Verification and Synthesis for Parametric Markov Chains

Bachelor Thesis

Matthias Volk

Theory of Hybrid Systems
RWTH Aachen University

Supervisor:
Prof. Dr. Erika Ábrahám

2nd Supervisor:
Prof. Dr. Ir. Joost-Pieter Katoen

Advisors:
Dipl. Inform. Florian Corzilius,
Dipl. Inform. Nils Jansen

Aachen, September 2012

Erklärung

Hiermit versichere ich, dass ich die vorgelegte Arbeit selbstständig verfasst und noch nicht anderweitig zu Prüfungszwecken vorgelegt habe. Alle benutzten Quellen und Hilfsmittel sind angegeben, wörtliche und sinngemäße Zitate wurden als solche gekennzeichnet.

Matthias Volk
Aachen, den 7. September 2012

Abstract

In this thesis we present a new approach for model checking Parametric Markov Chains (PMCs). In PMCs certain transition probabilities are left open by introducing parameters instead of concrete values. Model checking PMCs for unbounded reachability probabilities then yields a rational function as a probability bound for reaching the set of target states. These rational functions can be instantiated for a range of parameter values or analyzed further. Our algorithm extends an existing approach for model checking DTMCs and is implemented in the tool COMICS. Further, we developed our own data structure to simplify rational functions more efficiently. We demonstrated the applicability of our methods on a number of case studies.

Contents

1 Introduction

Modeling and analysis of stochastic processes is mostly achieved by using *Markov Chains*. The underlying graph structure of such a Markov Chain is a good visualization of the stochastic model and consists of states and transitions where every transition has a probability. Thus, for every finite run on states of the graph we multiply the corresponding transition probabilities and get a resulting probability for this run. Considering the set of all finite runs, we are able to compute the probability of reaching a specific target state by starting at an initial state. Determining these probabilities is the task of *Unbounded Reachability Analysis*, where we are interested in the total probability of eventually reaching one state from a set of target states.

Let us consider the example of two cowboys, "Lucky Luke" and "Joe Dalton", dueling in the Wild West at High Noon similar to the one in [8]. Both cowboys have a gun and try to shoot their opponent. However, there is a certain probability of missing the target. For example Luke hits his target in nearly every case and therefore has a probability of only 0.1 to miss his opponent. Joe, on the other hand, is often a bit nervous and hence misses the target with probability 0.4. As Luke shoots faster than his shadow he has the first shot in 3 of 4 duels. After that, the two shoot alternating. As nearly everything is in disfavor against Joe, we are interested in the probability of him surviving the duel, i. e., Joe hitting Luke first.

This scenario can be modeled by use of a Markov Chain as depicted in Figure 1.1.

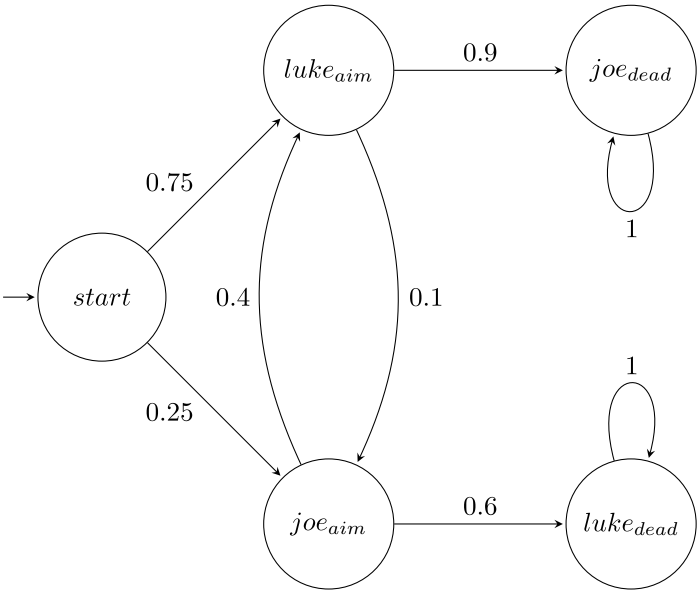


Figure 1.1: Duel between two cowboys

Starting in the *start*-state, with probability 0.25 we reach the state joe_{aim} indicating that Joe Dalton has the first shot. Then he hits Luke with a probability of 0.6 modeled by the transition to the target state $luke_{dead}$, where we can only take the self-loop. Multiplying every probability along this path we get $0.25 \cdot 0.6 = 0.15$ which is the probability of the event "Joe starts and kills Luke with his first shot". However, this is not the only possible event, where Joe survives the duel. For example, both cowboys could miss their first shot and Joe kills Luke with his second shot.

Therefore we have to consider every finite run in this graph, which starts in *start* and eventually reaches $luke_{dead}$.

This problem can be solved using *model checking*. Several different algorithms were developed for model checking Markov Chains. For example, matrix-vector multiplication is used in the tools PRISM [18] and MRMC [17]. In this thesis we use the SCC-based model checking approach presented in [1]. Their idea is to abstract underlying graph structures, mainly strongly connected subsets. Intuitively, they search for cycles in the graph and then abstract these cycles in a bottom-up manner. Thus, they eliminate all cycles in the graph but preserve the reachability property, i. e., no relevant information is lost. Applying this algorithm to our example, we get the resulting graph depicted in Figure 1.2.

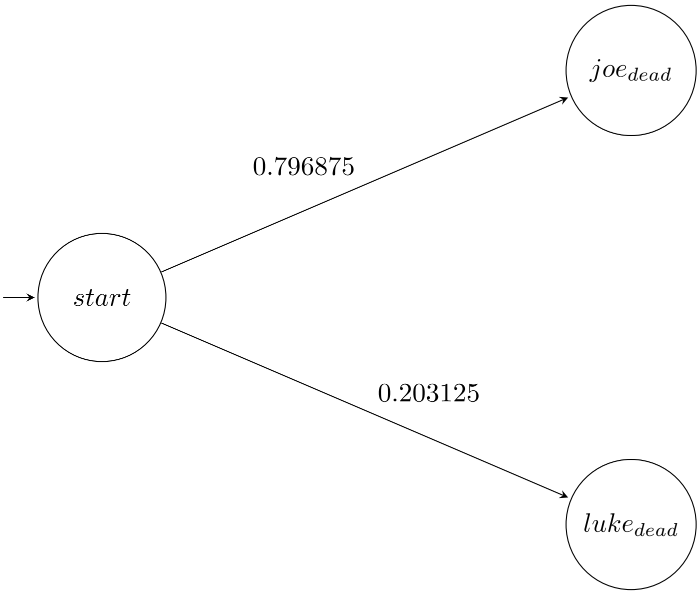


Figure 1.2: Result after model checking

Now we have the probabilities of surviving the duel for both cowboys. Joe only survives the duel in approximately 1 of 5 cases and therefore he should maybe prefer to yield.

In this example we had specific probabilities for the hit rate of both cowboys. However, we could also leave these probabilities open and analyze the surviving rate of Joe according to different hit rates. Thus, we introduce two parameters p_{luke} and p_{joe} for the probability of both cowboys to miss the target. Then we get a *Parametric Markov Chain* as visualized in Figure 1.3.

We show in this thesis that the SCC-based model checking algorithm for Markov Chains can be extended to handle also Parametric Markov Chains. Applying this adapted algorithm to our cowboy example, we get the result show in Figure 1.4.

Now we want Joe to survive the duel in at least 3 of 4 cases and are interested in the conditions for the miss rates of both cowboys to ensure Joe's surviving. Therefore we require the probability

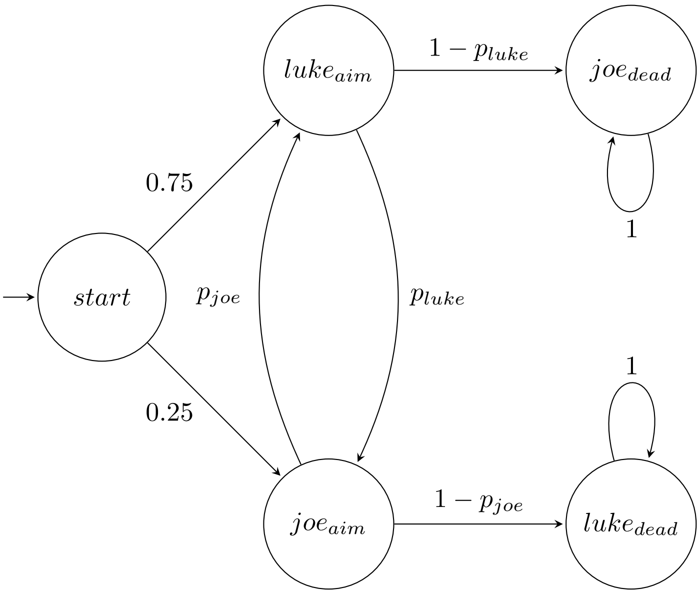


Figure 1.3: Duel between two cowboys with parametric hit rates

of reaching the state $luke_{dead}$ to be at least 0.75 and solve the following inequation:

$$\begin{aligned}
& \frac{0.75p_{luke} - 0.25p_{joe} - 0.75p_{luke}p_{joe} + 0.25}{1 - p_{luke}p_{joe}} \geq 0.75 \\
\Longleftrightarrow\ & \frac{0.75p_{luke} - 0.25p_{joe} - 0.75p_{luke}p_{joe} + 0.25}{1 - p_{luke}p_{joe}} \geq \frac{0.75 - 0.75p_{luke}p_{joe}}{1 - p_{luke}p_{joe}} \\
\Longleftrightarrow\ & \frac{0.75p_{luke} - 0.25p_{joe} - 0.5}{1 - p_{luke}p_{joe}} \geq 0 \\
\Longleftrightarrow\ & 0.75p_{luke} - 0.25p_{joe} - 0.5 \geq 0 \\
\Longleftrightarrow\ & p_{luke} \geq \frac{1}{3}p_{joe} + \frac{2}{3}
\end{aligned}$$

As a result, the miss rate of Luke has to be at least $\frac{2}{3}$ to ensure the surviving of Joe in 3 of 4 cases.

As this example illustrates, there is a wide range of applications for modeling real-world problems in Markov Chains and analyzing them by model checking. In general, Markov Chains are often used to model non-determinism by use of probabilities. Thus, instead of having to deal with non-deterministic models, we encode this non-determinism using uniformly distributed probabilities. In our example the non-deterministic choice of the first shooter is determined by probabilities with certain weights. Further on, unpredictable events can be modeled by Markov Chains as well. For example, the loss of a message in an unreliable channel can be modeled with probabilities.

Furthermore, Parametric Markov Chains, which we consider in this thesis, allow us to leave certain probabilities open. Therefore, the probability of an event to happen does not need to be fixed while modeling, but can be instantiated later. This allows for greater flexibility where certain parts of a Markov Chain can be left unclear and later be instantiated with a range of possible values. For Parametric Markov Chains, we only have to run the model checking algorithm once, but in the end we can instantiate every parameter with a range of values and therefore we get probability bounds for a set of similar Markov Chains. Thus, we avoid model checking a Markov Chain for every possible parameter instantiation. In this approach the user is in the focus, because he can interactively determine which parameter values he is interested in. As seen in the example, not only the result-

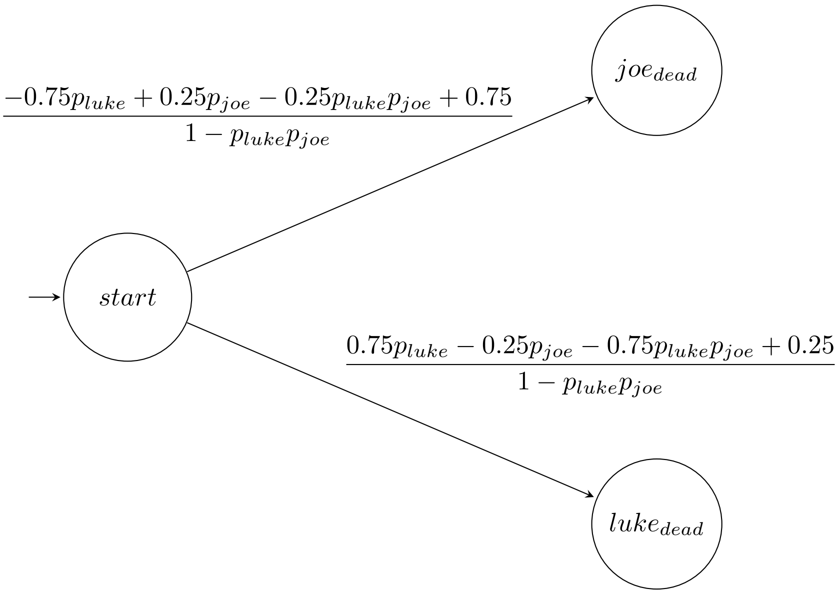

$\frac{-0.75p_{luke} + 0.25p_{joe} - 0.25p_{luke}p_{joe} + 0.75}{1 - p_{luke}p_{joe}}$
joe_{dead}
$start$
$\frac{0.75p_{luke} - 0.25p_{joe} - 0.75p_{luke}p_{joe} + 0.25}{1 - p_{luke}p_{joe}}$
$luke_{dead}$

Figure 1.4: Result after model checking the parametric model

ing probability bound can be important, but also the parameter range for a given probability bound.

Some work on model checking of Parametric Markov Chains has been done before. The main approach, which we are aware of, uses state elimination [13] and was first introduced in [5] and then later refined in [11]. Here, the main idea is to consider the probabilities of the Parametric Markov Chain as letters in a *Finite State Automaton*. Then, the state elimination algorithm for gaining a *regular expression* from an automaton is used. The states are successively eliminated to get finally a *rational function* for the probability of reaching a target state from an initial state. The author of [11] states that an exponential blowup of the resulting function can be avoided in most cases by simplifying the rational function in every elimination step.

This thesis is structured as follows. We start by giving an introduction to the topic of Markov Chains and rational functions in Chapter 2. Then we give an overview about related work, which has been done in this area before, in Chapter 3. In Chapter 4 we present our approach for model checking Parametric Markov Chains by using an SCC-based approach and further prove its correctness. An example showing all steps of our algorithm concludes the chapter. We implemented our algorithm in the COMICS tool [16]. Details on the implementation are given in Chapter 5 along with some benchmarks to compare our approach with the state elimination technique implemented in PARAM [10]. Finally, we give a conclusion in Chapter 6 and give an outlook to future work based on this approach.

2 Preliminaries

We first introduce the main concepts for the topic of Parametric Markov Chains. Intuitively, a Markov Chain is a transition system with probability distribution for the successor states of every state. We therefore have probabilities attached to the edges. That means, for Markov Chains the next state is stochastically determined by the probabilities of the transitions. Consider for example the Markov Chain depicted in Figure 2.1.

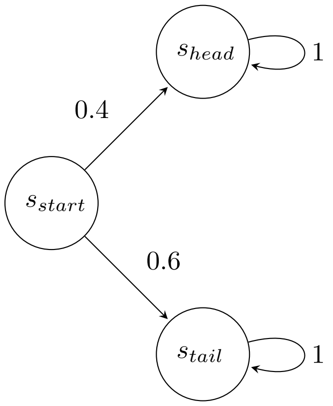


Figure 2.1: Toss of an unfair coin

This Markov Chain is a model for a coin toss with an unfair coin. Starting in state s_{start}, we reach the state s_{head} with probability 0.4, i. e., "head" occurs with probability 0.4. The state s_{tail}, indicating that "tail" occurs, can be reached with probability 0.6. We cannot predict for a coin toss whether we end in state s_{head}, i. e., whether "head" occurs. We can only say that this happening is not so probable as seeing "tail".

In this thesis we extend the common Markov Chain model to also handle rational functions. So instead of having real numbers as probabilities on the edges we also allow rational functions with parameters in it. Therefore we can left open several probabilities in our system and later instantiate them with different values. This allows for greater flexibility in the model and also some aspects do not have to be specified but can be left open instead.

For our later algorithm especially the concept of strongly connected components (SCCs) is very important. These SCCs yield to a set of states which we then abstract to reduce the state size of our Markov Chain and to easily gain a model checking result. We therefore also introduce SCCs here.

We now introduce the basic concepts that are used in this thesis. The definitions are similar to the ones used in [11] and [14].

2.1 Rational Function

We start with introducing *rational functions* which will encode our probabilities.

Definition 1 (Rational Function)**.** Let $V = \{x_1, ..., x_n\}$ be a finite set of *variables* with domain $\mathbb{R}$.

A *polynomial* g over V can be composed according to the following grammar:

$$g := c \mid x \mid g + g \mid (g) \cdot (g)$$

with g being a polynomial, $c \in \mathbb{R}$, $x \in V$ where $+$ is the standard addition and $\cdot$ the standard multiplication.
A *rational function* f over V is a fraction built by the standard division of two polynomials g_1, g_2 over V as

$$f = \frac{g_1}{g_2} \text{ with } g_2 \neq 0.$$

$g_2 \neq 0$ means that g_2 is not reducible to 0.

Example 1 (Rational Function)**.** For variables $p, q \in V, p + q \neq 0$ we have polynomials

$$f := p^2 + 2pq + q^2 \qquad g := (p^3 + 1)(p + q)$$

and a rational function

$$h := \frac{f}{g} = \frac{p^2 + 2pq + q^2}{(p^3 + 1)(p + q)} = \frac{p + q}{p^3 + 1}.$$

Using an *evaluation* we instantiate the variables in a rational function to gain real values for them.

Definition 2 (Evaluation)**.** Let $\mathbb{R}[x_1, ..., x_n]$ be the set of polynomials over the set of variables $V = \{x_1, ..., x_n\}$. We denote by $\mathcal{F}_V : \mathbb{R}[x_1, ..., x_n] \to \mathbb{R}$ the *set of rational functions* with input $\mathbb{R}[x_1, ..., x_n]$ and output $\mathbb{R}$ and by $dom(f)$ the *domain* of a function f. An *evaluation* u is a function $u : X \to \mathbb{R}$ for a subset $X \subseteq V$. The evaluation u is called *total* if $X = V$. For a rational function $f \in \mathcal{F}_V$ and an evaluation u, we denote by $f[X/u]$ the function obtained by substituting each $x \in (X \cap dom(u))$ with its evaluation $u(x)$.

Example 2 (Evaluation)**.** For the rational function h from Example 1, an evaluation u with $u(p) = 2$, $u(q) = 0.5$ and a set of variables $X_1 := \{p\}$ we get

$$h[X_1/u] = \frac{2 + q}{2^3 + 1} = \frac{2 + q}{9}.$$

For $X_2 := \{p, q\}$ we get

$$h[X_2/u] = \frac{2 + 0.5}{9} = \frac{5}{18} \approx 0.2778.$$

2.2 Finite State Automaton

We go on with the introduction of a *Finite State Automaton (FSA)*.

Definition 3 (FSA)**.** A *Finite State Automaton* $\mathcal{A}$ is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ with

- Q: a finite set of states
- Σ: a finite set of symbols called the alphabet
- $\delta : Q \times \Sigma \to 2^Q$: the transition function
- $q_0 \in Q$: the initial state
- $F \subseteq Q$: a finite set of accepting states.

FSAs recognize exactly the set of regular languages. Regular languages can also be expressed by regular expressions which we now define.

Definition 4 (Regular Expression)**.** A *regular expression* $\mathcal{R}$ for an alphabet Σ is defined as:

- ε is a regular expression
- $a \in \Sigma$ is a regular expression
- Let $r_a, r_b \in \mathcal{R}$ be regular expressions, then
  - $r_a \cdot r_b$ (concatenation) is a regular expression
  - $r_a \mid r_b$ (alternation) is a regular expression
  - r_a^* (Kleene star) is a regular expression

For every FSA $\mathcal{A}$ the equivalent regular expression $\mathcal{R}_\mathcal{A}$ can be constructed by use of the state-elimination algorithm [13]. However, as shown in [9] for a FSA with n states, the size of the resulting regular expression explodes: $n^{\mathcal{O}(\log(n))}$.

2.3 Discrete-time Markov Chain

We introduce the well-known concept of *Discrete-time Markov Chains (DTMC)* to model stochastic behavior in a transition system as explained before. In case of DTMCs we only have rational numbers as probabilities, which we will later extend to rational functions.

The set of *atomic propositions* AP intuitively is the set of possible "behaviors" occurring in the transition system. If a state has a certain "behavior", it is labeled with the corresponding atomic proposition. For example the atomic proposition "error" could be used to indicate states where an erroneous behavior happens. Model checking then could yield a probability of reaching such unwanted states.

Definition 5 (DTMC)**.** Let AP be a set of atomic propositions.
A *Discrete-time Markov Chain (DTMC)* is a tuple $\mathcal{D} = (S, I, P, L)$ with

- S: a non-empty finite[1] set of states
- $I : S \to [0,1]$: the initial distribution with $\sum_{s \in S} I(s) = 1$
- $P : S \times S \to [0,1]$: the transition probability matrix with $\forall s \in S : \sum_{s' \in S} P(s,s') = 1$
- $L : S \to 2^{AP}$: the state labeling function.

The transition probabilities must be within the interval $[0,1]$ to be valid. Further on, the outgoing probability of every state must sum up to 1.

Example 3 (DTMC)**.** Consider the example DTMC $\mathcal{D}_{ex}$ depicted in Figure 2.2.

Here we have a set of nine states and no labels. We have a single initial state s_1 and therefore an initial distribution with $I(s_1) = 1$ and $I(s_i) = 0$ for $2 \leq i \leq 9$. The initial distribution can also be written as a vector:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T .$$

[1] There are also DTMCs with countable but infinitely many states. However, we do not consider them here.

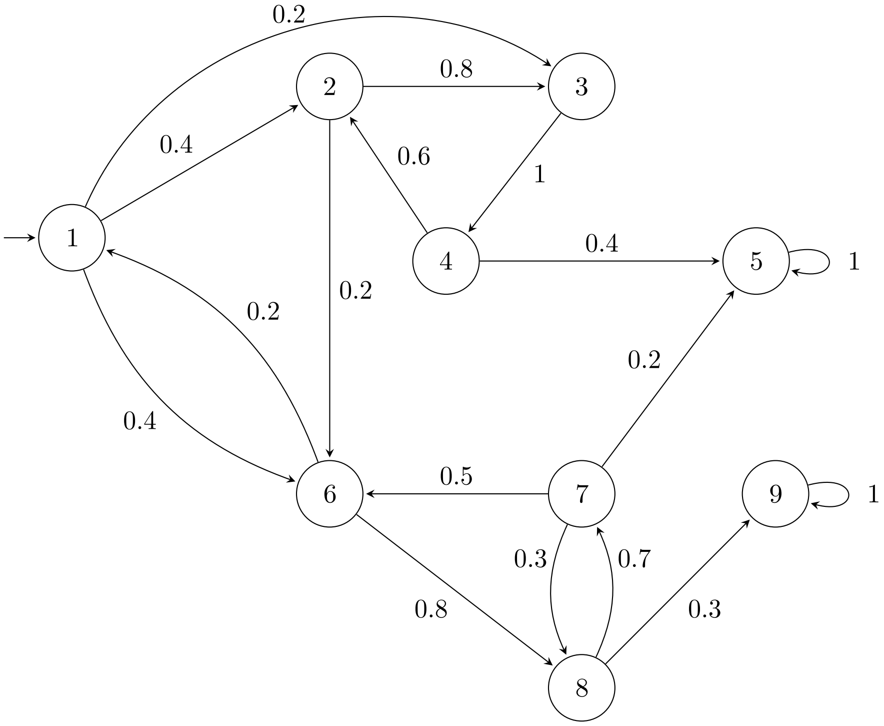


Figure 2.2: Example DTMC $\mathcal{D}_{ex}$

The transition probabilities are shown on every transition. For example it is $P(s_4, s_2) = 0.6$ and $P(s_4, s_5) = 0.4$. So the outgoing probabilities of state s_4 sum up to $1 = 0.6 + 0.4$. The transition probability matrix is:

$$\begin{pmatrix} 0 & 0.4 & 0.2 & 0 & 0 & 0.4 & 0 & 0 & 0 \\ 0 & 0 & 0.8 & 0 & 0 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.6 & 0 & 0 & 0.4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0.3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

From now on we also use the characterization DTMC for the *underlying directed graph* $G = (S, E)$ of a DTMC $\mathcal{D}$. Here the set of vertices in the graph is the set of states S in $\mathcal{D}$. In the underlying directed graph there is a transition from state $s \in S$ to state $s' \in S$ iff $P(s, s') > 0$ in the DTMC $\mathcal{D}$. The set of edges E of G can be defined as

$$E = \{(s, s') \mid s, s' \in S, P(s, s') > 0\}.$$

We consider the concept of paths within a set of states K, i. e., runs of the DTMC through these states. In the following definitions we assume DTMCs with a single initial state, i. e., $|\{s \in S \mid I(s) > 0\}| = 1$. Note that each DTMC can be transformed to this form by adding a new unique initial state with transitions to the former initial states. The notation is $\mathcal{D} = (S, s_I, P, L)$.

Definition 6 (Paths)**.** Assume a DTMC $\mathcal{D} = (S, I, P, L)$ and a state set $K \subseteq S$. A *finite path* of $\mathcal{D}$ in K is a finite sequence $\pi = s_1 ... s_n$ of states with $s_i \in K, 1 \leq i \leq n$ where for every $1 \leq i \leq n-1$ there exists a transition from s_i to s_{i+1}, i. e., $P(s_i, s_{i+1}) > 0$.

Similarly, an *infinite path* of $\mathcal{D}$ in K is an infinite sequence $\pi = s_1 s_2 s_3 ...$ of states with $s_i \in K$ and $P(s_i, s_{i+1}) > 0$ for all $i \in \mathbb{N}$. The infinite loop in a state s is denoted by s^ω.

The set of *paths* of $\mathcal{D}$ in K is denoted by Paths^K. It consists of all finite paths $\mathit{Paths}^K_{\mathit{fin}}$ and infinite paths $\mathit{Paths}^K_{\mathit{inf}}$ in K:

$$\mathit{Paths}^K = \mathit{Paths}^K_{\mathit{fin}} \uplus \mathit{Paths}^K_{\mathit{inf}}.$$

For states $s, t \in K$ we denote by $\mathit{Paths}^K_{\mathit{fin}}(s)$ the set of all finite paths in K starting in s, $\mathit{Paths}^K_{\mathit{fin}}(s,t)$ the set of all finite paths in K starting in s and ending in t, and similarly $\mathit{Paths}^K_{\mathit{inf}}(s)$ for all infinite paths in K starting in s.

Example 4 (Paths)**.** A path in the previous Example 3 could be $\pi_{ex} = s_1 s_2 s_3 s_4 s_5$ which is a finite path. The path $\pi_2 = s_1 s_2 s_3 s_4 s_5 s_5 s_5 ...$ is an infinite path with prefix π_{ex}. It is $\pi_{ex} \in \mathit{Paths}^S_{\mathit{fin}}(s_1)$ and $\pi_2 \in \mathit{Paths}^S_{\mathit{inf}}(s_1)$.

A state $t \in S$ is *reachable* from a state $s \in S$ iff $\mathit{Paths}^S_{\mathit{fin}}(s,t) \neq \emptyset$. A set of states $K \subseteq S$ is called *absorbing* in $\mathcal{D}$ iff $\exists s \in K$ such that $\forall s' \in S \setminus K : \mathit{Paths}^S_{\mathit{fin}}(s, s') = \emptyset$, i. e., from at least one state in K only states in K are reachable. K is called *bottom* iff every state in K is absorbing. A state s is called *bottom* iff $\{s\}$ is bottom.

We now define strongly connected subsets and strongly connected components:

Definition 7 (Strongly connected subset)**.** A set $K \subseteq S$ is called *strongly connected* in a DTMC $\mathcal{D} = (S, I, P, L)$ iff $\forall s, t \in K$ it is $\mathit{Paths}^K_{\mathit{fin}}(s,t) \neq \emptyset$.

In other words in a strongly connected subset for every pair of states s, s' there exists a path π from s to s' only visiting states in K

Definition 8 (SCC)**.** A *strongly connected component (SCC)* $K \subseteq S$ of a DTMC $\mathcal{D} = (S, I, P, L)$ is a strongly connected subset of S such that for all strongly connected subsets $K' \subseteq S$ of $\mathcal{D}$ with $K \subseteq K'$ we have $K = K'$.

Example 5 (SCC)**.** Some strongly connected subsets of the state space S of $\mathcal{D}_{ex}$ from Example 3 are $K_1 = \{s_7, s_8\}$ and $K_2 = \{s_2, s_3, s_4\}$. The only strongly connected component of $\mathcal{D}_{ex}$ is the state set $K_{SCC} = \{s_1, s_2, s_3, s_4, s_6, s_7, s_8\}$. The states s_5 and s_6 are bottom states, because they only have self-loops.

2.4 Probabilities

After introducing paths in a DTMC we now want to define the probabilities for paths. Before this we have to define probability spaces. For more details we refer to [2].

Definition 9 (σ-algebra)**.** A *σ-algebra* is a pair $(\mathit{Outc}, \mathfrak{E})$ with

- Outc: a non-empty set called "outcomes"
- $\mathfrak{E} \subseteq 2^{Outc}$: called "events" which contains the empty set and is closed under complement and countable union:
  - $\emptyset \in \mathfrak{E}$
  - $E \in \mathfrak{E} \implies \overline{E} = \mathit{Outc} \setminus E \in \mathfrak{E}$
  - $E_1, E_2, ... \in \mathfrak{E} \implies \bigcup\limits_{i \geq 1} E_i \in \mathfrak{E}$

$\mathfrak{E}$ also is closed under countable intersections as

$$\bigcap_{i \geq 1} E_i = \overline{\bigcup_{i \geq 1} \overline{E_i}}.$$

Next we define a probability measure where we give every event a certain probability to encode the likelihood of this event to happen.

Definition 10 (Probability measure)**.** A *probability measure* on $(Outc, \mathfrak{E})$ is a function $Pr : \mathfrak{E} \to [0,1]$ with $Pr(Outc) = 1$. Additionally for pairwise disjoint events $E_1, E_2, ...$ with $E_i \in \mathfrak{E}$ for all $i \geq 1$ it should hold:

$$Pr(\biguplus_{i\geq 1} E_i) = \sum_{i\geq 1} Pr(E_i).$$

In total we get a probability space.

Definition 11 (Probability space)**.** A *probability space* is a triple $(Outc, \mathfrak{E}, Pr)$ with a σ-algebra $(Outc, \mathfrak{E})$ and Pr a probability measure on this σ-algebra.

An example should make this concept clearer.

Example 6. Consider the example from Figure 2.1. Here we have two possible outcomes "heads" and "tails" for the coin toss: $Outc = \{heads, tails\}$. The σ-algebra is the powerset of $Outc$: $\mathfrak{E} = 2^{Outc}$. For this unfair coin the probability measure is given by

$$Pr(\emptyset) = 0, Pr(\{heads\}) = 0.4, Pr(\{tails\}) = 0.6, Pr(\{heads, tails\}) = Pr(Outc) = 1.$$

We now introduce the cylinder set of all infinite paths starting with the same finite path, i. e., having the same prefix.

Definition 12 (Cylinder set)**.** For a DTMC $\mathcal{D}$ and a finite path $\pi = s_1...s_n \in Paths^S_{fin}$, the *cylinder set* of π in $\mathcal{D}$ is defined as

$$Cyl(\pi) = \{\pi' \in Paths^S_{inf} \mid \pi \text{ is prefix of } \pi'\}.$$

By use of the cylinder set we can now define the concrete σ-algebra for a DTMC $\mathcal{D}$, where the cylinder sets are the "events".

Definition 13 (σ-algebra of a DTMC)**.** For a DTMC $\mathcal{D} = (S, s_I, P, L)$ and a state $s \in S$ we define the associated σ-algebra $\mathfrak{E}^{\mathcal{D}}$ as the smallest σ-algebra that contains all cylinder sets $Cyl(\pi)$ for all finite paths $\pi \in Paths^S_{fin}(s)$.

Then there exists a unique probability measure Pr on $\mathfrak{E}^{\mathcal{D}}$. This measure now gives us the probabilities for paths.

Definition 14 (Probabilities)**.** For a DTMCD $\mathcal{D} = (S, s_I, P, L)$ the *probability of a finite path* $\pi \in Paths^{\mathcal{D}}_{fin}, \pi = s_1...s_n$ is the product of all its transition probabilities:

$$Pr^{\mathcal{D}}_{fin}(\pi) = \Pi^{n-1}_{i=1} P(s_i, s_{i+1}).$$

For the associated cylinder set $Cyl(\pi)$ the probability is given as

$$Pr^{\mathcal{D}}(Cyl(\pi)) = Pr^{\mathcal{D}}_{fin}(\pi).$$

Let $s \in S$ and $R \subseteq Paths^{\mathcal{D}}_{fin}(s)$ be a set of finite paths and $R' \subseteq R$ the set of paths which are prefix-free in R. A path π is *prefix-free* in R iff for every path $\pi' \in R$, π is not a prefix of π'. Then the probability is

$$Pr^{\mathcal{D}}_{fin}(R) = \sum_{\pi \in R'} Pr^{\mathcal{D}}_{fin}(\pi).$$

For simplicity the superscript $^{\mathcal{D}}$ is left out, if it is clear from the context.

Example 7 (Probabilities)**.** For the finite path of Example 4 we get the following probability: $Pr_{fin}(\pi_{ex}) = P(s_1, s_2) \cdot P(s_2, s_3) \cdot P(s_3, s_4) \cdot P(s_4, s_5) = 0.4 \cdot 0.8 \cdot 1 \cdot 0.4 = 0.128$. The cylinder set of π_{ex} is $Cyl(\pi_{ex}) = \{s_1 s_2 s_3 s_4 s_5^{\omega}\}$. We then get the probability of the cylinder set of π_{ex} as: $Pr(Cyl(\pi_{ex})) = Pr_{fin}(\pi_{ex}) = 0.128$.

We now give a few remarks on special cases for probabilities. As stated in [2], the following property holds

Property 1 (Behavior for bottom SCCs)**.** For each state s in a (finite) DTMC $\mathcal{D}$ it holds that

$$Pr(\{\pi \in Paths^{\mathcal{D}}_{inf}(s) \mid \text{the set of infinitely often visited states along } \pi \text{ is a bottom SCC in } \mathcal{D}\}) = 1 \tag{2.1}$$

Therefore with probability 1 every infinite run in $\mathcal{D}$ eventually reaches a bottom SCC and visits all states there infinitely often.

We also have a special case for the probability of a set of paths containing a cycle in a DTMC $\mathcal{D}$.

Property 2 (Probability of cycle)**.** Assume a DTMC $\mathcal{D} = (S, s_I, P, L)$, states $s_1, ..., s_n \in S$ with $s_1...s_n s_1 \in Paths^{\mathcal{D}}_{fin}$ and a path $\pi' \in Paths^{\mathcal{D}}_{fin}(s_1)$. For the set of paths

$$P := \{\pi = (s_1...s_n)^k \pi' \in Paths^{\mathcal{D}}_{inf}(s_1) \mid k \geq 0\}$$

with

$$p_{\text{loop}} := Pr(s_1...s_n s_1) \in (0,1)$$

the probability of all these paths can be expressed as:

$$\begin{aligned} Pr(P) &= \sum_{k=0}^{\infty} Pr((s_1...s_n)^k \pi') \\ &= \sum_{k=0}^{\infty} p_{\text{loop}}^k \cdot Pr(\pi') \\ &= Pr(\pi') \cdot \sum_{k=0}^{\infty} p_{\text{loop}}^k \\ &\overset{\text{geometric series}}{=} Pr(\pi') \cdot \frac{1}{1 - p_{\text{loop}}} \end{aligned}$$

Because of $p_{\text{loop}} \in (0,1)$, the geometric series converges and the infinite sum can be computed.

2.5 Probabilistic Computation Tree Logic

After introducing Markov Chains and how to compute probabilities for certain paths, we are now interested in creating conditions for the atomic propositions which should hold in the states on a path. Instead of considering all possible paths in a DTMC $\mathcal{D}$ we only want to consider those, which fulfill certain conditions. As explained in the beginning, we could only be interested in those paths which eventually lead to a erroneous state. By using probabilities we cannot be certain if specific events always occur, but we can get the probability for an event to happen. To express conditions on paths we use *Probabilistic Computation Tree Logic (PCTL)* [12] which is based on *Computation Tree Logic (CTL)*.

Definition 15 (PCTL)**.** Let AP be a set of atomic propositions.
PCTL *state formulas* are formed according to the following grammar:

$$\begin{aligned} \Phi ::= &\text{true} && \text{(boolean)} \\ &\mid a && \text{(Atomic proposition } a \in AP) \\ &\mid \Phi \wedge \Phi && \text{(Conjunction)} \\ &\mid \neg\Phi && \text{(Negation)} \\ &\mid \mathbb{P}_J(\phi) && \text{(Probabilistic operator with an interval with rational bounds } J \subseteq [0,1] \subseteq \mathbb{R}) \end{aligned}$$

PCTL *path formulas* are formed according to the following grammar:

$$\begin{aligned} \phi ::= &\bigcirc \Phi && \text{(Next)} \\ &\mid \Phi U \Phi && \text{(Until)} \\ &\mid \Phi U^{\leq n} \Phi && \text{(Bounded until with } n \in \mathbb{N}) \end{aligned}$$

Other syntactic sugar can easily be derived:

$$\begin{aligned} \Phi_1 \vee \Phi_2 &= \neg(\neg\Phi_1 \wedge \neg\Phi_2) && \text{(Disjunction)} \\ \Diamond\Phi &= \text{true } U \Phi && \text{(Eventually)} \\ \Diamond^{\leq n}\Phi &= \text{true } U^{\leq n} \Phi && \text{(Bounded Eventually with } n \in \mathbb{N}) \\ \mathbb{P}_{\leq p}(\Box\Phi) &= \mathbb{P}_{\geq 1-p}(\Diamond\neg\Phi) && \text{(Always)} \end{aligned}$$

The satisfaction relation $\models$ for PCTL formulas and a DTMC $\mathcal{D}$ is defined as follows:

Definition 16 (Satisfaction relation $\models$)**.** Assume a DTMC $\mathcal{D} = (S, s_I, P, L)$, a state $s \in S$, a path $\pi = s_0 s_1 s_2 ... \in Paths^{\mathcal{D}}_{inf}$, PCTL state formulas Φ_1, Φ_2 and PCTL path formula ϕ. We then have:

$$\mathcal{D}, s \models a \iff a \in L(s) \tag{2.2}$$

$$\mathcal{D}, s \models \neg\Phi \iff \mathcal{D}, s \not\models \Phi \tag{2.3}$$

$$\mathcal{D}, s \models \Phi_1 \wedge \Phi_2 \iff \mathcal{D}, s \models \Phi_1 \text{ and } \mathcal{D}, s \models \Phi_2 \tag{2.4}$$

$$\mathcal{D}, s \models \mathbb{P}_J(\phi) \iff Pr(s \models \phi) = Pr(\{\pi \in Paths^{\mathcal{D}}_{inf}(s) \mid \mathcal{D}, \pi \models \phi\}) \in J \tag{2.5}$$

$$\mathcal{D}, \pi \models \bigcirc\Phi \iff s_1 \models \Phi \tag{2.6}$$

$$\mathcal{D}, \pi \models \Phi_1 U \Phi_2 \iff \exists 0 \leq j : (\mathcal{D}, s_j \models \Phi_2 \wedge (\forall 0 \leq k < j : \mathcal{D}, s_k \models \Phi_1)) \tag{2.7}$$

$$\mathcal{D}, \pi \models \Phi_1 U^{\leq n} \Phi_2 \iff \exists 0 \leq j \leq n : (\mathcal{D}, s_j \models \Phi_2 \wedge (\forall 0 \leq k < j : \mathcal{D}, s_k \models \Phi_1)) \tag{2.8}$$

It is $Sat(\Phi) = \{s \in S \mid \mathcal{D}, s \models \Phi\}$. We define $\mathcal{D} \models \Phi$ iff $\mathcal{D}, s_I \models \Phi$.

As seen in Equation 2.2, a state s satisfies an atomic proposition a iff this atomic proposition is an element of the labeling for this state. The satisfiability of conjunction and negation should be intuitively clear.

The probabilistic operator as in Equation 2.5 is satisfied iff the probability of all infinite paths starting in s and satisfying the formula ϕ is in the interval J. This operator is the main difference to plain CTL, because here the stochastic nature of the DTMC plays a role.

A "next" formula as in Equation 2.6 is satisfied for a path π iff the next state on this paths, i. e., the successor of the starting state s_0, satisfies the formula Φ.

An "until"formula of the form in Equation 2.7 is satisfied for a path π iff eventually Φ_2 is satisfied for some state s_j on the path. All predecessors of s_j have to satisfy Φ_1. For "bounded until" as in Equation 2.8, the state s_j has to occur within n steps.

Example 8 (PCTL)**.** For the DTMC in Figure 2.2 and the set $\{1, ..., 9\}$ of atomic propositions with $L(s_i) = \{i\}, 1 \leq i \leq 9$ we have the PCTL formula $\Phi_{ex} = \mathbb{P}_{\geq 0.5}(\Diamond 5)$, i. e., the probability of eventually reaching the state s_5 is greater or equal than 0.5. If we only want to consider the upper half of the graph, the PCTL formula would be $\Phi_2 = \mathbb{P}_{\geq 0.5}((1 \vee 2 \vee 3 \vee 4)\ U\ 5)$.

For our approaches we restrict the PCTL formulas to only "unbounded until" properties, i. e., the "next" and "bounded until" operators are left out.

2.5.1 Transformation of DTMCs for PCTL Formulas

For model checking formulas of the form $\Phi = \mathbb{P}_J(\Phi_1 U \Phi_2)$ on a DTMC $\mathcal{D} = (S, s_I, P, L)$ we transform $\mathcal{D}$ into a DTMC $\mathcal{D}' = (S, s_I, P', L')$ by applying three steps:

1. Generate a labeling for Φ_1 and Φ_2: every state $s \in S$ satisfying Φ_1 gets the additional label $L'(s) := L(s) \cup \{\Phi_1\}$. Checking the satisfiability of a formula Φ_1 on state s as $s \models \Phi_1$ may invoke a recursive model checking of the subformula on $\mathcal{D}$'. We label similarly with Φ_2.
2. Make all states satisfying Φ_2 absorbing in $\mathcal{D}$', because they are the target states to be reached. Further more, make all states satisfying $\neg\Phi_1 \wedge \neg\Phi_2$ absorbing in $\mathcal{D}$', because paths visiting this state do not satisfy the "until"-formula.
3. Compute the probability of reaching a state satisfying Φ_2 from an initial state in the DTMC $\mathcal{D}$'.

The main part of the model checking algorithm is Step 3. Here we only have to make a reachability analysis, that means only check the formula

$$\Phi_{trans} = \mathbb{P}_J(\Diamond\Phi_2) \tag{2.9}$$

on the transformed DTMC $\mathcal{D}$'. We therefore only need to consider model checking for reachability formulas of the form given in Equation 2.9. From now on we only consider transformed DTMCs, which we denote by $\mathcal{D}$.

2.6 Parametric Markov Chain

Now we extend DTMCs to Parametric Markov Chains (PMC) by allowing rational functions instead of real numbers as probabilities. Except for this difference, the definition of a PMC is similar to the one for a DTMC.

Definition 17 (PMC)**.** A *Parametric Markov Chain (PMC)* is a tuple $\mathcal{M} = (S, I, P, V, L)$ with

- S: a set of states
- $I : S \to \mathcal{F}_V$: the initial distribution with $\sum_{s\in S} I(s) = 1$
- $P : S \times S \to \mathcal{F}_V$: the parametric transition probability matrix with $\forall s \in S : \sum_{s'\in S} P(s, s') = 1$
- $V = \{x_1, ..., x_n\}$: the finite set of parameters with domain $\mathbb{R}$
- $L : S \to 2^{AP}$: the state labeling.

As for DTMCs, in the following we assume that there is a single initial state $s_I \in S$ with $I(s_I) = 1$.

Notice that we cannot yet make restrictions on the probabilities to be within $[0, 1]$. Now we have rational functions with parameters as probabilities and we cannot ensure such restrictions to hold for every possible evaluation. We therefore later define evaluations which satisfy these restrictions.

Example 9 (PMC)**.** Consider the example PMC $\mathcal{M}_{ex}$ depicted in Figure 2.3.

This PMC is very similar to the DTMC from Example 3. We added two parameters p and q, hence $V = \{p, q\}$. Further on, we replaced some probabilities with these parameters. For example we now have transition probabilities $P(s_4, s_2) = q$ and $P(s_4, s_5) = 1 - q$. This still sums up to $1 = q + (1 - q)$. Notice that for every parameter introduced on an outgoing transition of a state s we must have a dependent parameter on another outgoing transition of this state s to gain a total outgoing probability of 1. In this example we therefore also have the transition probability $1 - q$ additional to the probability q.

For an evaluation u and a PMC $\mathcal{M}$ we get an *induced PMC* $\mathcal{M}_u$ by substituting each variable in $dom(u)$ with its evaluation.

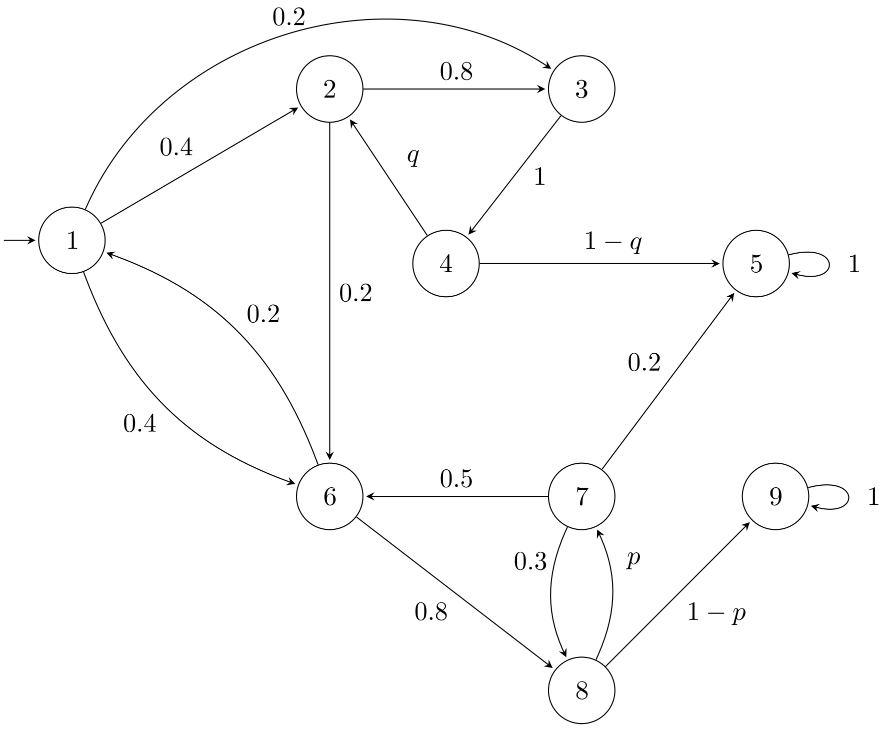


Figure 2.3: Example PMC $\mathcal{M}_{ex}$

Definition 18 (Induced PMC for evaluation)**.** By using an evaluation u we obtain a new PMC $\mathcal{M}_u = (S, I_u, P_u, V, L)$ with

- S as before
- $I_u : S \to \mathcal{F}_{V \setminus dom(u)}$ such that $\forall s \in S : I_u(s) := I(s)[dom(u)/u]$
- $P_u : S \times S \to \mathcal{F}_{V \setminus dom(u)}$ such that $\forall s, s' \in S : P_u(s, s') := P(s, s')[dom(u)/u]$
- $V_u : V \setminus dom(u)$
- L as before.

By using a total evaluation u with domain $dom(u) = V$ we substitute each variable and therefore obtain concrete values for each probability instead of rational functions. We then have an induced DTMC $\mathcal{D}_u$ and can check, if this DTMC is well-defined according to Definition 5.

Example 10 (Induced PMC for evaluation)**.** The DTMC $\mathcal{D}_{ex}$ from Example 3 is an induced DTMC for the example PMC $\mathcal{M}_{ex}$ from Example 9 by using the evaluation u_{ex} with $u_{ex}(p) = 0.7$ and $u_{ex}(q) = 0.6$.

We now define the set of probabilities $Prob^{\mathcal{M}} \subseteq \mathcal{F}_V$ for a PMC $\mathcal{M}$ as

$$Prob^{\mathcal{M}} := \{I(s) \mid s \in S\} \cup \{P(s, s') \mid s, s' \in S\}$$

and similarly $Prob_u^{\mathcal{M}}$ for an evaluation u.

Definition 19 (Well-defined evaluation)**.** A total evaluation u on a PMC $\mathcal{M}$ is called *well-defined* iff the following two conditions are satisfied:

- Every probability is in the interval $[0,1]$:

$$\forall f \in \mathit{Prob}_u^{\mathcal{M}} : f[\mathit{dom}(u)/u] \in [0,1].$$

- Every rational function f, which is not 0, does not evaluate to 0:

$$\forall f \in \mathit{Prob}_u^{\mathcal{M}} : f \neq 0 \implies f[\mathit{dom}(u)/u] \neq 0.$$

The first condition ensures well-defined probabilities, which are within $[0,1]$, and is necessary. The second condition could be omitted, but then transitions in a PMC could fall away, because their probability is 0. We then have varying structures of our underlying graph for different evaluations. This would affect the model checking result, because certain paths now could not reach the target state anymore, because of left out transitions. Further on, in our algorithm we search for SCCs. Therefore the underlying graph structure must stay unmodified and we require this second condition to be satisfied.

Example 11 (Well-defined evaluation)**.** It is easy to see that the evaluation u_{ex} from the previous Example 10 is well-defined, because $u_{ex}(p) = 0.7$, $u_{ex}(1-p) = 0.3$, $u_{ex}(q) = 0.6$ and $u_{ex}(1-q) = 0.4$ are within the interval $(0,1]$. Every other probability without a parameter also satisfies these conditions.

We define transitions and paths for PMCs similarly to DTMCs. In a PMC $\mathcal{M}$ there is a transition from state $s \in S$ to state $s' \in S$ iff $P_u(s,s') > 0$ for all well-defined evaluations u. Paths now are defined accordingly to DTMCs.

The probability of a finite path $\pi \in \mathit{Paths}_{\mathit{fin}}^{\mathcal{M}}$, $\pi = s_1 ... s_n$ is defined as

$$\mathit{Pr}_{\mathit{fin}}^{\mathcal{M}}(\pi) = \Pi_{i=1}^{n-1} P(s_i, s_{i+1}).$$

Now $\mathit{Pr}_{\mathit{fin}}^{\mathcal{M}}(\pi)$ is a rational function. Because of $P(s_i, s_{i+1})$ being transitions we have $P(s_i, s_{i+1}) \in (0,1]$ for every $i = 1, ..., n-1$ and therefore $\mathit{Pr}_{\mathit{fin}}^{\mathcal{M}}(\pi) \in (0,1]$ as well. Using cylinder sets we can define $\mathit{Pr}^{\mathcal{M}}(\pi)$ for infinite paths π as seen before in Definition 14.

3 Related Work

Some work has already been done on model checking Parametric Markov Chains. We will now give a short overview about the related work in this area and the ideas and algorithms developed so far to tackle this problem.

3.1 Model Checking by Symbolic State Elimination

The first approach in this area, that we are aware of, was done by Conrado Daws from the University of Nijmegen in 2004. Daws presented his idea in “Symbolic and Parametric Model Checking of Discrete-Time Markov Chains” [5].

First of all Daws derives a finite state automaton $\mathcal{A}$ from a given DTMC.

Definition 20 (Derived Finite State Automaton)**.** Let $\mathcal{D} = (S, s_I, P, L)$ be a DTMC, then a FSA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ is derived with

- $Q = S$
- $\Sigma = \{p/q \mid \exists i, j \in S : P(i,j) = \frac{p}{q} > 0, p, q \in \mathbb{N}\}$
- $\delta : Q \times \Sigma \to 2^S$: $\delta(p, a) = \{q \in S \mid P(p,q) = a$
- $q_0 = s_I$: the initial state
- $F \subseteq Q$.

By applying this main idea DTMCs can be represented by FSAs, because rational numbers are encoded as elements of the alphabet Σ.

Daws encodes “next” and “until” formulas by using the final states of a FSA similarly to final states in DTMCs. This can be done using similar approaches as shown in the transformation in Section 2.5.1.

By using probabilities as elements of the alphabet Σ, paths in a DTMC can now be represented as regular expressions in the FSA. Daws wants to get a regular expression representing all paths from the initial state to one of the final states. He shows that this regular expression can be obtained by using the state-elimination algorithm from [13]. The idea here is to successively eliminate states by merging in- and outgoing transitions of this state to one transition omitting this state. In this process the new transitions contain regular expressions instead of only letters of the alphabet. An example of this algorithm is given later in Example 12.

The resulting regular expression then can be evaluated to get the probabilities again for satisfying the given formula. This can be done using the evaluation function *val*.

Definition 21 (Evaluation of Regular Expressions)**.** Let $\mathcal{R}(\Sigma)$ be the set of regular expressions over the alphabet Σ.
Let $p/q \in \Sigma$ and r, s be regular expressions. The *evaluation function* $val : \mathcal{R}(\Sigma) \to \mathbb{Q}$ is defined as:

$$\begin{aligned} val(p/q) &:= \frac{p}{q} \\ val(r|s) &:= val(r) + val(s) \\ val(r.s) &:= val(r) \cdot val(s) \\ val(r^*) &:= \frac{1}{1 - val(r)} \end{aligned}$$

It is intuitive for the first three rules that we get the correct evaluated probability. For the last rule we have to see that in the DTMC the transition with probability r can be taken from zero up to infinitely many times. Therefore the probability can be expressed as

$$\sum_{i=0}^{\infty} r^i \overset{\text{(geometric series)}}{=} \frac{1}{1-r}.$$

Until then Daws only uses rational numbers as probabilities. However, he shows that his concept of using a symbolic representation can be extended easily to parametric probabilities. The alphabet Σ only has to be extended to use parameters as well. The evaluation function is extended as follows:

Definition 22 (Extended Evaluation)**.** Let $\langle\mathbb{Q}\rangle X$ be the set of polynomials on the set of parameters X with coefficients in $\mathbb{Q}$. A regular expression r is associated with a pair (P_r, Q_r) of polynomials on X.
Let $p/q \in \Sigma$ and r, s be regular expressions with associated pairs (P_r, Q_r) and (P_s, Q_s). The *extended evaluation function* $val : \mathcal{R}(\Sigma) \to \langle\mathbb{Q}\rangle X \times \langle\mathbb{Q}\rangle X$ is defined as:

$$\begin{aligned} val(p/q) &:= \frac{p}{q} \\ val(x) &:= x \\ val(r) &:= \frac{P_r}{Q_r} \\ val(r|s) &:= \frac{P_r Q_s + Q_r P_s}{Q_r Q_s} \\ val(r.s) &:= \frac{P_r P_s}{Q_r Q_s} \\ val(r^*) &:= \frac{Q_r}{Q_r - P_r} \end{aligned}$$

Daws shows that also Parametric Markov Chains can be handled by his approach and model checking PMCs is possible. Nevertheless, the state-elimination algorithm can lead to exponentially large regular expression as stated in [9].

3.2 Model Checking by State Elimination using Rational Functions

Ernst Moritz Hahn, Holger Hermanns and Lijun Zhang from the Saarland University extended the approach of Daws in 2009 and presented their ideas in “Probabilistic Reachability for Parametric Markov Models” [11].

Instead of using a symbolic representation as in [5] where the probabilities are letters of the alphabet, Hahn et al. use rational functions during the whole process. This allows for simplifying the rational functions during each step of the state-elimination algorithm.

Example 12. We give the example from their publication in Figure 3.1 to visualize the process of eliminating a state s.
By eliminating the state s, the in- and outgoing transitions of this state are merged. The corresponding probabilities now are composed on this merged transition.

The resulting rational function can be simplified during each elimination step and therefore the authors state, that the exponential blow up can be avoided in most cases.

This approach was also implemented in the tool PARAM [10]. One detail of the implementation is the sharing of rational functions to store every occurring rational function only once. Also cancellation has to be done only once for each rational function. A lookup table of arithmetic

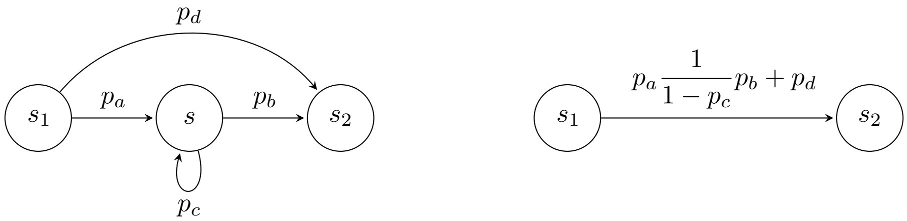


Figure 3.1: Eliminating state s

operations with their operands and their result has also been implemented to not compute this operation multiple times. As stated by the authors, this feature is only useful when using bisimulation.

3.3 Model Checking DTMCs using SCCs

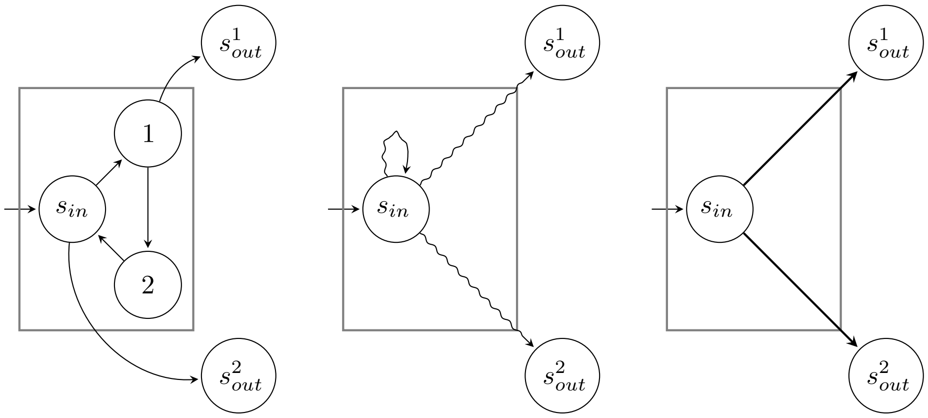
Our approach for PMCs builds upon on the SCC-based approach for model checking DTMCs which was presented by Erika Ábrahám, Nils Jansen, Ralf Wimmer, Joost-Pieter Katoen and Bernd Becker from RWTH Aachen University and Albert-Ludwigs-University Freiburg in "DTMC Model Checking by SCC Reduction" [1]. In this section we only give an intuitive overview about this approach, the details are explained later in Chapter 4 for PMCs.

The main idea in this approach is the bottom-up abstraction of strongly connected subsets. They therefore first search for SCCs in the whole graph. These SCCs can be divided into three different types of states. They first have *input states* which have incoming transitions from states outside the SCC. Then there are *output states* which are not part of the SCC but have an incomming transition from a state in the SCC. Last the *inner states* are all states in the SCC which are not input states.

Then they recursively search for strongly connected subsets in the set of inner states and therefore get a hierarchical structure of connected subsets in the original graph. Starting on the bottom level, they now "abstract" this connected subset to simplify it but preserve the reachability properties. As seen in Figure 3.2(a) they do not have any connected subsets in the inner states anymore, because the search has terminated on the lowest level. Therefore, all paths starting in an input state eventually reach an output state. Every cycle in this strongly connected subset has to visit one of the input states on every cycle run. In the end, the "direct" path from an input state to an output state is taken. Therefore every path consists of finitely many cycle runs and lastly the "direct" path to an output state as seen in Figure 3.2(b). Thus, instead of considering the whole strongly connected subset, they only have to consider all cycles on an input state and all "direct" paths.

Knowing that every path eventually reaches an output state with probability 1 as seen in Property 1, they also do not have to consider the cycles anymore, but instead scale the outgoing transitions to sum up to 1. This can be seen in Figure 3.2(c).

In the end they have only a few "abstracted" transitions from input states to output states and do not have to consider the whole strongly connected subset anymore. Applying this algorithm recursively they can abstract the whole graph and in the end the model checking result can be read from every transition.



(a) SCC with input and output states

(b) First abstraction

(c) Scaling for final abstraction

Figure 3.2: Abstraction

4 SCC-Based Model Checking for PMCs

We introduce our algorithm for model checking PMCs. This algorithm is based on the SCC-based model checking algorithm first presented in [1]. We use this algorithm and extend it to additionally handle Parametric Markov Chains.

4.1 Preprocessing

Before model checking the PMC $\mathcal{M} = (S, I, P, V, L)$ we first have to check its consistency. We therefore only consider well-defined evaluations of the rational functions as in 19. We gain the following constraints which have to hold for every well-defined total evaluation u. Furthermore, we check whether the PMC is well-defined in the sense that it has valid probability distributions.

Constraint 1. The probability of every edge must be greater than 0:
$\forall s, s' \in S : P(s, s') \neq 0 \implies P[X/u](s, s') > 0$

Constraint 2. The outgoing probability of each state must be 1:
$\forall s \in S : \sum\limits_{s' \in S} P(s, s') = 1.$

These constraints later can be used in the model checking algorithm to gain finer restrictions on our parameters. This is useful for the synthesis of the PMC and the analysis of the model checking probability.

Example 13 (Constraints)**.** For the PMC of Example 9 we gain constraints like

$$q > 0, \qquad 1 - q > 0, \qquad p > 0, \qquad 1 - p > 0.$$

4.2 Path Abstraction

According to [1] and [14], we extend the path abstraction for DTMCs as explained in Section 3.3 to paths consisting of rational functions in PMCs.

We first compute the inner strongly connected subsets of a graph in a bottom-up way. We then abstract these strongly connected subsets, i. e., we reduce every subset to its input states. Intuitively, for every nested strongly connected subset in our PMC we generate so-called “abstract” transitions from every input state s_{in} to every output state s_{out} of this subset which carry the whole probability mass of all paths from s_{in} to s_{out}. Therefore we do not consider the structure inside the strongly connected subset anymore and abstract this structure by these abstract transitions. This approach successively leads to a smaller acyclic graph and finally we only have transitions from the input states to the output states where we can easily get the probabilities by reading the abstract transitions. The whole process is visualized in Figure 3.2 of the previous section.

We first define some helpful concepts.
For a set of states $K \subseteq S$ in the PMC $\mathcal{M}$ we define the *input states* of K as

$$Inp^{\mathcal{M}}(K) = \{s \in K \mid \exists s' \in S \setminus K : P(s', s) > 0\} \cup \{s \in K \mid I(s) > 0\}$$

and the *output states* as

$$Out^{\mathcal{M}}(K) = \{s \in S \setminus K \mid \exists s' \in K : P(s', s) > 0\}.$$

The *inner states* of K are

$$K \setminus \mathit{Inp}^{\mathcal{M}}(K).$$

We skip the index $\mathcal{M}$ if it is clear from the context.

We define the PMC $\mathcal{M}_{ind}$ induced by a set of states K, i. e., only states in $K \cup \mathit{Out}^{\mathcal{M}}(K)$ are considered.

Definition 23 (Induced PMC)**.** For a PMC $\mathcal{M} = (S, I, P, V, L)$ and a non-absorbing subset $K \subseteq S$ of states $\mathcal{M}_{ind}^K = (S_{ind}^K, I_{ind}^K, P_{ind}^K, V_{ind}^K, L_{ind}^K)$ is the induced PMC with:

- $S_{ind}^K = K \cup \mathit{Out}^{\mathcal{M}}(K)$
- $I_{ind}^K(s) > 0 \iff s \in \mathit{Inp}\mathcal{M}(K)$, for all $s \in S_{ind}^K$
- $P_{ind}^K(s,t) = \begin{cases} P(s,t), & \text{if } s \in K, t \in S_{ind}^K \\ 1, & \text{if } s = t \in \mathit{Out}^{\mathcal{M}}(K) \\ 0, & \text{otherwise} \end{cases}$
- $V_{ind}^K = V$
- $L_{ind}^K(s) = L(s), \forall s \in S_{ind}^K$

We do not restrict the input probabilities $I_{ind}^K(s)$ for a state s, but only ensure them to be greater than 0. This is similar to the PRISM language [18], where there are also initial states but no input probabilities.

Example 14 (Induced PMC)**.** For the PMC of Example 9 and the subset $K := \{6, 7, 8\}$ the induced PMC $\mathcal{M}_{ind}^K$ is depicted in Figure 4.1.

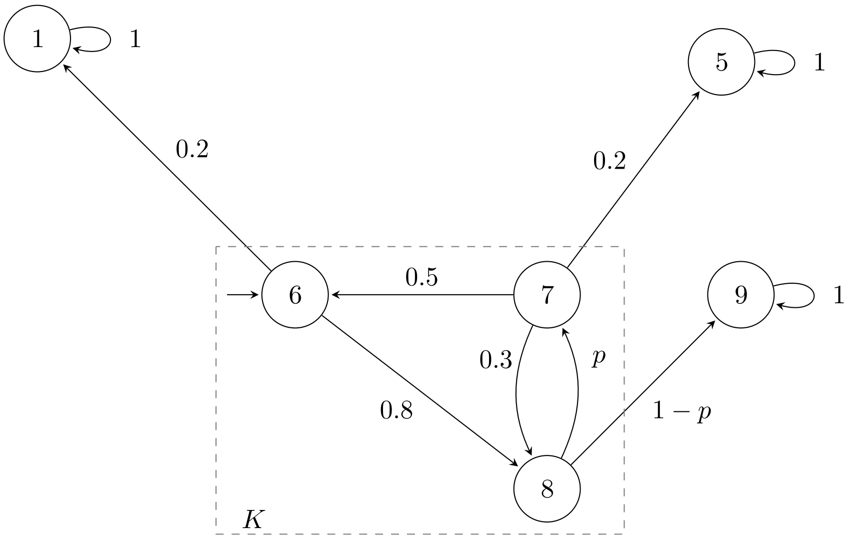


Figure 4.1: Induced PMC $\mathcal{M}_{ind}^K$

Now we want to abstract this induced PMC $\mathcal{M}$ to get a new abstracted PMC $\mathcal{M}_{abs}^K = (S_{abs}^K, I_{abs}^K, P_{abs}^K, V_{abs}^K, L_{abs}^K)$ where only the input and output states, but no inner states of K, are left. We construct this abstract PMC bottom-up and therefore we require K to have no cycles on inner states.

Definition 24 (Abstract PMC)**.** For a PMC $\mathcal{M} = (S, I, P, V, L)$, state set $K \subseteq S$ and the induced PMC $\mathcal{M}^K_{ind} = (S^K_{ind}, I^K_{ind}, P^K_{ind}, V^K_{ind}, L^K_{ind})$ we define the abstract PMC $\mathcal{M}^K_{abs} = (S^K_{abs}, I^K_{abs}, P^K_{abs}, V^K_{abs}, L^K_{abs})$ as

- $S^K_{abs} = \mathit{Inp}^{\mathcal{M}}(K) \cup \mathit{Out}^{\mathcal{M}}(K)$
- $I^K_{abs}(s) = I^K_{ind}(s), \ \forall s \in S^K_{abs}$
- $$P^K_{abs}(s_{in}, s_{out}) = \begin{cases} \dfrac{p^K_{abs}(s_{in}, s_{out})}{\displaystyle\sum_{s'_{out} \in \mathit{Out}(\mathcal{M}^K_{ind})} p^K_{abs}(s_{in}, s'_{out})}, & \text{if } s_{in} \in \mathit{Inp}^{\mathcal{M}}(K), \\ & s_{out} \in \mathit{Out}^{\mathcal{M}}(K) \\ 1, & \text{if } s_{in} = s_{out} \in \mathit{Out}^{\mathcal{M}}(K) \\ 0, & \text{otherwise} \end{cases}$$

  with

  $$p^K_{abs}(s_{in}, s_{out}) = \mathit{Pr}_{\mathit{fin}}(\{\pi = s_{in} s_1 ... s_n s_{out} \in \mathit{Paths}^{\mathcal{M}^K_{ind}}_{\mathit{fin}}(s_{in}, s_{out}) \mid \\ 1 \leq i \leq n : s_i \neq s_{in} \wedge s_i \neq s_{out} \wedge s_i \in K\}). \tag{4.1}$$

- $V^K_{abs} = V^K_{ind}$
- $L^K_{abs}(s) = L^K_{ind}(s), \ \forall s \in S^K_{abs}$.

Example 15 (Abstract PMC)**.** For the induced PMC $\mathcal{M}^K_{abs}$ of Example 14 the abstract PMC $\mathcal{M}^K_{abs}$ is depicted in Figure 4.2. The computation of the abstract transitions probabilities will be explained later.

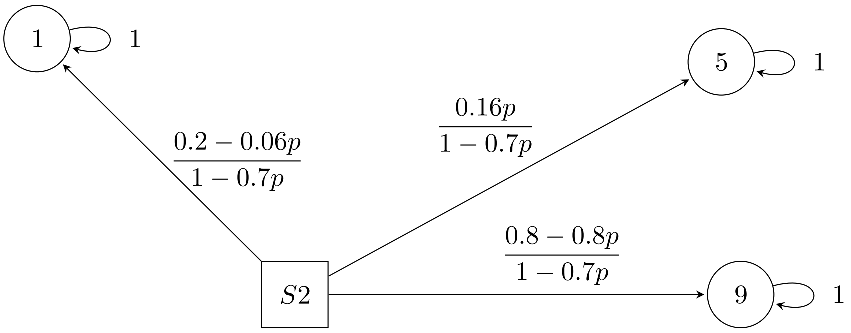


Figure 4.2: Abstracted PMC $\mathcal{M}^K_{abs}$

For an induced PMC $\mathcal{M}^K_{ind}$ an abstract transition $p^K_{abs}(s_{in}, s_{out})$ carries the probability mass of all finite paths starting in s_{in}, ending in s_{out} and not visiting s_{in} in between. Every finite path from s_{in} to s_{out} consists of cycles from s_{in} to s_{in} and finally a path from s_{in} to s_{out} not visiting s_{in} again. Therefore every finite path can be composed by cycling finitely many times and then taking the “direct” path from s_{in} to s_{out}. This was pictured in Figure 3.2(b) in Section 3.3.

The Subset K contains the states of a non-bottom SCC and has therefore no absorbing states. Then every path finally reaches an output state s_{out} with probability 1. We therefore only consider the probabilities of “direct” paths from input states to output states and scale these probabilities to sum up to 1 again.

We scale every transition by the total outgoing probability

$$\sum_{s_{out} \in \mathit{Out}^{\mathcal{M}}(K)} p^K_{abs}(s_{in}, s_{out}). \tag{4.2}$$

Thus, for every state s_{in}, the sum of all rational functions attached to outgoing transitions is 1 and we have a valid PMC.

4.3 Correctness of the Path Abstraction

This construction is correct and does not change the reachability properties. This is stated in the following theorem:

Theorem 1. *For a PMC $\mathcal{M} = (S, I, P, V, L)$, a non-absorbing state set $K \subseteq S$, the induced PMC $\mathcal{M}^K_{ind} = (S^K_{ind}, I^K_{ind}, P^K_{ind}, V^K_{ind}, L^K_{ind})$ and its abstraction $\mathcal{M}^K_{abs} = (S^K_{abs}, I^K_{abs}, P^K_{abs}, V^K_{abs}, L^K_{abs})$ it holds:*

1. *The input states of $\mathcal{M}^K_{ind}$ and $\mathcal{M}^K_{abs}$ coincide.*
2. *$\forall s_{in} \in Inp^{\mathcal{M}}(K), s_{out} \in Out^{\mathcal{M}}(K)$:*
   $Pr^{\mathcal{M}^K_{ind}}_{fin}(Paths^{\mathcal{M}^K_{ind}}_{fin}(s_{in}, s_{out})) = Pr^{\mathcal{M}^K_{abs}}_{fin}(Paths^{\mathcal{M}^K_{abs}}_{fin}(s_{in}, s_{out}))$.

Proof. The first statement directly follows from the construction of the abstract PMC.
For the second statement, K is not absorbing and therefore the probability of eventually reaching an output state of K is 1 as explained before. The probability of a self loop on s_{in} is:

$$p^K_{abs}(s_{in}, s_{in}) = 1 - \sum_{t \in Out^{\mathcal{M}^K_{ind}}(K)} p^K_{abs}(s_{in}, t). \tag{4.3}$$

This self-loop in $\mathcal{M}^K_{abs}$ is the abstract transition for all cycles in $\mathcal{M}^K_{ind}$ which go through s_{in}. Therefore it is

$$p^K_{abs}(s_{in}, s_{in}) = Pr_{fin}(\{s_{in}s_1...s_ns_{in} \in Paths^{\mathcal{M}^K_{ind}}_{fin}(s_{in}, s_{in}) \mid s_i \in K \setminus \{s_{in}\}, 1 \le i \le n\}. \tag{4.4}$$

We define

$$\begin{aligned} R_{in} =& \{s_{in}s_1...s_ns_{in} \in Paths^{\mathcal{M}^K_{ind}}_{fin} \mid s_j \in K \setminus \{s_{in}\}, 1 \le j \le n\} \\ R_{out} =& \{s_{in}s_1...s_ns_{out} \in Paths^{\mathcal{M}^K_{ind}}_{fin} \mid s_j \in K \setminus \{s_{in}\}, 1 \le j \le n\} \end{aligned}$$

Then it is:

$$\begin{aligned}
& Pr^{\mathcal{M}^K_{ind}}_{fin}(Paths^{\mathcal{M}^K_{ind}}_{fin}(s_{in}, s_{out})) \\
= \quad & Pr^{\mathcal{M}^K_{ind}}_{fin}(\bigcup_{i=0}^{\infty}\{\pi_1 \cdot ... \cdot \pi_i \cdot \pi_{i+1} \mid \pi_j \in R_{in}, 1 \le j \le i;\ \pi_{i+1} \in R_{out}\}) \\
= \quad & \sum_{i=0}^{\infty} Pr^{\mathcal{M}^K_{ind}}_{fin}(\{\pi_1 \cdot ... \cdot \pi_i \cdot \pi_{i+1} \mid \pi_j \in R_{in}, 1 \le j \le i;\ \pi_{i+1} \in R_{out}\}) \\
= \quad & \sum_{i=0}^{\infty}(Pr^{\mathcal{M}^K_{ind}}_{fin}(R_{in}))^i \cdot Pr^{\mathcal{M}^K_{ind}}_{fin}(R_{out}) \\
\overset{4.4}{=} \quad & \sum_{i=0}^{\infty}(p^K_{abs}(s_{in}, s_{in}))^i \cdot Pr^{\mathcal{M}^K_{ind}}_{fin}(R_{out}) \\
\overset{\text{geometric series}}{=} \quad & \frac{1}{1 - p^K_{abs}(s_{in}, s_{in})} \cdot Pr^{\mathcal{M}^K_{ind}}_{fin}(R_{out}) \\
\overset{4.3}{=} \quad & \frac{1}{\sum\limits_{t \in Out^{\mathcal{M}}(K)} p^K_{abs}(s_{in}, t)} \cdot Pr^{\mathcal{M}^K_{ind}}_{fin}(R_{out}) \\
\overset{4.1}{=} \quad & \frac{1}{\sum\limits_{t \in Out^{\mathcal{M}}(K)} p^K_{abs}(s_{in}, t)} \cdot p^K_{abs}(s_{in}, s_{out}) \\
\overset{\text{def.}}{=} \quad & P^K_{abs}(s_{in}, s_{out}) \\
= \quad & Pr^{\mathcal{M}^K_{abs}}_{fin}(Paths^{\mathcal{M}^K_{abs}}_{fin}(s_{in}, s_{out}))
\end{aligned}$$

As a result, the second statement of the theorem is correct and our abstraction is valid. □

4.4 Model Checking Algorithm

Now we give an algorithm for model checking PMCs.

For a PMC $\mathcal{M}$ we apply `Model_check` (Algorithm 1). First we search for the set C of SCCs in S. For every SCC K we now search for its inner strongly connected subsets in $K \setminus Inp^{\mathcal{M}}(K)$, i. e., we only consider inner states of K. This goes on recursively until there is no strongly connected subset anymore.

Algorithm 1

```
Model_check(PMC $M$)
begin
    Determine the set $C$ of all non-trivial SCCs of $M$;
    while $C \neq \emptyset$ do
        Choose connected subset $K \in C$;
        $C := C \setminus \{K\}$;
        $M := Abstract(M, K)$;
    end while
    Let $K$ be the set of all non-absorbing states of $M$;
    $M := Abstract(M, K)$;
    Return $M$;
end
```

Then we can compute the abstraction $\mathcal{M}^K_{abs}$ bottom-up, i. e., first we abstract the innermost subset which has no inner strongly connected subsets and therefore no inner cycles. This is done by `Abstract` (Algorithm 2). We gain a first abstraction of this innermost graph which is acyclic and can now consider the outer graph. This goes on until we have the abstraction of $\mathcal{M}$ on the highest level. Here we have no cycles anymore, because every SCC was abstracted. Then we abstract $S \setminus \{s \in S \mid s \text{ is absorbing}\}$ and $\mathcal{M}^K_{abs}$ remains with abstract transitions from the input states of $\mathcal{M}$ to the absorbing states.

Algorithm 2

```
Abstract(PMC $M$, non-absorbing subset $K$)
begin
    $C :=$ set of all non-trivial maximal strongly connected subsets      (1)
         $K'$ of $M$ with $K' \subseteq K \setminus Inp^M(K)$;           (2)
    while $C \neq \emptyset$ do                                            (3)
        Choose $K' \in C$;                                                 (4)
        $C := C \setminus K'$;                                             (5)
        $M := Abstract(M, K')$;                                            (6)
    end while                                                              (7)
    Abstract $M$ and gain $M^K_{abs}$                                      (8)
    Return $M$;                                                            (9)
end
```

One problem still remains: how can we compute the abstract transition probabilities P^K_{abs} in Line 8 of `Abstract` (Algorithm 2)? Here we have to consider the input states $Inp^M(K) = \{s^1_{in}, ..., s^n_{in}\}$ and output states $Out^M(K) = \{s^1_{out}, ..., s^m_{out}\}$. We distinguish three cases for the number of input and output states.

a) **Single output** ($n \geq 1$, $m = 1$):
As the strongly connected Subset K is not absorbing, all paths entering K will leave it eventually as seen in Property 1. That means, the probability of reaching the output state s^1_{out} from an input state s^i_{in} is 1:

$$P^K_{abs}(s^i_{in}, s^1_{out}) = 1, 1 \leq i \leq n. \tag{4.5}$$

The time complexity of this computation is constant: $\mathcal{O}(1)$.

Because of constructing the constraints before, we do not loose any information about our parameters by setting this probability to 1.

b) **Single input/multiple output** ($n = 1$, $m \geq 1$):
To define the abstraction, we need to determine the probabilities $p^K_{abs}(s^1_{in}, s^i_{out})$ again for all $1 \leq i \leq m$. Note that $K \setminus \mathit{Inp}^M(K)$ has no non-trivial strongly connected subsets. Therefore the set

$$R^i_{out} = \{s^1_{in}s_1...s_ns^i_{out} \in \mathit{Paths}^M_{\mathit{fin}}(s^1_{in}, s^i_{out}) \mid s_j \in K \setminus \{s_{in}\}, 1 \leq j \leq n\}$$

contains finitely many loop-free paths. The probability

$$p^K_{abs}(s^1_{in}, s^i_{out}) = \mathit{Pr}^M_{\mathit{fin}}(R^i_{out})$$

can be determined by computing

$$p^K_{abs}(s, s^i_{out}) = \begin{cases} 1, & \text{if } s = s^i_{out} \\ \displaystyle\sum_{s' \in succ(s), s' \neq s^1_{in}} P^K_{ind}(s, s') \cdot p^K_{abs}(s', s^i_{out}), & \text{otherwise} \end{cases}$$

We can compute the probabilities of the abstraction:

1. Compute the probability of all outgoing paths $p^K_{abs}(s^1_{in}, s^i_{out})$ for all $1 \leq i \leq m$.
2. Compute the probabilities from the input state to a output state:

$$P^K_{abs}(s^1_{in}, s^i_{out}) = \frac{p^K_{abs}(s^1_{in}, s^i_{out})}{\sum_{i=1}^m p^K_{abs}(s^1_{in}, s^i_{out})} \text{ for all } 1 \leq i \leq m$$

Computing $p^K_{abs}(s^1_{in}, s^i_{out})$ takes polynomial time for each $1 \leq i \leq m$ depending on the previous abstracted Subset M: $\mathcal{O}(pol_1(M))$. Therefore the first step has a time complexity of $\mathcal{O}(m \cdot pol_1(M))$. The time of the second step depends on the shape and degree of the rational functions and is polynomial again: $\mathcal{O}(pol_2(M, V))$. We finally get a polynomial time complexity for all three steps: $\mathcal{O}(pol_{all}(M, V))$.

c) **Multiple input/multiple output** ($n \geq 1$, $m \geq 1$):
We could use the approach of b) to compute the abstraction for every input state. We would use a copy of the induced PMC for each input state where the other input states become inner states. Finally all abstractions on the copies lead to the whole abstraction. However this would cost too much time, because it has a time complexity of $\mathcal{O}(n \cdot pol_{all}(M, V))$.

Instead we encode the paths in the induced PMC as a linear equation system: for each state $s \in K \cup \mathit{Out}^M(K)$ we introduce a new variable z_s for the linear equation system. These variables are independent from the parameters.

Now we can encode the outgoing transitions for every state $s \in K$ as:

$$z_s = \sum_{s' \in K \cup \mathit{Out}^M(K)} P(s, s') \cdot z_{s'}. \tag{4.6}$$

The probability for each finite path $s_1 ... s_n$ in the induced PMC can now be expressed recursively by considering the probability of the first transition $P(s_1, s_2)$ and the remaining probability z_{s_2}.

Next we eliminate the variables for every inner state $s \in K \setminus \mathit{Inp}^M(K)$ by substituting variables for inner states with their right hand side of Equation 4.6. We have no cycles on inner states and therefore this substitution terminates eventually. Thus we get an equation for every input state s^i_{in} as

$$z_{s^i_{in}} = \sum_{s' \in \mathit{Inp}^M(K) \cup \mathit{Out}^M(K)} (c_{s^i_{in}, s'} \cdot z_{s'}).$$

By eliminating $z_{s^i_{in}}$ from every right-hand side of the equations we finally can express every input state by use of the output states:

$$z_{s^i_{in}} = \sum_{j=1}^{m} c_{s^i_{in}, s^j_{out}} \cdot z_{s^j_{out}}. \tag{4.7}$$

This elimination is possible, because every path eventually reaches one of the output states as stated in Property 1.

As a result we have a solution for each $c_{s^i_{in}, s^j_{out}} = P^K_{abs}(s^i_{in}, s^j_{out})$, which is the probability we needed.

Now we can also compute the abstract probabilities and therefore we refine the `Abstract` function from the previous Algorithm 2 and get the new Algorithm 3.

Algorithm 3

```
Abstract(PMC $M$, non-absorbing subset $K$)
begin
    $C$ := set of all non-trivial maximal strongly connected subsets
        $K'$ of $M$ with $K' \subseteq K \setminus Inp^M(K)$;
    while $C \neq \emptyset$ do
        Choose $K' \in C$;
        $C := C \setminus K'$;
        $M := Abstract(M, K')$;
    end while
    if $|Out^M(K)| = 1$ then
        $M := M^K_{abs}$ by applying method a);
    else if $|Inp^M(K)| = 1$ then
        $M := M^K_{abs}$ by applying method b);
    else
        $M := M^K_{abs}$ by applying method c);
    end if
    Return $M$;
end
```

4.5 Example

We now show an execution of the `Model_check` algorithm on the PMC from Example 9. Here the main concepts of the algorithm and the ideas behind it will become clearer.

Strongly Connected Subsets in the Example We first have to determine the set C of all non-trivial strongly connected subsets of M recursively. This is denoted in Figure 4.3.

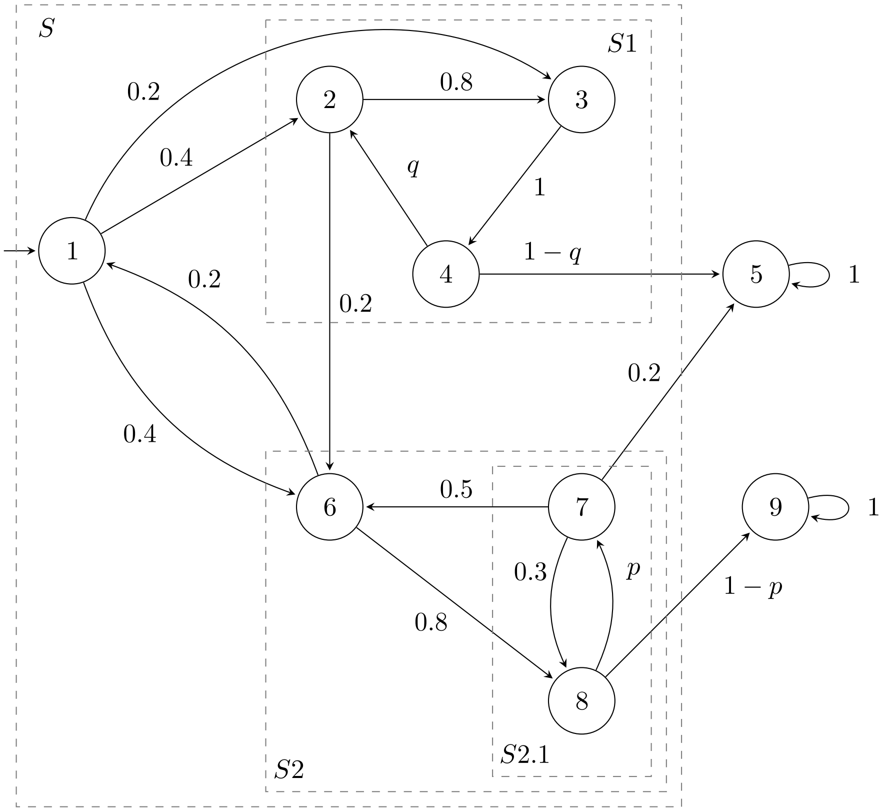


Figure 4.3: Strongly connected subsets of the example PMC

Abstraction of S2.1 The abstraction is bottom-up, i. e., we start with the abstraction of the strongly connected Subset $S2.1$. The induced PMC of Subset 2.1 is visualized in Figure 4.4.

We have one input state $s_{in}^1 = 8$ and three output states $s_{out}^1 = 5$, $s_{out}^2 = 6$, $s_{out}^2 = 9$. We then apply Case b) and first compute the probabilities of all outgoing paths p_{out}^i for $1 \leq i \leq 3$. That yields

$$\begin{aligned} p_{out}^1 &= p_{out}(5) {=} p \cdot 0.2 \\ p_{out}^2 &= p_{out}(6) {=} p \cdot 0.5 \\ p_{out}^3 &= p_{out}(9) {=} 1 - p. \end{aligned}$$

Next p_{in} is computed as

$$p_{in} = 1/(0.2p + 0.5p + 1 - p) = \frac{1}{1 - 0.3p}.$$

Finally we can compute the abstract probabilities:

$$\begin{aligned} P_{abs}^{S2.1}(s_{in}^1, s_{out}^1) &= P_{abs}^{S2.1}(s_{in}(8), s_{out}(5)) {=} \frac{0.2p}{1 - 0.3p} \\ P_{abs}^{S2.1}(s_{in}^1, s_{out}^2) &= P_{abs}^{S2.1}(s_{in}(8), s_{out}(6)) {=} \frac{0.5p}{1 - 0.3p} \\ P_{abs}^{S2.1}(s_{in}^1, s_{out}^3) &= P_{abs}^{S2.1}(s_{in}(8), s_{out}(9)) {=} \frac{1 - p}{1 - 0.3p}. \end{aligned}$$

These abstract probabilities yield the abstraction of $S2.1$ as seen in Figure 4.5.

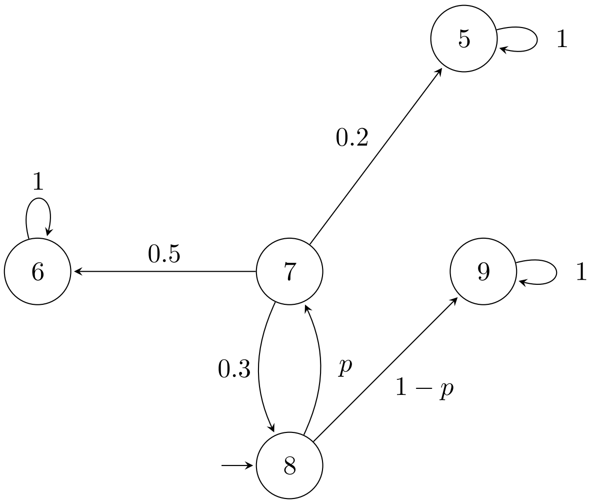


Figure 4.4: Induced PMC $\mathcal{M}\,^{S2.1}_{ind}$ of Subset 2.1

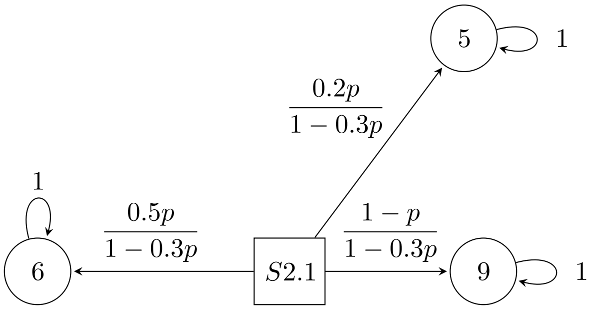


Figure 4.5: Resulting abstract PMC $\mathcal{M}\,^{S2.1}_{abs}$

Abstraction of S2 We have abstracted the Subset $S2.1$ and therefore now have no cycles here anymore. We can now abstract the next strongly connected Subset $S2$ which is depicted in Figure 4.6.

Here the input state is $s^1_{in} = 6$ and there are again three output states $s^1_{out} = 1$, $s^2_{out} = 5$, $s^2_{out} = 9$. Case b) can be applied again and we get p^i_{out} as:

$$\begin{aligned}
p^1_{out} &= p_{out}(1) &&= 0.2\\
p^2_{out} &= p_{out}(5) = 0.8 \cdot \frac{0.2p}{1-0.3p} &&= \frac{0.16p}{1-0.3p}\\
p^3_{out} &= p_{out}(9) = 0.8 \cdot \frac{1-p}{1-0.3p} &&= \frac{0.8-0.8p}{1-0.3p}.
\end{aligned}$$

The scale factor p_{in} is

$$p_{in} = 1/(0.2 + \frac{0.16p}{1-0.3p} + \frac{0.8-0.8p}{1-0.3p}) = 1/(\frac{0.2-0.06p+0.16p+0.8-0.8p}{1-0.3p}) = \frac{1-0.3p}{1-0.7p}.$$

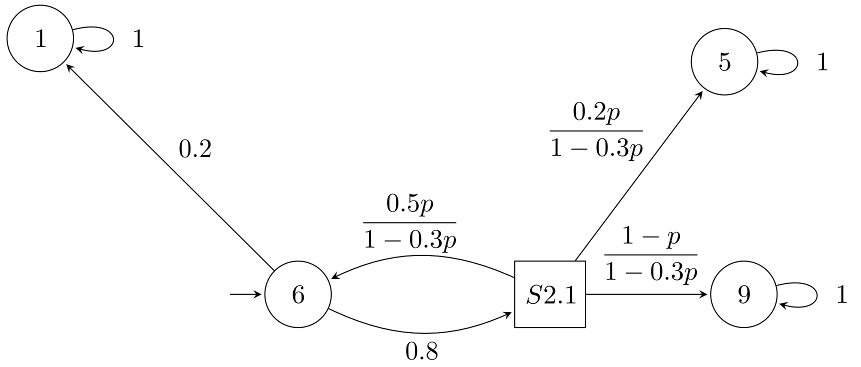


Figure 4.6: Induced PMC $\mathcal{M}^{S2}_{ind}$ of Subset 2

and we get the abstract probabilities:

$$\begin{aligned}
P^{S2}_{abs}(s^1_{in}, s^1_{out}) &= P^{S2}_{abs}(s_{in}(6), s_{out}(1)) && = \frac{0.2-0.06p}{1-0.7p} \\
P^{S2}_{abs}(s^1_{in}, s^2_{out}) &= P^{S2}_{abs}(s_{in}(6), s_{out}(5)) &= \frac{0.16p}{1-0.3p} \cdot \frac{1-0.3p}{1-0.7p} &= \frac{0.16p}{1-0.7p} \\
P^{S2}_{abs}(s^1_{in}, s^3_{out}) &= P^{S2}_{abs}(s_{in}(6), s_{out}(9)) &= \frac{0.8-0.8p}{1-0.3p} \cdot \frac{1-0.3p}{1-0.7p} &= \frac{0.8-0.8p}{1-0.7p}.
\end{aligned}$$

The resulting abstracted PMC can be seen in Figure 4.7.

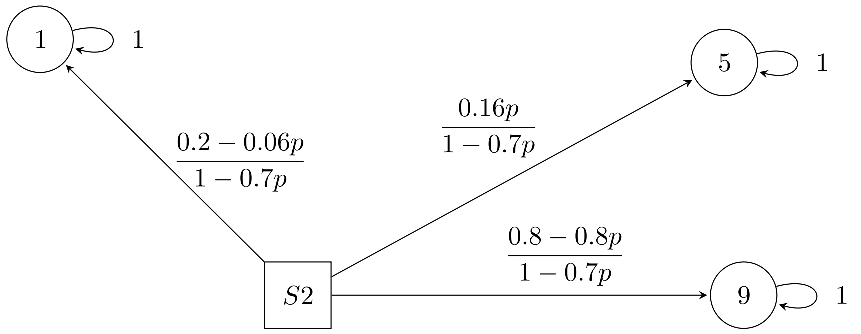


Figure 4.7: Resulting abstract PMC $\mathcal{M}^{S2}_{abs}$

Abstraction of S1 Next we start to abstract the strongly connected Subset $S1$ as seen in Figure 4.8.

Here we now have two input states $s^1_{in} = 2$ and $s^2_{in} = 3$ as well as two output states $s^1_{out} = 5$ and $s^2_{out} = 6$. Because of multiple input states we have to apply the Case c).

So we construct a linear equation system to solve for the abstract probabilities. We introduce the new variables z_2, z_3, z_4, z_5, z_6. For every state in $S1$, that means for $2, 3, 4$, we create a linear equation as follows:

$$z_2 = 0.8 \cdot z_3 + 0.2 \cdot z_6 \tag{4.8}$$
$$z_3 = 1 \cdot z_4 \tag{4.9}$$
$$z_4 = q \cdot z_2 + (1-q) \cdot z_5. \tag{4.10}$$

This linear equation systems is solved for every input state, i. e., for 2 and 3.

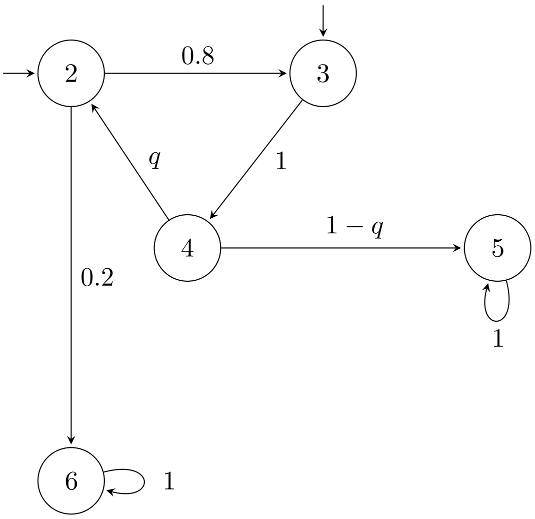


Figure 4.8: Induced PMC $\mathcal{M}_{ind}^{S1}$ of Subset 1

We begin with z_2:

$$\begin{aligned}
z_2 &\overset{4.8}{=} 0.8 \cdot z_3 + 0.2 \cdot z_6 \\
&\overset{4.9}{=} 0.8 \cdot z_4 + 0.2 \cdot z_6 \\
&\overset{4.10}{=} 0.8 \cdot (q \cdot z_2 + (1-q) \cdot z_5) + 0.2 \cdot z_6 \\
&= 0.8q \cdot z_2 + (0.8 - 0.8q) \cdot z_5 + 0.2 \cdot z_6 \\
\iff z_2 - 0.8q \cdot z_2 &= (0.8 - 0.8q) \cdot z_5 + 0.2 \cdot z_6 \\
\iff z_2 &= \frac{1}{1-0.8q} \cdot (0.8 - 0.8q) \cdot z_5 + \frac{1}{1-0.8q} \cdot 0.2 \cdot z_6 .
\end{aligned}$$

Next we solve for z_3:

$$\begin{aligned}
z_3 &\overset{4.9}{=} z_4 \\
&\overset{4.10}{=} q \cdot z_2 + (1-q) \cdot z_5 \\
&\overset{4.8}{=} q \cdot (0.8 \cdot z_3 + 0.2 \cdot z_6) + (1-q) \cdot z_5 \\
&= 0.8q \cdot z_3 + (1-q) \cdot z_5 + 0.2q \cdot z_6 \\
\iff z_3 - 0.8q \cdot z_3 &= (1-q) \cdot z_5 + 0.2q \cdot z_6 \\
\iff z_3 &= \frac{1}{1-0.8q} \cdot (1-q) \cdot z_5 + \frac{1}{1-0.8q} \cdot 0.2q \cdot z_6 .
\end{aligned}$$

Finally we can easily read the abstract probabilities as the coefficients before the variables:

$$\begin{aligned}
P_{abs}^{S1}(s_{in}^1, s_{out}^1) = P_{abs}^{S1}(s_{in}(2), s_{out}(5)) &= \frac{0.8 - 0.8q}{1 - 0.8q} \\
P_{abs}^{S1}(s_{in}^1, s_{out}^2) = P_{abs}^{S1}(s_{in}(2), s_{out}(6)) &= \frac{0.2}{1 - 0.8q} \\
P_{abs}^{S1}(s_{in}^2, s_{out}^1) = P_{abs}^{S1}(s_{in}(3), s_{out}(5)) &= \frac{1 - q}{1 - 0.8q} \\
P_{abs}^{S1}(s_{in}^2, s_{out}^2) = P_{abs}^{S1}(s_{in}(3), s_{out}(6)) &= \frac{0.2q}{1 - 0.8q} .
\end{aligned}$$

These abstract probabilities yield the abstraction of $S1$ as seen in Figure 4.9.

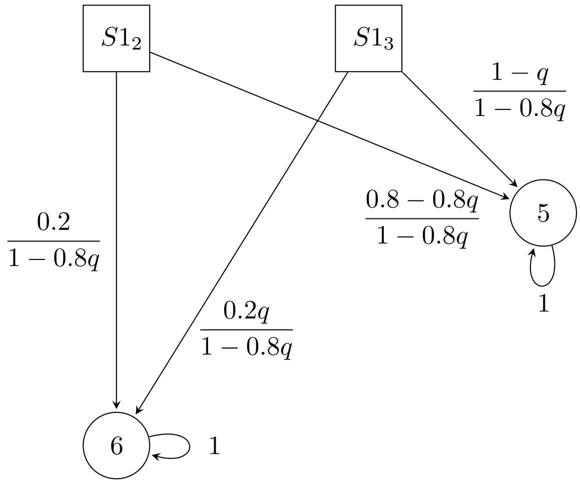


Figure 4.9: Resulting abstract PMC $\mathcal{M}\,^{S1}_{abs}$

Final Abstraction Now we make the final abstraction. Here the initial state 1 is the input state $s^1_{in} = 1$. We further have two absorbing states which are the output states $s^1_{out} = 5$ and $s^2_{out} = 9$. The PMC before the last abstraction is show in Figure 4.10.

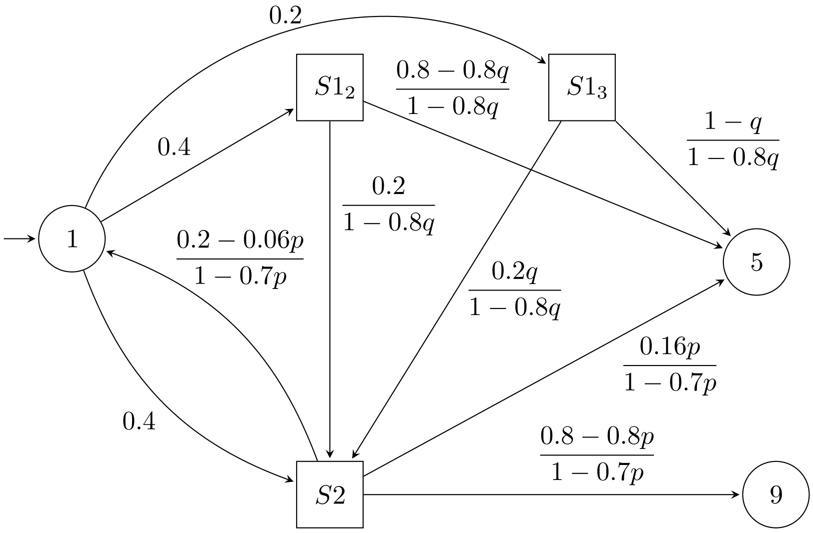


Figure 4.10: PMC $\mathcal{M}\,^{S}_{ind}$before final abstraction

We again apply Case b) and get:

$$\begin{aligned}
p^1_{out} &= p_{out}(5)\\
&= 0.4 \cdot \frac{0.16p}{1-0.7p} + 0.4 \cdot (\frac{0.2}{1-0.8q} \cdot \frac{0.16p}{1-0.7p} + \frac{0.8-0.8q}{1-0.8q}) + 0.2 \cdot (\frac{0.2q}{1-0.8q} \cdot \frac{0.16p}{1-0.7p} + \frac{1-q}{1-0.8q})\\
&= \frac{0.16p}{1-0.7p}(0.4 + \frac{0.08}{1-0.8q} + \frac{0.04q}{1-0.8q}) + \frac{0.32-0.32q}{1-0.8q} + \frac{0.2-0.2q}{1-0.8}\\
&= \frac{0.16p}{1-0.7p} \cdot \frac{0.4-0.32q+0.08+0.04q}{1-0.8q} + \frac{0.52-0.52q}{1-0.8q}\\
&= \frac{0.16p}{1-0.7p} \cdot \frac{0.48-0.28q}{1-0.8q} + \frac{0.52-0.52q}{1-0.8q}\\
p^2_{out} &= p_{out}(9)\\
&= 0.4 \cdot \frac{0.8-0.8p}{1-0.7p} + 0.4 \cdot \frac{0.2}{1-0.8q} \cdot \frac{0.8-0.8p}{1-0.7p} + 0.2 \cdot \frac{0.2q}{1-0.8q} \cdot \frac{0.8-0.8p}{1-0.7p}\\
&= \frac{0.8-0.8p}{1-0.7p} \cdot (0.4 + \frac{0.08}{1-0.8q} + \frac{0.04q}{1-0.8q})\\
&= \frac{0.8-0.8p}{1-0.7p} \cdot \frac{0.4-0.32q+0.08+0.04q}{1-0.8q}\\
&= \frac{0.8-0.8p}{1-0.7p} \cdot \frac{0.48-0.28q}{1-0.8q}.
\end{aligned}$$

We then compute p_{in}:

$$\begin{aligned}
p_{in} &= 1/(p^1_{out} + p^2_{out})\\
&= 1/(\frac{0.16p}{1-0.7p} \cdot \frac{0.48-0.28q}{1-0.8q} + \frac{0.52-0.52q}{1-0.8q} + \frac{0.8-0.8p}{1-0.7p} \cdot \frac{0.48-0.28q}{1-0.8q})\\
&= 1/(\frac{0.48-0.28q}{1-0.8q} \cdot \frac{0.8-0.64p}{1-0.7p} + \frac{0.52-0.52q}{1-0.8q})\\
&= \frac{(1-0.8q)(1-0.7p)}{(0.48-0.28q) \cdot (0.8-0.64p) + (0.52-0.52q)(1-0.7p)}.
\end{aligned}$$

Finally we compute the abstract probabilities:

$$\begin{aligned}
P^S_{abs}(s^1_{in}, s^1_{out}) &= P^S_{abs}(s_{in}(1), s_{out}(5)) = p_{in} \cdot p^1_{out}\\
&= \frac{0.16p \cdot (0.48-0.28q) + (1-0.7p) \cdot (0.52-0.52q)}{(0.48-0.28q) \cdot (0.8-0.64p) + (0.52-0.52q)(1-0.7p)}\\
&= \frac{0.0768p - 0.0448pq + 0.52 - 0.52q - 0.364p + 0.364pq}{0.384 - 0.3072p - 0.224q + 0.1792pq + 0.52 - 0.52 - 0.364p + 0.364pq}\\
&= \frac{-0.2872p - 0.52q + 0.3192pq + 0.52}{-0.6712p - 0.744q + 0.5432pq + 0.904}\\
P^S_{abs}(s^1_{in}, s^2_{out}) &= P^S_{abs}(s_{in}(1), s_{out}(9)) = p_{in} \cdot p^2_{out}\\
&= \frac{(0.8-0.8p) \cdot (0.48-0.28q)}{(0.48-0.28q) \cdot (0.8-0.64p) + (0.52-0.52q)(1-0.7p)}\\
&= \frac{-0.384p - 0.224q + 0.224pq + 0.384}{-0.6712p - 0.744q + 0.5432pq + 0.904}.
\end{aligned}$$

We have the final abstraction of our PMC as in Figure 4.11.

Here we can easily read the model checking results for reaching a target state t as $P^S_{abs}(1, t)$ for $t \in \{5, 9\}$.

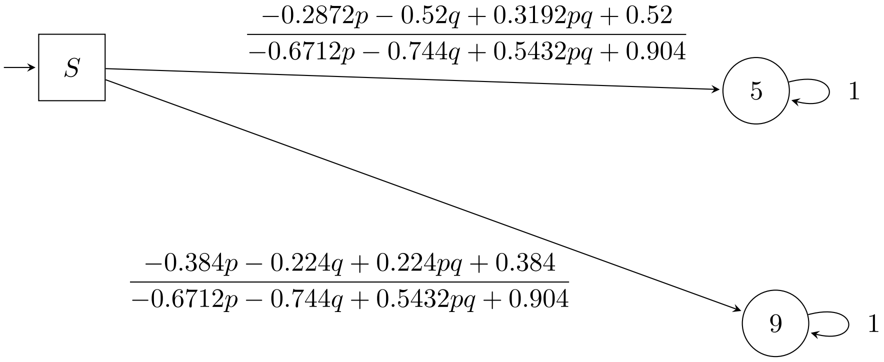

$\frac{-0.2872p-0.52q+0.3192pq+0.52}{-0.6712p-0.744q+0.5432pq+0.904}$
S
5
1
$\frac{-0.384p-0.224q+0.224pq+0.384}{-0.6712p-0.744q+0.5432pq+0.904}$
9
1

Figure 4.11: Final abstract PMC $\mathcal{M}\,^{S}_{abs}$

5 Implementation

5.1 Integration into COMICS

The algorithms `Model_check` as given in Algorithm 1 and `Abstract` as in Algorithm 3 are not only a theoretical idea but also already integrated into the framework of the COMICS tool. This feature will be part of the next release on the COMICS website[1].

COMICS is the abbreviation for *COmputing MInimal CounterexamleS for Discrete-Time Markov Chains* and is presented in [16] and [15]. As already seen in this title the original idea of this tool was to give algorithms for model checking DTMCs and searching for counterexamples. We extended this approach to PMCs and now we can model check both types of Markov Chains.

The COMICS tool consists of two main parts: a command line version written in C++ and a GUI version written in Java. The algorithms all are implemented in C++ ensuring an efficient computation. The Java part only visualizes the graph structure and the resulting model checking probabilities as well as counterexamples for DTMCs. After model checking a DTMC $\mathcal{D}$ we get a probability bound for all infinite paths in $\mathcal{D}$ reaching the set of target states from the initial states. Then we can give a probability bound p which is less or equal than the probability bound of the model checking. The task is now to generate a *counterexample* for p. A counterexample is a set of infinite paths starting in an initial state and reaching a target state with a total probability greater than the probability bound, i. e., it is a counterexample for the formula

$$\Phi = \mathbb{P}_{<p} \Diamond target.$$

Instead of searching for paths, the approach in COMICS is to generate a subset K of the states of the original DTMC $\mathcal{D}$ where all infinite paths in K form a counterexample.

In the GUI of the tool a user can also interactively guide the process of generating such counterexamples. Therefore we can take advantage of the user's expert knowledge or narrow down parts of the graph the user is interested in.

5.1.1 Structure

When compiling the tool it is possible to configure the type of the internal probability representation. At the moment three different types are available. The default type represents probabilities as `double`. This floating-point representation may yield rounding errors and therefore lesser precision. To gain correct and exact values the second type uses the *GNU Multiple Precision Arithmetic Library (GMP)* [7], which computes with arbitrary precision, hence does not lose any precision, but lacks speed. We developed a third type to handle the parametric case where we can also handle parameters as probabilities.

To represent rational functions we use the *GiNaC* library [3] which already has most of the necessary functions to deal with rational functions. For a better encapsulation we wrote our own `Polynomial` and `Rational` classes. In `Polynomial` we use GiNaC's expression class `ex` to hold a polynomial. This allows us to easier gain control over the whole arithmetic computation process of GiNaC.

Additionally we wrote a singleton class `Parameters` to manage everything in the topic of parameters. This class does not only hold all parameters in the PMC but also all ever appearing Polynomial expressions. We therefore hold each expression in the memory only once.

[1] http://www-i2.informatik.rwth-aachen.de/i2/comics/

In the preprocessing of the algorithm we also generate constraints for all transitions as stated in Section 4.1. These constraints later can be used to restrict the bounds of every parameter and therefore also refine the bounds of the resulting model checking probability.

Last we also introduced a special class `AbstractParameter` to store newly introduced abstract transitions. We assign a new variable to every such abstract transition. This will be helpful in the future for the topic of counterexamples. By evaluating the probability for such an abstract transition one could measure its impact on the total probability and therefore decide, whether this transition is necessary to form a certain counterexample.

Lastly, for reading PMCs we developed a new file format `.pmc` similarly to the `.dtmc` format from COMICS. We integrated the Prismparser of PARAM [10] and can read prism files directly now. Especially for loading the later case studies this is very helpful.

5.1.2 Simplifying rational functions

As seen before in Chapter 4 we adopted the algorithms for DTMCs to PMCs. We therefore can also use it for the parametric case and only have to change the type of the probabilities. This is done while compiling. To avoid a blowup of the used rational functions we try to simplify them whenever a new abstract transition is introduced. We therefore assign not only the equation with its underlying abstract transitions, but also the rational function it represents, to every new abstract transition. However this simplification is not enough as internal benchmarks pointed out, because the functions grow too large to be handled efficiently later on. We therefore simplify a rational function after every arithmetic operation, i. e., after addition and multiplication, and can keep the rational functions small.

We have implemented our own algorithms for simplifying rational functions to benefit from the special structure of the occurring rational functions. The main aspect here is the cancellation of the numerator and the denominator of a rational function. Thus, we best need a factorization of the polynomials to easily search for the *Greatest Common Divisor (GCD)* which is used to cancel a rational function. As every rational function is computed in a bottom-up manner from parametric probabilities, we store a factorization of every polynomial as a binary tree. Here every polynomial has two polynomials as factors. This data structure is visualized in Figure 5.1.

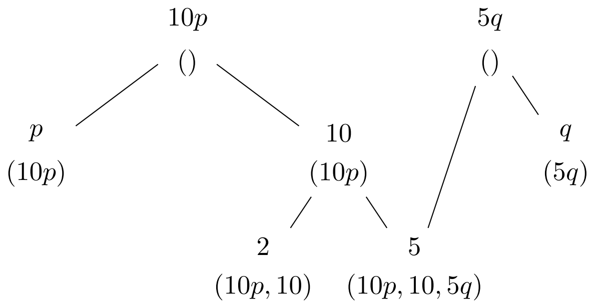


Figure 5.1: Data structure for factorization

For each polynomial we additionally save a list of all multiples, i. e., all parents of this polynomial. E.g., after every multiplication of two polynomials $p_1 \cdot p_2 = p_3$ we store this new polynomial p_3 along with its factors p_1 and p_2. Additionally we traverse the tree to store this new polynomial p_3 as a multiple of p_1, p_2 and their children. Thus we use the bottom-up computation to generate a factorization for our polynomials.

By use of this data structure, the computation of a GCD of two polynomials is linear in size of their factors. We traverse the factorization tree of one polynomial. For every leaf of this tree we search in the corresponding multiple list for the second polynomial. If this search is successful, we have another factor of the GCD.

This GCD then can be used to cancel rational functions. Similarly we can compute the

Least Common Multiple (LCM) to expand two rational functions for addition. Thus, even while computing an addition the resulting rational function is kept small. However, an addition destroys the factorization of a polynomial. Hence, we use GiNaC's implemented GCD function as a fallback to cancel the rational function after performing the addition.

All in all the used simplification via GiNaC costs a lot of time and slows the whole computation process. Another aspect is the high memory consumption due to the storage of all occurring polynomials. Therefore the implementation still leaves room for optimization.

5.2 Case studies

We have done some case studies to show the competitiveness of our tool COMICS and compared it to the tool PARAM [10]. For each case study we list the state count and the number of transitions in the PMC, the total computation time for the model checking algorithm and the memory consumption. For our implementation we also measured the time spent on simplifying the rational functions. We list the time needed for the cancellation algorithms using our specific data structure and the time spent on GiNaC's simplifications as fall-back. The number of stored polynomials is shown as well. For better comparability we do not use bisimulation in PARAM. All case studies were done on a 2.66 GHz Intel Core 2 Quad CPU with 4 GB RAM.

5.2.1 Crowds protocol

The *crowds protocol* [19] should allow anonymous communication in the Internet. This is done via random routing. If a user wants to send a message to a different user, he has two possible actions. He either sends the message directly to the user or to another random user. He then acts as a router by only forwarding the message. An intruder now could not be certain if the message and the sender belong together or if the sender only forwarded the message. Thus the anonymity could be preserved. We have N honest users, M dishonest users and therefore

$$B = \frac{M}{M+N}$$

is the percentage of untrustworthy users. R many messages are sent during the whole process. The users send the message directly to its destination with probability $1 - p_f$ or forward it to a random user with probability p_f. The parameters in our PMC are p_f and B, were N and R are instantiated with concrete values. We are now interested in the probability, that one user is identified more often by a bad member than every other.

The results of this case study can be found in Table 5.2.1.

Table 5.1: Results for the crowds protocol

Param.		Graph		COMICS					PARAM	
N	R	States	Trans.	Cancel (s)	GiNaC (s)	Time (s)	Poly.	Mem. (MB)	Time (s)	Mem. (MB)
2	1	16	18	0	0	0	12	1.43	0	1.36
2	2	77	101	0	0.02	0.02	99	1.43	0.03	1.36
2	3	138	198	0	0.09	0.11	239	1.43	0.10	1.36
3	2	183	243	0.01	0.03	0.07	192	1.43	0.14	1.36
3	3	396	576	0.02	0.26	0.39	515	1.43	0.62	1.36
3	5	1198	2038	0.56	2.44	3.89	1613	2.05	5.58	1.36
5	5	8653	14953	4.28	22.27	33.21	10097	12.24	148.12	5.01

We see that our tool performs better than PARAM on this case study and is faster for greater state spaces. The memory consumption only increases moderately and we can benefit from storing all polynomials here. Notice also, that our own cancellation routine only takes a small amount of time while the fall-back with GiNaC approximately takes 2/3 of the total time. Therefore we should try to replace these algorithms with our own approaches in the future.

It is also interesting to see that the resulting probability bounds seem to have a structure for their denominators as seen in Table 5.2.1.

Table 5.2: Denominator of the probability bounds for the crowds protocol

N	R	Result
2	2	$4 \cdot (1 + B \cdot p_f - p_f)^2$
2	3	$9 \cdot (1 + B \cdot p_f - p_f)^2$
3	2	$8 \cdot (1 + B \cdot p_f - p_f)^3$
3	3	$9 \cdot (1 + B \cdot p_f - p_f)^3$
3	5	$125 \cdot (1 + B \cdot p_f - p_f)^3$
5	5	$625 \cdot (1 + B \cdot p_f - p_f)^5$

Therefore a general denominator likely has the form

$$R^N \cdot (1 + B \cdot p_f - p_f)^N$$

This result could only be seen with our tool, because we try to represent polynomials through their factorization. On the other hand, PARAM expands all polynomials and the resulting rational function becomes unintuitive.

5.2.2 Zeroconf protocol

The zeroconf protocol [4] models the automatic assignment of addresses to hosts in a network. A new host joining the network picks a random number from the set of address numbers of size K. Then the host asks, if other hosts are using this address and waits for answers. With m hosts we have a collision probability

$$q = \frac{m}{K}.$$

In case of a collision the host gets no answer with probability p. In this case the host repeats his question and waits for an answer again. If the host gets no answer within n tries, he will erroneously consider his address as valid. This model is depicted in Figure 5.2.

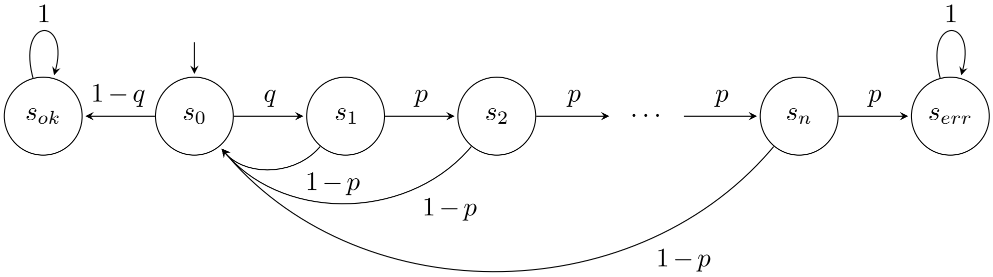


Figure 5.2: Zeroconf protocol

We use p and q as parameters in our PMC and are interested in the probability of eventually reaching a valid state, i. e.,

$$\Phi = \mathbb{P}{<}\ p \Diamond s_{ok}$$

We instantiated n with several values. The results of this case study can be found in Table 5.2.2.

This case study shows that our tool still needs too much memory in specific cases, which slows down the whole computation. Especially in this example we spend nearly no time simplifying rational functions, but still need more than 1GB of memory.

Table 5.3: Results for the zeroconf protocol

Param.	Graph		COMICS					PARAM	
N	States	Trans.	Cancel (s)	GiNaC (s)	Time (s)	Poly.	Mem. (MB)	Time (s)	Mem. (MB)
10	14	25	0	0	0	22	0.25	0	0.19
100	104	205	0	0	0	112	0.57	0	0.23
1000	1004	2005	0	0	0.23	1012	19.73	0.03	0.77
10000	10004	20005	0.01	0.01	55.07	10013	1797.00	0.38	6.47

Table 5.4: Probability bounds for the zeroconf protocol

N	Result
10	$\frac{1-q}{1-q+q\cdot p^{10}}$
100	$\frac{1-q}{1-q+q\cdot p^{100}}$
1000	$\frac{1-q}{1-q+q\cdot p^{1000}}$
10000	$\frac{1-q}{1-q+q\cdot p^{10000}}$

Here the resulting probability bounds are also very intuitively as seen in Table 5.2.2. The general probability of reaching the state s_{ok} for n tries likely has the form

$$\frac{1-q}{1-q+q\cdot p^{N}}$$

5.3 Analyzing the Model Checking Result

After gaining the resulting probability bound as a rational function, we also want to analyze it. We therefore tried to use the SMT solver *dReal* [6], which checks for satisfiability over the reals for a given formula which can be nonlinear. Our implementation can generate the corresponding input format for dReal, such that the user can run the tool afterwards to analyze it. By adding own constraints for the probability bound or the parameter range, the user can use dReal to check, whether there still exists a well-defined evaluation for this restricted case. Thus, this is a first step to a detailed analysis of the model checking result.

6 Conclusion and Future Work

In this thesis we presented our new extension of the SCC-based model checking algorithm for Parametric Markov Chains. This new algorithm for PMCs is very similar to our existing approach for DTMCs and therefore could be well adapted. We proved the correctness of this approach and introduced a set of constraints which comes along with every PMC as seen in Section 4.1. We implemented our algorithm in our tool COMICS and showed it competitiveness with the tool PARAM by two case studies. The main advantage of our tool is the intuitive representation of the rational functions which allows for a better analysis of the resulting probability bounds.

This new model checking approach allows for further improvements, which we will tackle in the future. The implementation can be made more efficiently by improving the existing handling of rational functions. Especially the handling of additions can be optimized to preserve the factorization as much as possible. Further on, we should store the rational functions uniquely as well to not have the same function multiple times. We therefore avoid canceling a rational function more than once. However, this would increase the memory consumption another time. Thus, a better memory management is preferable, where for example unused polynomials are destroyed. These improvements then should make the tool more competitive.

Another interesting topic for the future is the evaluation of the resulting rational functions. At the moment, these functions are only used to instantiate several parameter values to avoid multiple model checking. Fortunately the rational functions carry more information, which we would like to use. As seen in Chapter 1 the resulting functions can also be used to restrict the range of the original parameter values. Therefore we already store the constraints on every transition as seen in Section 4.1. These constraints can later be used to restrict the parameter values and therefore the resulting probability bound. By using the knowledge and interest of the user, we can also restrict the solution space further. Therefore we like the user to restrict interactively certain bounds and compute the new ranges for all the remaining parameters and probability bounds. This approach would strengthen the advantage of PMCs compared to DTMCs.

A further topic could be the generation of counterexamples. For a given probability bound p a *counterexample* is a set of paths which summed up probability is greater or equal than the given probability bound p. Therefore theses paths refute the PCTL formula $\mathbb{P}_{<p}(\Diamond target)$. The goal is to find a preferably small set of states forming such a subgraph. This counterexample then indicates the parts of the Markov Chain which are responsible for unwanted behavior.

For DTMCs exists already a hierarchical counterexample generation. This generation is based on the expansion of the underlying strongly connected subsets of the abstracted graph while model checking.

However before starting to adapt the algorithm, first the concept of counterexamples on PMCs has to be defined. There are two different cases to consider. We can still generate the minimal set of states which forms a counterexample for all parameter instantiation. This would be independent of the concrete parameter values. A second approach would be to lessen the range of the parameters to refuse the given probability bound. This would be a counterexample consisting of the parameters instead of a set of states.

After defining counterexamples, one question would be, if the hierarchical approach could be extended to PMCs as well. This approach searches for the most probable paths, but with rational functions it is not uniquely determined. Therefore we would need to define a relational operator on rational functions to determine which subsystems to consider to form a counterexample. Another approach would be to encode PMCs into SAT formulas and then use SAT solvers to search for the minimal set of states refuting the probability bound.

Concluding there are many opportunities of research in this area in the future. We consider our model checking approach just as the beginning of new insights in the topic of Parametric Markov Chains.

Bibliography

[1] Erika Ábrahám, Nils Jansen, Ralf Wimmer, Joost-Pieter Katoen, and Bernd Becker. DTMC model checking by SCC reduction. In *7th Int. Conf. on Quantitative Evaluation of Systems (QEST'10)*, pages 37–46. IEEE Computer Society, 2010.

[2] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[3] Christian Bauer, Alexander Frink, and Richard Kreckel. Introduction to the GiNaC framework for symbolic computation within the C++ programming language. *J. Symb. Comput.*, 33(1):1–12, 2002.

[4] Henrik Bohnenkamp, Peter Van Der Stok, Holger Hermanns, and Frits Vaandrager. Cost-optimization of the IPv4 zeroconf protocol. In *International Conference on Dependable Systems and Networks (DSN'03*, pages 531–540. IEEE, 2003.

[5] Conrado Daws. Symbolic and Parametric Model Checking of Discrete-Time Markov Chains. In *First Int. Colloquium on Theoretical Aspects of Computing (ICTAC'04)*, volume 3407 of *Lecture Notes in Computer Science*, pages 280–294. Springer, 2004.

[6] dReal solver - Website, 2012. http://www.cs.cmu.edu/~sicung/dReal/.

[7] GNU Multiple Precision Arithmetic Library - Website, 2012. http://gmplib.org/.

[8] Friedrich Gretz, Joost-Pieter Katoen, and Annabelle McIver. Operational versus weakest precondition semantics for the probabilistic guarded command language. In *9th Int. Conf. on Quantitative Evaluation of Systems (QEST'12)*. IEEE Computer Society, 2012.

[9] Hermann Gruber and Jan Johannsen. Optimal lower bounds on regular expression size using communication complexity. In *Foundations of Software Science and Computation Structures*, pages 273–286. Springer, 2008.

[10] Ernst Moritz Hahn, Holger Hermanns, Björn Wachter, and Lijun Zhang. PARAM: A Model Checker for Parametric Markov Models. In *CAV*, pages 660–664, 2010.

[11] Ernst Moritz Hahn, Holger Hermanns, and Lijun Zhang. Probabilistic reachability for parametric Markov models. *STTT*, pages 1–17, 2010.

[12] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[13] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation, Second Edition*. Addison-Wesley, 2000.

[14] Nils Jansen, Erika Ábrahám, Jens Katelaan, Ralf Wimmer, Joost-Pieter Katoen, and Bernd Becker. Hierarchical counterexamples for discrete-time Markov chains. In *9th Int. Symp. on Automated Technology for Verification and Analysis (ATVA'11)*, volume 6996 of *Lecture Notes in Computer Science*, pages 443–452. Springer, 2011.

[15] Nils Jansen, Erika Ábrahám, Maik Scheffler, Matthias Volk, Andreas Vorpahl, Ralf Wimmer, Joost-Pieter Katoen, and Bernd Becker. The COMICS tool - Computing minimal counterexamples for discrete-time Markov chains. *CoRR*, abs/1206.0603, 2012.

[16] Nils Jansen, Erika Ábrahám, Matthias Volk, Ralf Wimmer, Joost-Pieter Katoen, and Bernd Becker. The COMICS tool - Computing minimal counterexamples for DTMCs. In *10th Int. Symp. on Automated Technology for Verification and Analysis (ATVA'12)*, volume 7561 of *Lecture Notes in Computer Science*, pages 349–353. Springer, October 2012. (to appear).

[17] Joost-Pieter Katoen, Ivan S. Zapreev, Ernst Moritz Hahn, Holger Hermanns, and David N. Jansen. The ins and outs of the probabilistic model checker MRMC. In *6th Int. Conf. on Quantitative Evaluation of Systems (QEST'09)*, pages 167–176. IEEE Computer Society, 2009. www.mrmc-tool.org.

[18] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011.

[19] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, 1998.