $\cdot_{\mathcal{F}}$, $:_{\mathcal{F}}$ and $+_{\mathcal{F}}$ correspond to multiplication, division[4] and addition, respectively. Due to space limitations, we

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

[4]$\mathcal{F}_{g_1} :_{\mathcal{F}} \mathcal{F}_{g_2}$ 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 $g_1$.

omit in the remaining of this paper the trivial cases involving $\mathcal{F}_0$. Therefore we define

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

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 4 illustrates the application of the above operations.

**Operations on rational functions.** We represent a rational function $\frac{g_1}{g_2}$ by separate factorizations $\mathcal{F}_{g_1}$ and $\mathcal{F}_{g_2}$ for the numerator $g_1$ and the denominator $g_2$, respectively. For multiplication $\frac{g_1}{g_2} = \frac{h_1}{h_2} \cdot \frac{q_1}{q_2}$, we compute $\mathcal{F}_{g_1} = \mathcal{F}_{h_1} \cdot_{\mathcal{F}} \mathcal{F}_{q_1}$ and $\mathcal{F}_{g_2} = \mathcal{F}_{h_2} \cdot_{\mathcal{F}} \mathcal{F}_{q_2}$. Division is reduced to multiplication according to $\frac{h_1}{h_2} : \frac{q_1}{q_2} = \frac{h_1}{h_2} \cdot \frac{q_2}{q_1}$.

For the addition $\frac{g_1}{g_2} = \frac{h_1}{h_2} + \frac{q_1}{q_2}$, we compute $g_2$ with $\mathcal{F}_{g_2} = \mathcal{F}_{h_2} \cup_{\mathcal{F}} \mathcal{F}_{q_2}$ as a common multiple of $h_2$ and $q_2$, such that $g_2 = h_2 \cdot h_2'$ with $\mathcal{F}_{h_2'} = \mathcal{F}_{g_2} :_{\mathcal{F}} \mathcal{F}_{h_2}$, and $g_2 = q_2 \cdot q_2'$ with $\mathcal{F}_{q_2'} = \mathcal{F}_{g_2} :_{\mathcal{F}} \mathcal{F}_{q_2}$. For the numerator $g_1$ we first determine a common divisor $d$ of $h_1$ and $q_1$ by $\mathcal{F}_d = \mathcal{F}_{h_1} \cap_{\mathcal{F}} \mathcal{F}_{q_1}$, such that $h_1 = d \cdot h_1'$ with $\mathcal{F}_{h_1'} = \mathcal{F}_{h_1} :_{\mathcal{F}} \mathcal{F}_d$, and $q_1 = d \cdot q_1'$ with $\mathcal{F}_{q_1'} = \mathcal{F}_{q_1} :_{\mathcal{F}} \mathcal{F}_d$. The numerator $g_1$ is $d \cdot (h_1' \cdot h_2' + q_1' \cdot q_2')$ with factorization $\mathcal{F}_d \cdot_{\mathcal{F}} (\mathcal{F}_{h_1'} \cdot_{\mathcal{F}} \mathcal{F}_{h_2'} +_{\mathcal{F}} \mathcal{F}_{q_1'} \cdot_{\mathcal{F}} \mathcal{F}_{q_2'})$.

The rational function $\frac{g_1}{g_2}$ resulting from the addition is further simplified by cancellation, i. e., dividing $g_1$ and $g_2$ by their greatest common divisor (gcd) $g$. Given the factorizations $\mathcal{F}_{g_1}$ and $\mathcal{F}_{g_2}$, Algorithm 2 calculates the factorizations $\mathcal{F}_g$, $\mathcal{F}_{\frac{g_1}{g}}$, and $\mathcal{F}_{\frac{g_2}{g}}$.

Intuitively, the algorithm maintains the fact that $G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F_1'$ is a factorization of $g_1$, where $G$ contains common factors of $g_1$ and $g_2$, $F_1$ is going to be checked whether it contains further common factors, and $F_1'$ does not contain any common factors. In the outer while-loop, an element $r_1^{e_1}$ to be checked is taken from $F_1$. In the inner while-loop, a factorization $G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F_2'$ of $g_2$ is maintained such that $F_2'$ does not contain any common factors with $r_1$, and $F_2$ is still to be checked.

Now we explain the algorithm in more detail. Initially, a factorization $G$ of a common divisor of $g_1$ and $g_2$ is set to $\mathcal{F}_{g_1} \cap_{\mathcal{F}} \mathcal{F}_{g_2}$ (Line 1). The remaining factors of $g_1$ and $g_2$ are stored in $F_1$ resp. $F_2$. The sets $F_1'$ and $F_2'$ contain factors of $g_1$ resp. $g_2$ whose greatest common divisor is 1 (Line 2). 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 using standard gcd computation for polynomials (Line 7). Note that the main concern of Algorithm 2 is to avoid the application of this expensive operation as far as possible and to apply it to preferably simple polynomials otherwise. 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 $g := \gcd(r_1, r_2) = 1$ are found, we just shift $r_2^{e_2}$ from $F_2$ to $F_2'$ (Line 10). 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.

**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** $g :=$ 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)
        $F_2 := F_2 \cdot_{\mathcal{F}} F'_2$                                                          (19)
        $F'_2 := \{1^1\}$                                                                              (20)
    **end while**                                                                                      (21)
    **return** $(F'_1, F_2, G)$                                                                        (22)
**end**
```

$g^{e_2-\min(e_1,e_2)}$ (Line 12-15). 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 $F_2$ (Line 18-20). 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* 4. 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 $(\{(z)^1\},\ \{(1)^1\},\ \{(x)^1,\ (y)^1\})$. Using these results, we can also refine $\mathcal{F}_{xyz} = F'_1 \cdot_{\mathcal{F}} G = \{(x)^1,\ (y)^1,\ (z)^1\}$ and $\mathcal{F}_{xy} = F_2 \cdot_{\mathcal{F}} G = \{(x)^1,\ (y)^1\}$.

**Theorem 2.** *Let $p_1$ and $p_2$ be two polynomials with factorizations $\mathcal{F}_{p_1}$ resp. $\mathcal{F}_{p_2}$. 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. $\frac{p_2}{g}$.*

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 [17]. The prototype is available on the project homepage[5]. Moreover, we implemented the state-elimination approach 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 14 hours (50400 seconds) and a 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) [18] 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) [19] is designed for anonymous network communication using random routing, parametrized in how many members are "good" or "bad" and the probability if a good member delivers a message or randomly routes it to another member. *NAND multiplexing* (NAND) [20] 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

[5]http://goo.gl/nS378q

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 the PARAM tool itself.[6] 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 list the number of polynomials which are intermediately stored.

| | Graph | | SCC MC | | | STATE ELIM | | | PARAM | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | States | Trans. | Time | Poly | Mem | Time | Poly | Mem | Time | Mem |
| BRP | 3528 | 4611 | 29.05 | 3283 | 48.10 | **4.33** | 8179 | 61.17 | 98.99 | 32.90 |
| BRP | 4361 | 5763 | 511.50 | 4247 | 501.71 | **6.87** | 9520 | 78.49 | 191.52 | 58.43 |
| BRP | 7048 | 9219 | 548.73 | 6547 | 281.86 | **25.05** | 16435 | 216.05 | 988.28 | 142.66 |
| BRP | 10759 | 13827 | 147.31 | 9231 | 176.89 | **85.54** | 26807 | 682.24 | 3511.96 | 304.07 |
| BRP | 21511 | 27651 | 1602.53 | 18443 | 776.48 | **718.66** | 53687 | 3134.59 | 34322.60 | 1757.12 |
| CROWDS | 198201 | 348349 | **60.90** | 13483 | 140.15 | 243.07 | 27340 | 133.91 | 46380.00 | 227.66 |
| CROWDS | 482979 | 728677 | **35.06** | 35916 | 478.85 | 247.75 | 65966 | 297.40 | TO | — |
| CROWDS | 726379 | 1283297 | **223.24** | 36649 | 515.61 | 1632.63 | 73704 | 477.10 | TO | — |
| CROWDS | 961499 | 1452537 | **81.88** | 61299 | 1027.78 | 646.76 | 112452 | 589.21 | TO | — |
| CROWDS | 1729494 | 2615272 | **172.59** | 97655 | 2372.35 | 1515.63 | 178885 | 1063.15 | TO | — |
| CROWDS | 2888763 | 5127151 | **852.76** | 110078 | 2345.06 | 12326.80 | 224747 | 2123.96 | TO | — |
| NAND | 7393 | 11207 | 8.35 | 15688 | 114.60 | 17.02 | 140057 | 255.13 | **5.00** | 10.67 |
| NAND | 14323 | 21567 | 39.71 | 25504 | 366.79 | 59.60 | 405069 | 926.33 | **15.26** | 16.89 |
| NAND | 21253 | 31927 | 100.32 | 35151 | 795.31 | 121.40 | 665584 | 2050.67 | **29.51** | 24.45 |
| NAND | 28183 | 42287 | 208.41 | 44799 | 1405.16 | 218.85 | 925324 | 3708.27 | **50.45** | 30.47 |
| NAND | 78334 | 121512 | **639.29** | 184799 | 3785.11 | — | — | MO | 1138.82 | 111.58 |

For BRP, STATE ELIM always outperforms PARAM and SCC MC by up to two orders of magnitude. On larger instances, SCC MC is faster than PARAM while on smaller ones PARAM is faster and has a smaller memory consumption.

In contrast, the crowds protocol always 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. This is actually measured by the timeout; using PARAM we could not retrieve results for larger instances.

To give an example where PARAM performs mostly better than our approaches, we consider NAND. Its graph is acyclic consisting mainly of single paths leading to states that have a high number of outgoing edges, i. e., many paths join at these states and diverge again. Together with a large number of different probabilities, this involves the addition of many polynomials, whose factorizations are completely stored. The SCC approach performs better here, as for acyclic graphs just the linear equation system is solved, as described in Section 3. This seems to be superior to the state elimination as implemented in our tool. We don't know about PARAM's interior for these special cases. As a solution, our implementation offers the possibility to limit the number of stored polynomials, which decreases the memory consumption at the price of losing information about the factorizations. However, an efficient strategy to manage this bounded pool of polynomials is not yet implemented. Therefore, we refrain from presenting experimental results

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

for this scenario.

## 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 [21] 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.

## Bibliography

[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] Kwiatkowska, M., Norman, G., Parker, D.: Prism 4.0: Verification of probabilistic real-time systems. In: Proc. of CAV. Volume 6806 of LNCS, Springer (2011) 585–591

[8] Han, T., Katoen, J.P., Mereacre, A.: Approximate parameter synthesis for probabilistic time-bounded reachability. In: Proc. of RTSS, IEEE CS (2008) 173–182

[9] 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

[10] 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

[11] Jansen, N., Corzilius, F., Volk, M., Wimmer, R., Ábrahám, E., Katoen, J.P., Becker, B.: Accelerating parametric probabilistic verification. CoRR **abs/1312.3979** (2013)

[12] Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press (2008)

[13] 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

[14] Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM Journal on Computing **1**(2) (1972) 146–160

[15] Quarteroni, A., Sacco, R., Saleri, F.: Numerical Mathematics. Springer (2000)

[16] Jovanovic, D., de Moura, L.M.: Solving non-linear arithmetic. In: Proc. of IJCAR. Volume 7364 of LNCS, Springer (2012) 339–354

[17] 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

[18] Helmink, L., Sellink, M., Vaandrager, F.: Proof-checking a data link protocol. In: Proc. of TYPES. Volume 806 of LNCS, Springer (1994) 127–165

[19] Reiter, M.K., Rubin, A.D.: Crowds: Anonymity for web transactions. ACM Trans. on Information and System Security **1**(1) (1998) 66–92

[20] Han, J., Jonker, P.: A system architecture solution for unreliable nanoelectronic devices. IEEE Transactions on Nanotechnology **1** (2002) 201–208

[21] 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

**Theorem 1** *Assume a PDTMC $\mathcal{M} = (S, V, I, P)$ with absorbing states $T \subseteq S$, and let $\mathcal{M}_{\text{abs}}$ be the abstraction of $\mathcal{M}$. Then for all $s_{\text{I}} \in S_{\text{I}}$ and $t \in T$ it holds that*

$$\Pr^{\mathcal{M}}(\text{Paths}^{\mathcal{M}}(s_{\text{I}}, t)) = \Pr^{\mathcal{M}_{\text{abs}}}(\text{Paths}^{\mathcal{M}_{\text{abs}}}(s_{\text{I}}, t)) \, .$$

*Proof.* First note that all initial states and absorbing states in $\mathcal{M}$ are also states of the abstraction.

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

$$p^{\mathcal{M}}_{\text{abs}}(s_{\text{I}}, s_{\text{I}}) = 1 - \sum_{t \in T} p^{\mathcal{M}}_{\text{abs}}(s_{\text{I}}, t). \tag{3}$$

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

$$R_{\text{loop}} = \{s_{\text{I}} s_1 \ldots s_n s_{\text{I}} \in \text{Paths}^{\mathcal{M}} \mid s_i \notin \{s_{\text{I}}\} \cup T, 1 \leq i \leq n\} \tag{4}$$

$$R_{\text{out}} = \{s_{\text{I}} s_1 \ldots s_n t \in \text{Paths}^{\mathcal{M}} \mid s_i \notin \{s_{\text{I}}\} \cup T, 1 \leq i \leq n, t \in T\} \tag{5}$$

As the self-loop on $s_{\text{I}}$ represents the paths of $R_{\text{loop}}$, it holds that

$$p^{\mathcal{M}}_{\text{abs}}(s_{\text{I}}, s_{\text{I}}) = \Pr(R_{\text{loop}}). \tag{6}$$

We now have:

$$\begin{aligned}
& \Pr^{\mathcal{M}}(\text{Paths}^{\mathcal{M}}(s_{\text{I}}, t)) \\
= \; & \Pr^{\mathcal{M}}\Big(\bigcup_{i=0}^{\infty} \{\pi_1 \cdot \cdots \cdot \pi_i \cdot \pi_{\text{out}} \mid \pi_j \in R_{\text{loop}}, 1 \leq j \leq i;\ \pi_{\text{out}} \in R_{\text{out}}\}\Big) \\
= \; & \sum_{i=0}^{\infty} \Pr^{\mathcal{M}}(\{\pi_1 \cdot \cdots \cdot \pi_i \cdot \pi_{\text{out}} \mid \pi_j \in R_{\text{loop}}, 1 \leq j \leq i;\ \pi_{\text{out}} \in R_{\text{out}}\}) \\
= \; & \sum_{i=0}^{\infty} (\Pr^{\mathcal{M}}(R_{\text{loop}}))^i \cdot \Pr^{\mathcal{M}}(R_{\text{out}}) \\
= \; & \sum_{i=0}^{\infty} (p^{\mathcal{M}}_{\text{abs}}(s_{\text{I}}, s_{\text{I}}))^i \cdot \Pr^{\mathcal{M}}(R_{\text{out}}) \quad \text{(Equation (6))} \\
= \; & \frac{1}{1 - p^{\mathcal{M}}_{\text{abs}}(s_{\text{I}}, s_{\text{I}})} \cdot \Pr^{\mathcal{M}}(R_{\text{out}}) \quad \text{(Geometric Series)} \\
= \; & \frac{1}{\sum_{s_{\text{out}} \in T} p^{\mathcal{M}}_{\text{abs}}(s_{\text{I}}, s_{\text{out}})} \cdot \Pr^{\mathcal{M}}(R_{\text{out}}) \quad \text{(Equation (3))}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{\sum\limits_{s_{\text{out}} \in T} p^{\mathcal{M}}_{\text{abs}}(s_{\text{I}}, s_{\text{out}})} \cdot p^{\mathcal{M}}_{\text{abs}}(s_{\text{I}}, t) \quad \text{(Definition 8)} \\
&= P_{\text{abs}}(s_{\text{I}}, t) \quad \text{(Definition 8)} \\
&= \Pr^{\mathcal{M}_{\text{abs}}}(\text{Paths}^{\mathcal{M}_{\text{abs}}}(s_{\text{I}}, t)) \, .
\end{aligned}
$$

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

**Theorem 2** *Let $p_1$ and $p_2$ be two polynomials with factorizations $\mathcal{F}_{p_1}$ resp. $\mathcal{F}_{p_2}$. Applying Algorithm 2 to these factorizations results in* $\text{gcd}(\mathcal{F}_{p_1}, \mathcal{F}_{p_2}) = (\mathcal{F}_{r_1}, \mathcal{F}_{r_2}, G)$ *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 denote 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 standard greatest common divisor computation for polynomials by gcd.

We define the following Hoare-style assertion network:

```
**GCD**(factorization $\mathcal{F}_{g_1}$, factorization $\mathcal{F}_{g_2}$)
**begin**
{$true$}                                                                                     (1)
    $G := (\mathcal{F}_{g_1} \cap_{\mathcal{F}} \mathcal{F}_{g_2})$                                                          (2)
{$G = \mathcal{F}_{g_1} \cap_{\mathcal{F}} \mathcal{F}_{g_2}$}                                                                        (3)
    $F_i := \mathcal{F}_{g_i} :_{\mathcal{F}} G$ and $F'_i := \{1^1\}$ for $i = 1, 2$                                           (4)
{$\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F'_2 \wedge \mathcal{P}(F'_1) = 1 \wedge \mathcal{P}(F'_2) = 1$}                    (5)
    **while** exists $r_1^{e_1} \in F_1$ with $r_1 \neq 1$ **do**                                              (6)
{$\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F'_2 \wedge$
 $\text{gcd}(\mathcal{P}(F'_1), \mathcal{P}(F_2 \cdot_{\mathcal{F}} F'_2)) = 1 \wedge \text{gcd}(r_1^{e_1}, \mathcal{P}(F'_2)) = 1 \wedge r_1^{e_1} \in F_1$}                          (7)
        $F_1 := F_1 \setminus \{r_1^{e_1}\}$                                                             (8)
{$\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \cdot_{\mathcal{F}} \{r_1^{e_1}\} \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F'_2 \wedge$
 $\text{gcd}(\mathcal{P}(F'_1), \mathcal{P}(F_2 \cdot_{\mathcal{F}} F'_2)) = 1 \wedge \text{gcd}(r_1^{e_1}, \mathcal{P}(F'_2)) = 1$}                                        (9)
        **while** $r_1 \neq 1$ and exists $r_2^{e_2} \in F_2$ with $r_2 \neq 1$ **do**                            (10)
{$\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \cdot_{\mathcal{F}} \{r_1^{e_1}\} \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F'_2 \wedge$
 $\text{gcd}(\mathcal{P}(F'_1), \mathcal{P}(F_2 \cdot_{\mathcal{F}} F'_2)) = 1 \wedge \text{gcd}(r_1^{e_1}, \mathcal{P}(F'_2)) = 1 \wedge r_2^{e_2} \in F_2$}                          (11)
            $F_2 := F_2 \setminus \{r_2^{e_2}\}$                                                         (12)
{$\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \cdot_{\mathcal{F}} \{r_1^{e_1}\} \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F'_2 \cdot_{\mathcal{F}} \{r_2^{e_2}\} \wedge$
 $\text{gcd}(\mathcal{P}(F'_1), \mathcal{P}(F_2 \cdot_{\mathcal{F}} F'_2 \cdot_{\mathcal{F}} \{r_2^{e_2}\})) = 1 \wedge \text{gcd}(r_1^{e_1}, \mathcal{P}(F'_2)) = 1$}                             (13)
            **if** $\neg\text{irreducible}(r_1) \vee \neg\text{irreducible}(r_2)$ **then** $g := \text{gcd}(r_1, r_2)$             (14)
            **else** $g := 1$                                                                     (15)
{$\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \cdot_{\mathcal{F}} \{r_1^{e_1}\} \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F'_2 \cdot_{\mathcal{F}} \{r_2^{e_2}\} \wedge$
 $\text{gcd}(\mathcal{P}(F'_1), \mathcal{P}(F_2 \cdot_{\mathcal{F}} F'_2 \cdot_{\mathcal{F}} \{r_2^{e_2}\})) = 1 \wedge \text{gcd}(r_1^{e_1}, \mathcal{P}(F'_2)) = 1 \wedge g = \text{gcd}(r_1, r_2)$}          (16)
            **if** $g = 1$ **then**                                                               (17)
{$\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \cdot_{\mathcal{F}} \{r_1^{e_1}\} \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F'_2 \cdot_{\mathcal{F}} \{r_2^{e_2}\} \wedge$
```

```
$\gcd(\mathcal{P}(F'_1), \mathcal{P}(F_2 \cdot_{\mathcal{F}} F'_2 \cdot_{\mathcal{F}} \{r_2^{e_2}\})) = 1 \wedge \gcd(r_1^{e_1}, \mathcal{P}(F'_2)) = 1 \wedge \gcd(r_1, r_2) = 1\}$   (18)
                $F'_2 := F'_2 \cdot_{\mathcal{F}} \{r_2^{e_2}\}$   (19)
$\{\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \cdot_{\mathcal{F}} \{r_1^{e_1}\} \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F'_2 \wedge$
$\gcd(\mathcal{P}(F'_1), \mathcal{P}(F_2 \cdot_{\mathcal{F}} F'_2)) = 1 \wedge \gcd(r_1^{e_1}, \mathcal{P}(F'_2)) = 1\}$   (20)
            **else**   (21)
$\{\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \cdot_{\mathcal{F}} \{r_1^{e_1}\} \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F'_2 \cdot_{\mathcal{F}} \{r_2^{e_2}\} \wedge$
$\gcd(\mathcal{P}(F'_1), \mathcal{P}(F_2 \cdot_{\mathcal{F}} F'_2 \cdot_{\mathcal{F}} \{r_2^{e_2}\})) = 1 \wedge \gcd(r_1^{e_1}, \mathcal{P}(F'_2)) = 1 \wedge g = \gcd(r_1, r_2)\}$   (22)
                $r_1 := \frac{r_1}{g}$   (23)
$\{\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \cdot_{\mathcal{F}} \{(r_1 \cdot g)^{e_1}\} \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F'_2 \cdot_{\mathcal{F}} \{r_2^{e_2}\} \wedge$
$\gcd(\mathcal{P}(F'_1), \mathcal{P}(F_2 \cdot_{\mathcal{F}} F'_2 \cdot_{\mathcal{F}} \{r_2^{e_2}\})) = 1 \wedge \gcd((r_1 \cdot g)^{e_1}, \mathcal{P}(F'_2)) = 1 \wedge g = \gcd((r_1 \cdot g), r_2)\}$   (24)
                $F_i := F_i \cdot_{\mathcal{F}} \{g^{e_i - \min(e_1, e_2)}\}$ for $i = 1, 2$   (25)
$\{\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \cdot_{\mathcal{F}} \{r_1^{e_1}, g^{\min(e_1,e_2)}\} \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F'_2 \cdot_{\mathcal{F}} \{(\frac{r_2}{g})^{e_2}, g^{\min(e_1,e_2)}\} \wedge$
$\gcd(\mathcal{P}(F'_1), \mathcal{P}(F_2 \cdot_{\mathcal{F}} F'_2 \cdot_{\mathcal{F}} \{(\frac{r_2}{g})^{e_2}, g^{\min(e_1,e_2)}\})) = 1 \wedge \gcd((r_1 \cdot g)^{e_1}, \mathcal{P}(F'_2)) = 1 \wedge$
$g = \gcd((r_1 \cdot g), r_2)\}$   (26)
                $F'_2 := F'_2 \cdot_{\mathcal{F}} \{(\frac{r_2}{g})^{e_2}\}$   (27)
$\{\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \cdot_{\mathcal{F}} \{r_1^{e_1}, g^{\min(e_1,e_2)}\} \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F'_2 \cdot_{\mathcal{F}} \{g^{\min(e_1,e_2)}\} \wedge$
$\gcd(\mathcal{P}(F'_1), \mathcal{P}(F_2 \cdot_{\mathcal{F}} F'_2 \cdot_{\mathcal{F}} \{g^{\min(e_1,e_2)}\})) = 1 \wedge \gcd((r_1 \cdot g)^{e_1}, \mathcal{P}(F'_2)) = 1\}$   (28)
                $G := G \cdot_{\mathcal{F}} \{g^{\min(e_1,e_2)}\}$   (29)
$\{\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \cdot_{\mathcal{F}} \{r_1^{e_1}\} \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F'_2 \wedge$
$\gcd(\mathcal{P}(F'_1), \mathcal{P}(F_2 \cdot_{\mathcal{F}} F'_2)) = 1 \wedge \gcd(r_1^{e_1}, \mathcal{P}(F'_2)) = 1\}$   (30)
            **end if**   (31)
$\{\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \cdot_{\mathcal{F}} \{r_1^{e_1}\} \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F'_2 \wedge$
$\gcd(\mathcal{P}(F'_1), \mathcal{P}(F_2 \cdot_{\mathcal{F}} F'_2)) = 1 \wedge \gcd(r_1^{e_1}, \mathcal{P}(F'_2)) = 1\}$   (32)
        **end while**   (33)
$\{\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \cdot_{\mathcal{F}} \{r_1^{e_1}\} \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F'_2 \wedge$
$\gcd(\mathcal{P}(F'_1), \mathcal{P}(F_2 \cdot_{\mathcal{F}} F'_2)) = 1 \wedge \gcd(r_1^{e_1}, \mathcal{P}(F'_2)) = 1 \wedge (r_1 = 1 \vee \mathcal{P}(F_2) = 1)$   (34)
        $F'_1 := F'_1 \cdot_{\mathcal{F}} \{r_1^{e_1}\}$   (35)
$\{\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \cdot_{\mathcal{F}} F'_2 \wedge \gcd(\mathcal{P}(F'_1), \mathcal{P}(F_2 \cdot_{\mathcal{F}} F'_2)) = 1\}$   (36)
        $F_2 := F_2 \cdot_{\mathcal{F}} F'_2$   (37)
$\{\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \wedge \gcd(\mathcal{P}(F'_1), \mathcal{P}(F_2)) = 1\}$   (38)
        $F'_2 := \{1^1\}$   (39)
$\{\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F_1 \cdot_{\mathcal{F}} F'_1 \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \wedge \gcd(\mathcal{P}(F'_1), \mathcal{P}(F_2)) = 1 \wedge \mathcal{P}(F'_2) = 1\}$   (40)
    **end while**   (41)
$\{\mathcal{F}_{g_1} = G \cdot_{\mathcal{F}} F'_1 \wedge \mathcal{F}_{g_2} = G \cdot_{\mathcal{F}} F_2 \wedge \gcd(\mathcal{P}(F'_1), \mathcal{P}(F_2)) = 1\}$   (42)
    **return** $(F'_1, F_2, G)$   (43)
**end**
```

The above assertion network is inductive.

- For the assignments, their preconditions imply their postconditions after substituting the assigned expression for the assigned variables. (For simplicity, we handle the first if-then-else statement in lines (14)-(15) also as atomic assignment.)

- For the if-then-else statement in lines (17)-(31), its precondition (16) implies the precondition (18) of the if-branch if the branching condition holds, and the precondition (22) of the else-branch if the condition does not hold. The postconditions (20) and (30) of both branches imply the postcondition (32) of the if-then-else statement.

- For the outer while-loop (6)-(41), its precondition (5) as well as the postcondition (40) of its body imply the precondition (7) of the body if the loop condition holds, and they both imply the postcondition (42) of the while loop if the loop condition does not hold.
- The inner while loop's inductivity can be shown similarly.

That means, the assertion (42) always holds before returning, implying the correctness of the algorithm.

The algorithm is also complete, since it always terminates: We can use as ranking function the sum of the degrees of all polynomials in $F_1$ for the outer loop and in $F_2$ for the inner loop to show their termination.

$\square$

## MEALS Partner Abbreviations

**SAU:** Saarland University, D

**RWT:** RWTH Aachen University, D

**TUD:** Technische Universität Dresden, D

**INR:** Institut National de Recherche en Informatique et en Automatique, FR

**IMP:** Imperial College of Science, Technology and Medicine, UK

**ULEIC:** University of Leicester, UK

**TUE:** Technische Universiteit Eindhoven, NL

**UNC:** Universidad Nacional de Córdoba, AR

**UBA:** Universidad de Buenos Aires, AR

**UNR:** Universidad Nacional de Río Cuarto, AR

**ITBA:** Instituto Técnológico Buenos Aires, AR