

Efficiently Computing Poisson Probabilities for Model Checking Continuous-time Markov Chains

Rheinisch–Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik 2

Bachelor Thesis
Phillip Florian

Reviewers:
Prof. Dr. Joost-Pieter Katoen
Prof. Dr. Erika Ábrahám

Statement

Hiermit erkläre ich, Phillip Florian, an Eides statt, dass ich die vorliegende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Aachen, December 22, 2014

Phillip Florian

Abstract

Whenever requirements of a system need to be analysed, it is useful to automatically verify these requirements.

This bachelor thesis explains how the verification of requirements for systems that can be modelled by Continuous-time Markov Chains is performed. It will also present an algorithm for efficiently computing poisson probabilities by Fox and Glynn [6] and give detailed proofs for this algorithm.

Furthermore, a way to improve the computation-time for a CTMC Model Checker additionally to the efficient computation of poisson probabilities will be given.

The practical part of this Bachelor thesis is the implementation of A CTMC model checker with the introduced optimizations for the StoRM project which then will be benchmarked against the PRISM Model Checker by Oxford University [11] and against MRMC by RWTH Aachen [9] with respect to memory and runtime requirements.

Contents

1	Introduction	1
2	Preliminaries	3
2.1	Basics of Measure Theory	3
2.2	Model Checking of Discrete-Time Markov Chains	4
2.2.1	Discrete-Time Markov Chain	4
2.2.2	Reachability Probabilities	5
2.2.3	Probabilistic Computation Tree Logic	8
2.2.4	DTMC vs. PCTL Model Checking	8
2.3	Exponential distribution	10
3	Continuous-Time Markov Chain Model Checking	11
3.1	Continuous-time Markov Chain	11
3.2	Timed reachability probabilities	13
3.3	Computing transient probabilities	14
3.4	Efficient computation of reachability probabilities	18
3.5	Continuous Stochastic Logic	19
3.6	CTMC vs CSL Model Checking	19
4	Efficient computation of poisson probabilities	22
4.1	Intuition behind the Fox Glynn algorithm	23
4.2	Finding the truncation points	23
4.3	Improvement of CTMC Model Checking by usage of Fox & Glynn's algorithm	25
5	Implementation	26
6	Benchmark against MRMC and Prism	28
7	Conclusion	31
8	Proofs	32

1 Introduction

In a time where stochastic systems become more and more complex people want to be able to model such systems in an abstract way and get algorithms to verify certain behaviour of those models. This is where probabilistic model checking comes into play. But what is probabilistic model checking?

In probabilistic model checking you basically take a system and some requirement and want to know whether the system satisfies the requirements and with which probability those requirements are fulfilled. As seen in the graphic below the requirements and the system need to be changed in a way that the model checker is able to deal with the problem. For probabilistic model checking the model checker is also bound to a certain precision as some computations in such a model checker use approximations for a lower runtime.

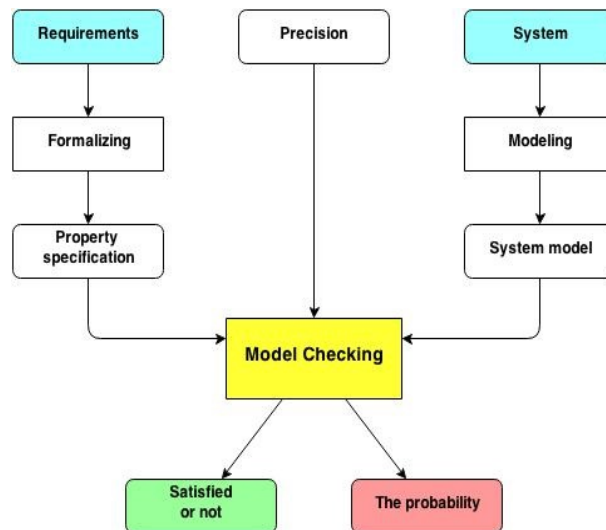


Figure 1.1: Requirements and results for model checking

This Bachelor thesis deals with model checking Continuous-Time Markov chains (CTMC). In practice CTMCs are used in many areas for performance analysis. For example they can be used for calculating the behaviour of enzymes in biological systems [16], checking properties for stochastic activity networks [13], stochastic petri nets [12], stochastic

1 Introduction

process algebra [3] and many more. Because there are those many applications a CTMC model checker with a low runtime is quite important.

Also as the computation of poisson probabilities is essential for a CTMC model checking, this thesis will explain how to efficiently compute poisson probabilities by using the algorithm of Fox and Glynn [6]. Further this thesis will show an other way to optimize to the general algorithms for CTMC model checking. The practical part of this thesis is an implementation of a CTMC model checker for the StoRM projekt of RWTH-Aachen c.s. chair I2 including all mentioned improvements. This implementation will then be benchmarked against the former project of chair I2, the MRMC [9] as well as against the PRISM model checker from Oxford [11].

2 Preliminaries

The preliminaries will give a brief introduction to measure theory and later on explain what a Discrete-Time Markov Chain (DTMC) is and how model checking on an DTMC is performed and afterwards there will be an introduction to exponential distributions before model checking for Continuous-time Markov Chains will be explained.

2.1 Basics of Measure Theory

As this thesis deals with probabilistic measurements of events in a CTMC or DTMC, we are going to deal with problems, where we have to be able to compute the outcomes of certain events. Due to this we need at least some basics of measure theory which will be quite briefly explained in this section.

Definition 2.1.1.

- A **sample space** Ω consisting of all possible outcomes of the experiment.
- A set of **events** \mathcal{F} where each event is a set containing zero or more outcomes of the experiment.
- A **σ -algebra** which is a pair (Ω, \mathcal{F}) with $\Omega \neq \emptyset$ and $\mathcal{F} \subseteq 2^\Omega$ a collection of subsets of sample spaces Ω such that $\Omega \in \mathcal{F}$, where \mathcal{F} is closed under complement e.g. $A \in \mathcal{F} \Rightarrow \Omega - A \in \mathcal{F}$, and \mathcal{F} is also closed under countable union e.g. $(\forall i \geq 0. A_i \in \mathcal{F}) \Rightarrow \cup_{i \geq 0} A_i \in \mathcal{F}$
- A **probability function** assigning probabilities to events. $Pr : \mathcal{F} \rightarrow [0, 1]$ where Ω is the certain event and $Pr\{\cup_{i \in I} A_i\} = \sum_{i \in I} Pr\{A_i\}$ for any $A_i \in \mathcal{F}$ with $A_i \cap A_j = \emptyset, \forall i \neq j$.
- A **probability space** \mathcal{P} that is a structure $(\Omega, \mathcal{F}, Pr)$, where (Ω, \mathcal{F}) is a σ -algebra.

The events in \mathcal{F} of a probability space $(\Omega, \mathcal{F}, Pr)$ are called measurable [2].

Example:

Lets take the simple example of throwing two distinguishable fair coins. The possible outcomes are defined by $\Omega = \{(H, H), (H, T), (T, H), (T, T)\}$, where H means the outcome of the corresponding throw was head and T means it was tails. Now let $\mathcal{F} = 2^\Omega$, and take that all outcomes have the same probability $Pr\{(X, Y)\} = \frac{1}{4}$, $X, Y \in \{H, T\}$. If we now want to compute the probability of throwing the same outcome on both coins we get:

$$\begin{aligned} Pr\{(H, H), (T, T)\} &= Pr\{(H, H)\} + Pr\{(T, T)\} \\ &= \frac{1}{4} + \frac{1}{4} = \frac{1}{2} \end{aligned}$$

2.2 Model Checking of Discrete-Time Markov Chains

Discrete-time Markov chains are one of the most basic probabilistic models and are needed for model checking Continuous-time Markov Chains. Thus this section will explain what a DTMC is, how properties are formalized and how model checking of a DTMC is performed.

2.2.1 Discrete-Time Markov Chain

A Discrete-Time Markov Chain is a deterministic transition-system augmented with probabilities where transitions are taken after discrete time steps. So Discrete-Time Markov Chains can be used to model scenarios where events take place with a certain probability after a discrete amount of time or where the time is not of interest. [2]

Definition 2.2.1 (Discrete-time Markov chain).

A DTMC \mathcal{D} is a tuple $(S, \mathbf{P}, \nu_{init}, AP, L)$ where:

- S is a countable non-empty set of states.
- \mathbf{P} is a stochastic transition probability matrix s.t. $\sum_{s'} \mathbf{P}(s, s') = 1$.
- $\nu_{init} : S \rightarrow [0, 1]$, is the initial distribution with $\sum_s \nu_{init} = 1$.
- AP is a set of atomic propositions.
- $L : S \rightarrow 2^{AP}$, is the labelling function, assigning to state s , the set $L(s)$ of atomic propositions that are valid in s .

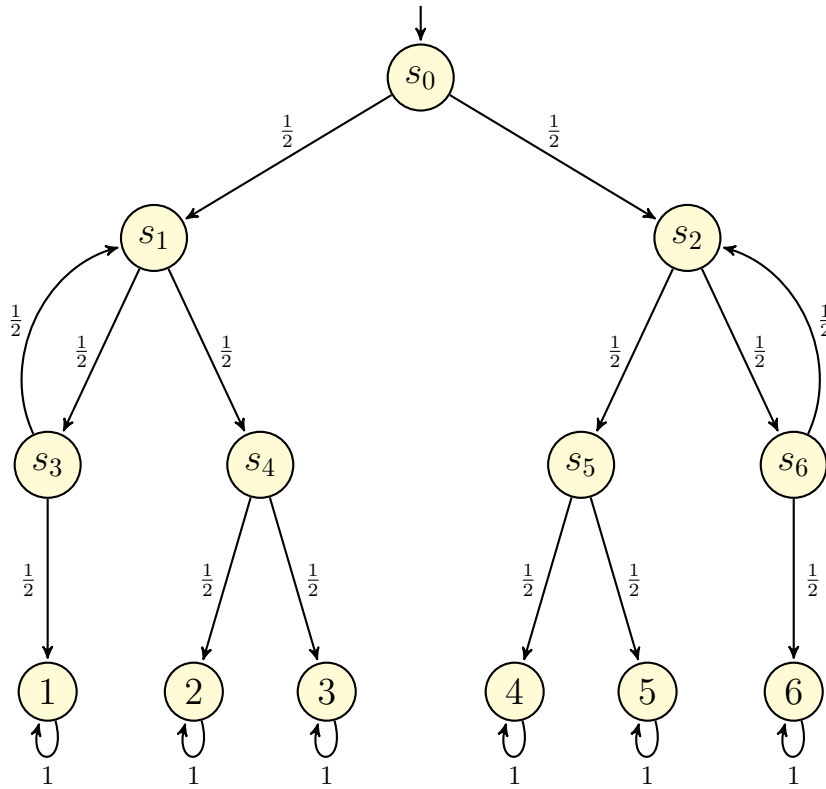


Figure 2.1: Simulating a die by a fair coin [Knuth & Yao]

Example:

Assume you want to play a game and need a die, but only have a fair coin. Then you can simulate a die using the coin like stated in the DTMC in Figure 2.1, where heads means "go left" and tails means "go right". In this DTMC the states '1' to '6' represent the outcome of the die throw.

2.2.2 Reachability Probabilities

As we want to be able to prove that the DTMC shown in Figure 2.1 really simulates a fair die we need to know how to compute the probability of reaching certain states. Looking at Figure 2.1. we want to be able to compute the probabilities of reaching the states 1 to 6 from the initial state s_0 .

In general we are interested in events like reaching a state s with a certain probability. To formalize these events we first need to introduce paths within a DTMC.

2 Preliminaries

Definition 2.2.2 (Paths in a DTMC).

Paths in a DTMC \mathcal{D} are maximal paths of state instants denoted as $\pi = s_0, s_1, s_2 \dots$ where $s_i \in S$ and $\mathbf{P}(s_i, s_{i+1}) > 0$.

$Paths(\mathcal{D})$ denotes the set of all infinite paths $\pi = s_0 s_1 s_2 \dots$, $s_i \in S$, $\mathbf{P}(s_i, s_{i+1}) > 0 \forall i$ in DTMC \mathcal{D} . Further, $Paths_{\mathcal{D}}(s)$ denotes the set of infinite paths in \mathcal{D} that start in s .

Using paths we are able to define the event of eventually reaching a set of target states G . Therefore we introduce the reachability event denoted $\diamond G$, $G \subseteq S$, where $\diamond G$ means eventually reach a state $s \in G$. Formally: $\diamond G = \{\pi = s_0, s_1, \dots \in Paths(\mathcal{D}) \mid \exists i \in \mathbb{N}. s_i \in G\}$. The other important event for now is the restricted reachability event $\overline{F} \mathbf{U} G$, $\overline{F}, G \subseteq S$. Intuitively this operator means 'do not visit a F state until you visit a G state'. Formally: $\overline{F} \mathbf{U} G = \{\pi = s_0, s_1, \dots \in Paths(\mathcal{D}) \mid \exists i \in \mathbb{N}. s_i \in G, \forall n < i. s_n \notin F\}$. [10]

The probabilities of events in a DTMC \mathcal{D} are measured by using the cylinder sets of \mathcal{D} .

Definition 2.2.3 (Cylinder set).

The cylinder set of a finite path $\hat{\pi} = s_0 s_1 \dots \in pref(Paths(\mathcal{D}))$ consists of all infinite paths that have prefix $\hat{\pi}$. It is formally defined by:

$$Cyl(\hat{\pi}) = \{\pi \in Paths(\mathcal{D}) \mid \hat{\pi} \text{ is a prefix of } \pi\}$$

The probability measure of such a cylinder set is defined by [2]:

$Pr\{Cyl(s_0 \dots s_n)\} = \iota_{init}(s_0) \cdot \mathbf{P}(s_0 s_1 \dots s_n)$, where $\mathbf{P}(s_0 s_1 \dots s_n) = \prod_{0 \leq i < n} \mathbf{P}(s_i, s_{i+1})$ for $n > 0$ and $\mathbf{P}(s_0) = \iota_{init}(s_0)$.

Using cylinder sets we are now able to compute the probabilities of the introduced events. It is easy to show that the events can be represented by cylinder sets that are pairwise disjoint and therefore the probability for the reachability event can be defined by:

$$\begin{aligned} Pr(\diamond G) &= \sum_{s_0 \dots s_n \in pref(Paths(\mathcal{D})) \cap (S \setminus G)^* \cdot G} Pr(Cyl(s_0 \dots s_n)) \\ &= \sum_{s_0 \dots s_n \in pref(Paths(\mathcal{D})) \cap (S \setminus G)^* \cdot G} \iota_{init}(s_0) \cdot \mathbf{P}(s_0 \dots s_n) \end{aligned}$$

$Pr(\overline{F} \mathbf{U} G)$ is defined analogically.

$$\begin{aligned} Pr(\overline{F} \mathbf{U} G) &= \sum_{s_0 \dots s_n \in pref(Paths(\mathcal{D})) \cap (S \setminus (G \cup \overline{F}))^* \cdot G} Pr(Cyl(s_0 \dots s_n)) \\ &= \sum_{s_0 \dots s_n \in pref(Paths(\mathcal{D})) \cap (S \setminus (G \cup \overline{F}))^* \cdot G} \iota_{init}(s_0) \cdot \mathbf{P}(s_0 \dots s_n) \end{aligned}$$

Example:

Now we can show that the DTMC in Figure 2.1 really models a 6-sided die by usage of a fair coin. To do so, compute the probabilities for $\diamond 1$ and $\diamond 2$. It can be shown that this suffices every path in $\diamond 1$ has a symmetric path in $\diamond 6$ and for all paths in $\diamond i$ $i \in \{2, 3, 4, 5\}$ there exists a symmetric path in the other sets as well.

First lets compute $Pr(\diamond 1)$.

By applying the definition of Pr as stated above we get:

$$\begin{aligned}
 Pr(\diamond 1) &= \sum_{s_0 \dots s_n \in ((S \setminus 1)^*)1} \mathbf{P}(s_0 \dots s_n) \\
 &= \mathbf{P}(s_0 s_1 s_3 1) + \mathbf{P}(s_0 s_1 s_3 s_1 s_3 1) + \dots \\
 &= \sum_{k=0}^{\infty} \mathbf{P}(s_0 s_1 (s_3 s_1)^k s_3 1) \\
 &= \frac{1}{8} \cdot \sum_{k=0}^{\infty} \left(\frac{1}{4}\right)^k \\
 &= \frac{1}{8} \cdot \frac{1}{1 - \frac{1}{4}} \\
 &= \frac{1}{6}
 \end{aligned}$$

and for $\diamond 2$ we get:

$$\begin{aligned}
 Pr(\diamond 2) &= \sum_{s_0 \dots s_n \in ((S \setminus 2)^*)2} \mathbf{P}(s_0 \dots s_n) \\
 &= \mathbf{P}(s_0 s_1 s_4 2) + \mathbf{P}(s_0 s_1 s_3 s_1 s_4 2) + \dots \\
 &= \sum_{k=0}^{\infty} \mathbf{P}(s_0 s_1 (s_3 s_1)^k s_4 2) \\
 &= \frac{1}{8} \cdot \sum_{k=0}^{\infty} \left(\frac{1}{4}\right)^k \\
 &= \frac{1}{8} \cdot \frac{1}{1 - \frac{1}{4}} \\
 &= \frac{1}{6}
 \end{aligned}$$

As the sets of paths are symmetric as stated above it follows that $Pr(\diamond i) = \frac{1}{6}$, $\forall i \in \{1, \dots, 6\}$ and hence we have shown that the DTMC models a 6-sided die by fair coin. [2]

2.2.3 Probabilistic Computation Tree Logic

The Probabilistic Computation Tree Logic is a temporal logic used for formally specifying properties over DTMCs [2]. A state s in a DTMC can either satisfy a formula Φ (denoted $s \models \Phi$) or not.

Definition 2.2.4 (Probabilistic Computation Tree Logic).

Property specifications for a DTMC can be formalized by usage of state-formulas and path-formulas. They are defined as follows:

- PCTL state formulas over the set AP are formed according to the grammar:

$$\Phi ::= \text{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid P_J(\varphi)$$

where $a \in AP$, φ is a path formula and $J \subseteq [0, 1]$, $J \neq \emptyset$ is a non-empty interval. For J notations like < 0.5 and ≥ 0.3 can be used since they can be rewritten into intervals.

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

$$\varphi ::= \bigcirc\Phi \mid \Phi_1 \mathbf{U} \Phi_2 \mid \Phi_1 \mathbf{U}^{\leq n} \Phi_2$$

where Φ_1 and Φ_2 are state formulas and $n \in \mathbb{N}$.

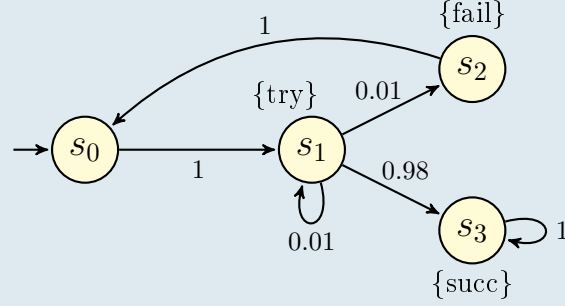
$\bigcirc\Phi$ means in the successor state Φ holds. $\Phi \mathbf{U}^{\leq n} \Psi$ means Φ holds until Ψ holds and Ψ holds within n steps. This thesis does not cover the computation of probabilities for 'step-bounded until' and the 'next' property.

2.2.4 DTMC vs. PCTL Model Checking

We now want to check whether or not a state $s \in S$ of a finite DTMC \mathcal{D} satisfies a PCTL state formula Φ . In order to check whether $s \models \Phi$ we compute the satisfaction set $Sat(\Phi) = \{s \in S \mid s \models \Phi\}$. This is done recursively by a bottom-up traversal of Φ 's sub-formulas. If $s \in Sat(\Phi)$ it follows $s \models \Phi$. In order to determine whether $s \in Sat(\mathbb{P}_J(\varphi))$, the probability $Pr\{s \models \varphi\}$ for the path formula φ needs to be computed [2].

Example:

(Example from [10])

 Lets take a look at the DTMC shown below. and check the PCTL formula $\Phi = P_{\geq 0.9}(\bigcirc(\neg try \vee succ)) \equiv P_{\geq 0.9}(\bigcirc(\neg(try \wedge \neg succ)))$

 Now lets check $s \models \Phi$. Therefore we have to compute the satisfaction set of Φ . As explained before this is done by a bottom-up traversal over the sub formulas of Φ . Therefore:

$$\begin{aligned}
 Sat(try) &= \{s_1\} \\
 Sat(succ) &= \{s_3\} \\
 Sat(\neg try) &= \{s_0, s_2, s_3\} \\
 Sat(\neg try \vee succ) &= \{s_0, s_2, s_3\}
 \end{aligned}$$

 Now we need to compute $Pr(\bigcirc(\neg try \vee succ))$. Therefore compute the probability of reaching a state in $Sat(\neg try \vee succ)$ within one step.

$$\begin{aligned}
 \underline{Pr}(\bigcirc(\neg try \vee succ)) &= \mathbf{P} \cdot \underline{(\neg try \vee succ)} \\
 &= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.098 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0.99 \\ 1 \\ 1 \end{pmatrix}
 \end{aligned}$$

Looking at the resulting vector, we can now determine

$$Sat(\Phi) = \{s_1, s_2, s_3\}$$

 As the initial state $s_0 \notin Sat(\Phi)$ this DTMC does not satisfy Φ .

2.3 Exponential distribution

As exponential distributions are very important to understand the behaviour of a CTMC this section will give a short introduction to exponential distributions.

In general, exponential distributions are adequate for describing many real-life phenomena like the time until a radioactive particle decays [5] or the time between mutations on a DNA strand [14].

Definition 2.3.1 (Density of a exponential distribution). *The density of an exponentially distributed random variable Y with rate $\lambda \in \mathbb{R}_{>0}$ is:*

$$f_Y(x) = \lambda \cdot e^{-\lambda \cdot x} \text{ for } x > 0 \text{ and } f_Y(x) = 0 \text{ otherwise.}$$

The cumulative distribution of random variable Y with rate $\lambda \in \mathbb{R}_{>0}$ is:

$$F_Y(d) = \int_0^d \lambda \cdot e^{-\lambda \cdot x} dx = 1 - e^{-\lambda \cdot d}.$$

As seen in Figure 2.2 a higher parameter λ leads to a faster approach of the cumulative distribution towards 1

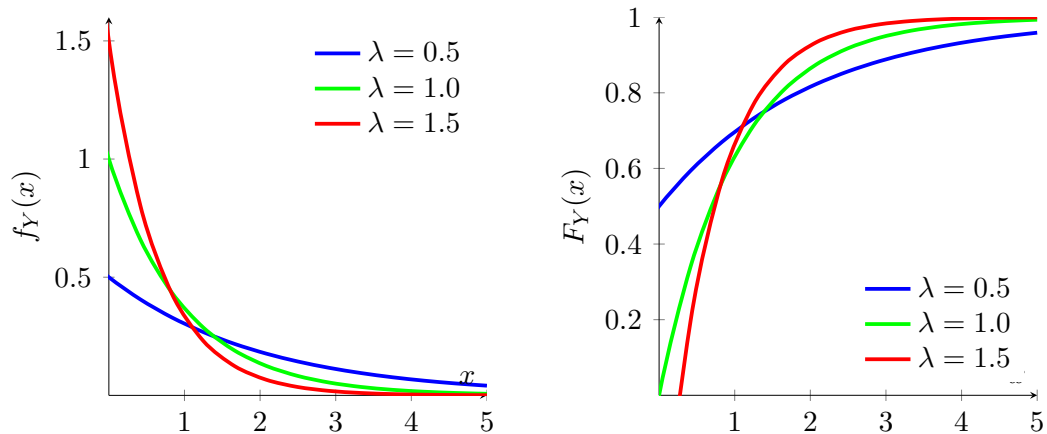


Figure 2.2: Grafical representation of $f_Y(x)$ (left) and $F_Y(x)$ (right).

3 Continuous-Time Markov Chain Model Checking

As mentioned before CTMCs are used to model and verify a large class of stochastic systems. Therefore, this section will explain what a CTMC is and how it behaves with respect to time. Additionally, it will be shown how properties are formalized and how to verify whether a CTMC satisfies these properties.

3.1 Continuous-time Markov Chain

Many real-life systems deal with a continuous flow of time, where changes can appear at any given point in time. For example, chemical reactions [5] or biological pathways [14], where an event is not guaranteed within a time-bound, but it gets more and more likely to occur if the observed interval of time is increased. As changes within a DTMC only occur after discrete time-steps they are not fit for modelling such systems, but Continuous-time Markov Chains (CTMC) can be used to model such behaviour.

A Continuous-time Markov Chain can be seen as an extension to DTMCs where transitions are not taken after a discrete time step. Instead the probability of taking an outgoing transition depends on the time spent in this state.

Definition 3.1.1 (Continuous-time Markov Chain).

A CTMC \mathcal{C} is a tuple $(S, \mathbf{P}, r, \iota_{init}, AP, L)$ with:

- $(S, \mathbf{P}, \iota_{init}, AP, L)$ is a DTMC.
- $r : S \rightarrow \mathbb{R}_{\geq 0}$, the exit-rate function

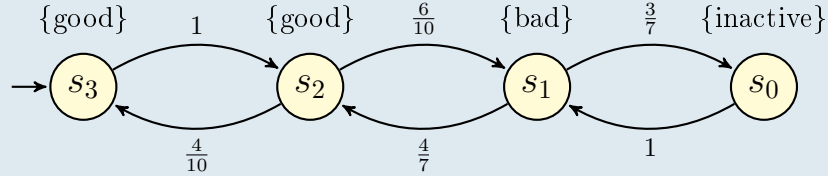
Further, $\mathbf{R}(s, s') = \mathbf{P}(s, s') \cdot r(s)$ is the rate-matrix of the CTMC \mathcal{C}

In a CTMC the residence time in state s is exponentially distributed with rate $r(s)$. In other words, the higher the rate of a state s is, the higher the probability that some outgoing transition of s will be taken in a fixed interval of time.

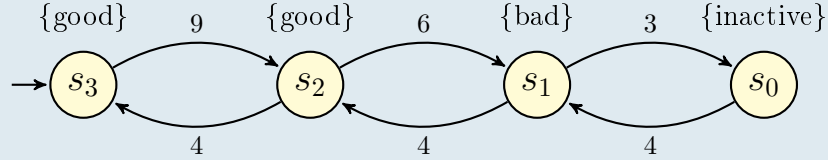
Note that the definition of a CTMC varies from paper to paper since a CTMC can be described either by giving a probability matrix and an exit-rate vector, as done here, or by giving a rate matrix instead of the probability matrix.

Example:

A factory owns three identical machines which each fail with rate 3. The fabric owns only a single mechanic who is able to repair a failed machine with rate 4. This scenario is modelled by the following CTMC, where in state s_i exactly i machines are working and the labeling describes how well the system is running. In this representation the exit-rates are written next to the corresponding state.



An alternative the transitions can be labelled with the corresponding rates as shown below



As mentioned before, the residence time in a state s is exponentially distributed with rate $r(s)$. We now are interested in the probabilities of leaving a state s within an interval of time $[0, t]$ and the probability of going from state s to a successor state s' within the interval of time $[0, t]$ [10].

Lemma 3.1.1 (Residence time distribution). *The probability to take any outgoing transition of s in $[0, t]$ is:*

$$\int_0^t r(s) \cdot e^{-r(s) \cdot x} dx = 1 - e^{-r(s) \cdot t}$$

Lemma 3.1.2 (State-to-state timed transition probability). *The probability of moving from a state s to s' within $[0, t]$ is:*

$$\mathbf{P}(s, s') \cdot \int_0^t r(s) \cdot e^{-r(s) \cdot x} dx = \mathbf{P}(s, s') \cdot (1 - e^{-r(s) \cdot t})$$

The probabilities for intervals of the form $[t, t']$ and $[t', \text{inf}]$ are computed analogously.

As in DTMCs we are mostly interested in the probability of reachability events. Therefore we first need to define timed paths over a CTMC [10].

Definition 3.1.2 (Timed paths). *Paths in a CTMC \mathcal{C} are maximal paths of alternating states and time instants denoted as $\pi = s_0, t_0, s_1, t_1, s_2 \dots$ where $s_i \in S$, $t_i \in \mathbb{R}_{>0}$ and $\mathbf{P}(s_i, s_{i+1}) > 0$ for all i .*

Further, $\text{Paths}(\mathcal{C})$ denotes the set of all paths π in \mathcal{C} .

3.2 Timed reachability probabilities

To specify certain points along a path π the following notations will be used:

- $\pi[i] := s_i$, the $(i + 1)$ -st state along the timed path π .
- $\pi\langle i \rangle := t_i$, the time spent in state s_i .
- $\pi@t$, the state occupied in π at time $t \in \mathbb{R}_{\leq 0}$, i.e. $\pi@t := \pi[i]$ where i is the smallest index such that $\sum_{j=0}^i \pi\langle j \rangle > t$.

Now that paths over a CTMC are introduced we are going to deal with probability measurements. As we did for DTMC we are going to define a probability space over the paths of the CTMC.

Definition 3.1.3 (Cylinder set). *Let $s_0, \dots, s_k \in S$ with $\mathbf{P}(s_i, s_{i+1}) > 0$ for $0 \leq i < k$ and I_0, \dots, I_{k-1} non-empty intervals in $\mathbb{R}_{>0}$ with rational bounds. The cylinder set of $s_0, I_0, \dots, I_{k-1}, s_k$ is defined by:*

$$\begin{aligned} \text{Cyl}(s_0, I_0, \dots, I_{k-1}, s_k) = \{ \pi \in \text{Paths}(\mathcal{C}) \mid \forall 0 \leq i \leq k. \pi[i] = s_i \\ \text{and } i < k \Rightarrow \pi\langle i \rangle \in I_i \} \end{aligned}$$

For the measurements of paths the cylinder sets will serve as basic events of the smallest σ -algebra on $\text{Paths}(\mathcal{C})$.

The σ -algebra associated with the CTMC \mathcal{C} is the smallest σ -algebra $\mathcal{F}(\text{Paths}(s_0))$ containing all cylinder sets $\text{Cyl}(s_0, I_0, \dots, I_{k-1}, s_k)$ where $s_0 \dots s_k$ is a path in \mathcal{C} (starting in s_0) and where I_0, \dots, I_{k-1} range over all sequences of non-empty intervals in $\mathbb{R}_{>0}$. The sample space for a given state s will be the set of all interval-timed paths $s_0 I_0 \dots I_{k-1} s_k$ with $s = s_0$. Now let Pr be the unique probability measure on the σ -algebra $\mathcal{F}(\text{Paths}(s_0))$ defined by induction on k as follows $Pr(\text{Cyl}(s_0)) = \nu_{init}(s_0)$ and for $k > 0$

$$\begin{aligned} Pr(\text{Cyl}(s_0, I_0, \dots, I_{k-1}, s_k)) = Pr(\text{Cyl}(s_0, I_0, \dots, I_{k-2}, s_{k-1})) \cdot \\ \int_{I_{k-1}} \mathbf{R}(s_{k-1}, s_k) \cdot e^{-r(s_{k-1})t} dt \end{aligned}$$

3.2 Timed reachability probabilities

With usage of paths and cylinder sets we are now able to formalize timed reachability events and compute their probabilities. Note that we are only interested in time bounded events as probabilities for events without time-bound can be computed on the underlying DTMC of a CTMC [8].

Definition 3.2.1 (timed reachability). *Eventually reach a state in $G \subseteq S$ in the interval I . Formally:*

$$\diamond^I G = \{ \pi \in \text{Paths}(\mathcal{C}) \mid \exists t \in I. \pi@t \in G \}$$

Definition 3.2.2 (Constrained timed reachability). *Reach a state in $G \subseteq S$ in the interval I without visiting a state in $F \subseteq S$ before. Formally:*

$$\overline{F}U^I G = \{\pi \in \text{Paths}(\mathcal{C}) \mid \exists t \in I. \pi @ t \in G \wedge \forall d < t. \pi @ d \notin F\}$$

Now that we are able to formally describe the events we are interested in we are going to compute their probabilities.

For computing the probability of those events we have to take all cylinder sets starting in s in account where the maximal and minimal time to reach a target state lies within I . This computation can be expressed with the following function [10].

Let function $x_s(t) = Pr(s \models \overline{F}U^{\leq t} G)$ with $x_s(t) = 0$ for all t if G is not reachable from s via \overline{F} states. Further, $x_s(t) = 1$ for all t if $s \in G$. For any other state $s \in S$ it holds

$$x_s(t) = \int_0^t \sum_{s' \in S} \underbrace{\mathbf{R}(s, s') \cdot e^{-r(s) \cdot x}}_{\text{probability to move to state } s' \text{ within } [0, x]} \cdot x_{s'}(t - x) dx$$

As solving these nested integrals is quite complex we will reduce this problem to an easier one. Therefore we first introduce transient probabilities and afterwards reduce the computation of timed reachability probabilities to the computation of transient probabilities.

3.3 Computing transient probabilities

Transient probabilities are used to describe the state of the system at a fixed point in time t .

Let $p_s(t)$ denote the probability of being in state s at time t . Further let $\underline{p}(t) = (p_{s_1}(t), \dots, p_{s_n}(t))$. The probability of being in a certain state at time t can be computed by usage of the fixed-point-notation $\underline{p}'(t) = \underline{p}(t) \cdot (\mathbf{R} - \mathbf{r})$, where \mathbf{r} is the diagonal matrix of vector \underline{r} (see [8]). Solving this differential equation yields:

$$\underline{p}(t) = \underline{p}(0) \cdot e^{(\mathbf{R} - \mathbf{r}) \cdot t}$$

which can be rewritten by using Taylor-Maclaurin expansion [10] resulting in

$$\underline{p}(t) = \underline{p}(0) \cdot e^{(\mathbf{R} - \mathbf{r}) \cdot t} = \underline{p}(0) \cdot \sum_{i=0}^{\infty} \frac{((\mathbf{R} - \mathbf{r}) \cdot t)^i}{i!}$$

The matrix $(\mathbf{R} - \mathbf{r})$ can contain both positive and negative entries and can also contain values larger than one, thus the computation $(\mathbf{R} - \mathbf{r})^i$ would result be numerically instable as the matrix entries could get arbitrarily large and create overflows in the computation. To avoid the numerical instability of the computation we will modify the system by using a uniformized CTMC and then compute the transient probabilities by exploiting the characteristics of such a uniformized CTMC [10].

3.3 Computing transient probabilities

Definition 3.3.1 (Uniformization of a CTMC). *Let $r \in \mathbb{R}_{>0}$ such that $r \geq \max_{s \in S} \{r(s)\}$. Then $\text{unif}(\mathcal{C}) = (S, \mathbf{P}^{\text{unif}(\mathcal{C})}, \bar{r}, \nu_{\text{init}}, AP, L)$ with $\bar{r}(s) = r$ for all $s \in S$, and:*

$$\begin{aligned} \mathbf{P}^{\text{unif}(\mathcal{C})}(s, s') &= \frac{r(s)}{r} \cdot \mathbf{P}(s, s') \text{ for all } s' \neq s \text{ and} \\ \mathbf{P}^{\text{unif}(\mathcal{C})}(s, s) &= \frac{r(s)}{r} \cdot \mathbf{P}(s, s) + 1 - \frac{r(s)}{r} \end{aligned}$$

The uniformization basically normalizes all exit-rates to the maximal exit rate. As most of the exit-rate of most states will be increased self loops need to be added to those states. Note that the rate matrix is not affected by the uniformization. Therefore it holds $\mathbf{R} = \mathbf{R}^{\text{unif}(\mathcal{C})} = \mathbf{P}^{\text{unif}(\mathcal{C})} \cdot r$.

We want to use this construction of a uniformized CTMC for the computation of the transient probabilities and therefore need to guarantee that our original CTMC \mathcal{C} and the uniformized CTMC $\text{unif}(\mathcal{C})$ are related in a way that preserves transient probabilities. Therefore we need them to be weakly bisimilar to each other. It can be shown that the following holds [15].

Definition 3.3.2 (Weak probabilistic bisimulation). *Let $\mathcal{C} = (S, \mathbf{P}, r, \nu_{\text{init}}, AP, L)$ be a CTMC and $R \subseteq S \times S$ an equivalence. Then: R is a probabilistically weak bisimulation on S if for any $(s, t) \in R$*

1. $L(s) = L(t)$, and
2. $\mathbf{R}(s, C) = \mathbf{R}(t, C)$ for all $C \in S/R$ with $C \neq [s]_R = [t]_R$.

Where $[s]_R$ represents the group of states bisimilar to s under R .

Definition 3.3.3 (Weak probabilistic bisimilarity). *Let \mathcal{C} be a CTMC and $s, t \in S$ states in \mathcal{C} . Then: s is probabilistically weak bisimilar to t , denoted $s \approx_m t$, if there exists a probabilistically weak bisimulation R with $(s, t) \in R$.*

Lemma 3.3.1 (Preservation of transient probabilities). *For all CTMCs \mathcal{C} with states $s, u \in S$ and $t \in \mathbb{R}_{\geq 0}$ it holds:*

$$s \approx_m u \text{ implies } p_s(t) = p_u(t)$$

In other words, if s and u are probabilistically weak bisimilar their transient probabilities coincide.

As mentioned before the rate matrix \mathbf{R} is not affected by the uniformization. Thus a CTMC \mathcal{C} is probabilistically weak bisimilar to its uniformized CTMC $\text{unif}(\mathcal{C})$

Now the question is, how can we use the uniformization to compute

$\underline{p}(t) = \underline{p}(0) \cdot \sum_{i=0}^{\infty} \frac{((\mathbf{R}-\mathbf{r}) \cdot t)^i}{i!}$ without having to deal with numeric instability and arbitrary large values?

According to the lemma 3.1, uniformization preserves the transient probabilities of the

3 Continuous-Time Markov Chain Model Checking

original CTMC, and therefore we are allowed to replace \mathbf{R} and \mathbf{r} by its variants $\bar{\mathbf{R}} = \mathbf{R}^{unif(C)}$ and $\bar{\mathbf{r}} = r \cdot \mathbf{I}$ from the uniformized CTMC.

With $\bar{\mathbf{P}} = \mathbf{P}^{unif(C)}$ and $\bar{\mathbf{R}}(s, s') = \bar{\mathbf{P}}(s, s') \cdot \bar{\mathbf{r}}(s) = \bar{\mathbf{P}}(s, s') \cdot r$ it holds

$$\begin{aligned} \underline{p}(t) &= \underline{p}(0) \cdot e^{\mathbf{R} - \mathbf{r}} = \underline{p}(0) \cdot e^{(\bar{\mathbf{R}} - \bar{\mathbf{r}}) \cdot t} \\ &= \underline{p}(0) \cdot e^{(\bar{\mathbf{P}} \cdot r - \mathbf{I} \cdot r) \cdot t} = \underline{p}(0) \cdot e^{(\bar{\mathbf{P}} - \mathbf{I}) \cdot t \cdot r} \\ &= \underline{p}(0) \cdot e^{-r \cdot t} \cdot e^{\bar{\mathbf{P}} \cdot r \cdot t} \end{aligned}$$

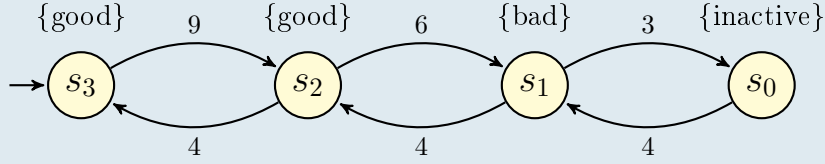
If we now apply the Taylor-Maclaurin expansion this results in

$$\begin{aligned} \underline{p}(t) &= \underline{p}(0) \cdot e^{-r \cdot t} \cdot e^{\bar{\mathbf{P}} \cdot r \cdot t} = \underline{p}(0) \cdot e^{-r \cdot t} \cdot \sum_{i=0}^{\infty} \frac{(r \cdot t)^i}{i!} \cdot \bar{\mathbf{P}}^i \\ &= \underline{p}(0) \cdot \sum_{i=0}^{\infty} e^{-r \cdot t} \frac{(r \cdot t)^i}{i!} \cdot \bar{\mathbf{P}}^i \end{aligned}$$

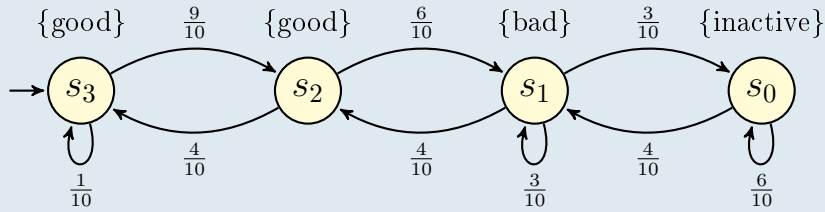
As $\bar{\mathbf{P}}$ is a stochastic matrix the matrix exponential $\bar{\mathbf{P}}^i$ is numerically stable.

3.3 Computing transient probabilities

Example:



Take a look at the CTMC \mathcal{D} above and suppose we now want to compute the probability to be in states s_i at time $t = 5$ with accuracy-bound of $\varepsilon = 10^{-6}$ e.g. compute $\underline{p}(5)$. Therefore we first uniformize \mathcal{C} with rate $r=10$. The result of $unif(\mathcal{C}, r)$ is shown below. Note that transitions are labelled with probabilities instead of rates here.



Now we can just plug in all given values to compute $\underline{p}(5)$

$$\begin{aligned} \underline{p}(5) &= \underline{p}(0) \cdot \sum_{i=0}^{\infty} \cdot e^{-10 \cdot 5} \frac{(10 \cdot 5)^i}{i!} \cdot \mathbf{P}^i \\ &= (1, 0, 0, 0) \cdot \sum_{i=0}^{\infty} \cdot e^{-50} \frac{(50)^i}{i!} \cdot \begin{pmatrix} 0.1 & 0.9 & 0 & 0 \\ 0.4 & 0 & 0.6 & 0 \\ 0 & 0.4 & 0.3 & 0.3 \\ 0 & 0 & 0.4 & 0.6 \end{pmatrix}^i \end{aligned}$$

Now we can naively compute a truncation point R with accuracy $\sum_{i=0}^x e^{-50} \cdot \frac{50^i}{i!} \geq 1 - \varepsilon = 10^{-6}$. This gives us the truncation point $R = 87$ Which reduces the computation to

$$\begin{aligned} \underline{p}(5) &= (1, 0, 0, 0) \cdot \sum_{i=0}^{87} \cdot e^{-50} \frac{(50)^i}{i!} \cdot \begin{pmatrix} 0.1 & 0.9 & 0 & 0 \\ 0.4 & 0 & 0.6 & 0 \\ 0 & 0.4 & 0.3 & 0.3 \\ 0 & 0 & 0.4 & 0.6 \end{pmatrix}^i \\ &= (0.109215, 0.245734, 0.368600, 0.27645) \end{aligned}$$

3.4 Efficient computation of reachability probabilities

As stated before the computation of reachability probabilities can be reduced to the computation of transient probabilities. To do so we change the system in a way that timed reachability probabilities coincide with transient probabilities, for which we already know how to compute them [8].

Looking at the computation of $Pr(s \models \diamond^{\leq t} G)$ in a CTMC \mathcal{C} we can observe that once a path π reaches G within time t , the remaining behaviour along π is not important any more. Therefore, making all states in G absorbing does not affect the probability of the time-bounded reachability event. But since all states in G are now absorbing the probability to reach G within time t is the same as being in G at time t . Remember that $\mathcal{C}[G]$ denotes the CTMC obtained by making all states of G absorbing.

$$\underline{Pr}^{\mathcal{C}}(s \models \diamond^{\leq t} G) = \underline{Pr}^{\mathcal{C}[G]}(s \models \diamond^{=t} G) = \sum_{i=0}^{\infty} e^{-r \cdot t} \frac{(r \cdot t)^i}{i!} \cdot \mathbf{p}^{unif(\mathcal{C}[G])} \cdot \Phi_G$$

For computing the probability of constrained reachability events $\overline{F} \mathbf{U}^{\leq t} G$ we just need a further modification of states namely making all states in $G \cup F$ absorbing. This way we assure that we only reach states in G via valid paths and the computation can again be reduced to the computation of transient probabilities.

$$\underline{Pr}^{\mathcal{C}}(s \models \overline{F} \mathbf{U}^{\leq t} G) = \underline{Pr}^{\mathcal{C}[G \cup F]}(s \models \diamond^{=t} G)$$

We are now able to compute time-bounded reachability probabilities for $I = [t, t]$ and $I = [0, t]$. So there are still two kinds of intervals to consider. The first one is $I = [t, t]$ and the second one is $I = [t, \infty)$.

For the computation of $Pr^{\mathcal{C}}(s \models \overline{F} \mathbf{U}^{[t, t']} G)$ we split the computation into two parts. The first part is the probability of remaining in \overline{F} states until time t . The second part is the probability of reaching a G state, while remaining in \overline{F} states, within the time interval $[0, t' - t]$. Combining those two computations yields:

$$\underline{Pr}^{\mathcal{C}}(s \models \overline{F} \mathbf{U}^{[t, t']} G) = \underbrace{\sum_{i=0}^{\infty} e^{-r \cdot t} \frac{(r \cdot t)^i}{i!} \cdot \mathbf{P}^{unif(\mathcal{C}[F])}}_{\text{prob. of staying within } \overline{F} \text{ until } t} \cdot \underbrace{\underline{Pr}^{\mathcal{C}[G \cup F]}(s \models \diamond^{\leq (t'-t)} G)}_{\text{prob. of reaching } G \text{ within } (t'-t)}$$

Now only $[t, \infty)$ is left to consider. As for $[t, t']$ the computation can be split into two parts exactly as before, but since the second part now yields $Pr(s \models \overline{F} \mathbf{U}^{[0, \infty)} G)$, we can use the underlying DTMC $emb(\mathcal{C}) = (S, \mathbf{P}, \nu_{init}, AP, L)$ to compute this probability. Computing the latter probability by usage of the embedded DTMC yields:

$$\underline{Pr}^{\mathcal{C}}(s \models \overline{F} \mathbf{U}^{[t, \infty)} G) = \sum_{i=0}^{\infty} e^{-r \cdot t} \frac{(r \cdot t)^i}{i!} \cdot \mathbf{P}^{unif(\mathcal{C}[F])} \cdot \underline{Pr}^{emb(\mathcal{C}[G \cup F])}(s \models \diamond G)$$

3.5 Continuous Stochastic Logic

We are now able to compute probabilities of reachability events on a CTMC. Now we need a logic to formalize the events we want to compute. Therefore we use the Continuous Stochastic Logic (CSL) which is a branching-time temporal logic based on CTL [1].

Definition 3.5.1 (Continuous Stochastic Logic).

Property specifications for a CTMC can be formalized by usage of state-formulas and path-formulas. They are defined as follows:

- CSL state formulas over the set AP are formed according to the grammar:

$$\Phi ::= true \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid P_J(\varphi)$$

where $a \in AP$, φ is a path formula and $J \subseteq [0, 1]$, $J \neq \emptyset$ is a non-empty interval.

- CSL path formulas are formed according to the following grammar:

$$\varphi ::= \bigcirc\Phi \mid \bigcirc^I\Phi \mid \Phi_1 \mathbf{U}^I \Phi_2 \mid \Phi_1 \mathbf{U} \Phi_2$$

where Φ_1 and Φ_2 are state formulas and I is an non-empty interval of $\mathbb{R}_{\geq 0}$.

Except for the time-bounded operators the semantics are the same as for PCTL for DTMCs. The satisfaction relation \models is defined for state formulas as follows.

$$\begin{aligned} s \models true & \quad \text{always} \\ s \models a & \quad \Leftrightarrow a \in L(s) \\ s \models \Phi_1 \wedge \Phi_2 & \quad \Leftrightarrow s \models \Phi_1 \text{ and } s \models \Phi_2 \\ s \models \neg\Phi & \quad \Leftrightarrow s \models \Phi \text{ does not hold} \\ s \models P_J(\varphi) & \quad \Leftrightarrow Pr(s \models \varphi) \in J \end{aligned}$$

where $Pr(s \models \varphi) = Pr_s\{\pi \in Paths(s) \mid \pi \models \varphi\}$.

Let $\pi = s_0 t_0 s_1 t_1 \dots$ be an infinite path in CTMC \mathcal{C} . The satisfaction relation \models for path formulae is defined as follows.

$$\begin{aligned} \pi \models \bigcirc^I\Phi & \quad \Leftrightarrow s_1 \models \Phi \wedge t_0 \in I \\ \pi \models \Phi \mathbf{U}^I \Psi & \quad \Leftrightarrow \exists t \in I. ((\forall t' \in [0, t). \pi @ t' \models \Phi) \wedge \pi @ t \models \Psi) \end{aligned}$$

3.6 CTMC vs CSL Model Checking

Now we have everything we need to perform model checking on CTMCs. For a given CTMC \mathcal{C} , a state $s \in S$ and a CSL formula Φ , we now want to check wheter $s \models \Phi$. Like for model checking on DTMCs this is done by computing the satisfaction sets of the formula. The satisfaction set of Φ (denoted $Sat(\Phi)$) is computed by a bottom-up traversal of Φ 's parse tree where nodes of the parse tree represent the subformulas of

3 Continuous-Time Markov Chain Model Checking

Φ . After obtaining the satisfaction set we need to check if $s \in \text{Sat}(\Phi)$. If s is in the satisfaction set of a formula it satisfies this formula.

$\text{Sat}(\cdot)$ is defined by structural induction as follows:

$$\begin{aligned}
 \text{Sat}(\text{true}) &= S \\
 \text{Sat}(a) &= \{s \in S \mid a \in L(s)\} \text{ for any } a \in AP \\
 \text{Sat}(\Phi \wedge \Psi) &= \text{Sat}(\Phi) \cap \text{Sat}(\Psi) \\
 \text{Sat}(\neg\Psi) &= S \setminus \text{Sat}(\Psi) \\
 \text{Sat}(P_J(\varphi)) &= \{s \in S \mid \text{Pr}(s \models \varphi) \in J\}
 \end{aligned}$$

To compute $\text{Sat}(P_J(\varphi))$ we need to compute the probability of the path-formulae φ .

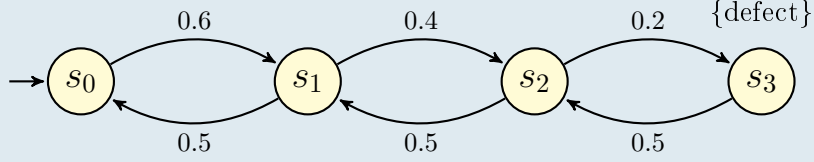
$\text{Pr}(s \models \bigcirc\Phi)$ and $\text{Pr}(s \models \Phi\mathbf{U}\Psi)$ are computed on the embedded DTMC.

Further, for the time-bounded constrained reachability operator we already have seen how to compute the probabilities. And as $\text{Pr}(s \models \bigcirc^I\Phi)$ is the probability of going from state s directly to a state $s' \in \text{Sat}(\Phi)$ in I we can apply definition 2.2.3. This yields

$$\text{Pr}(s \models \bigcirc^I\Phi) = \sum_{s' \in \text{Sat}(\Phi)} \mathbf{P}(s, s') \cdot (e^{-r(s) \cdot \text{inf}(I)} - e^{-r(s) \cdot \text{sup}(I)})$$

Example:

For this example look at the CTMC from 3.1. again.



Let us check the formula $\Phi = \neg defect \wedge P_{\geq 0.75}(true \mathbf{U}^{[0,29]} defect)$. Now for the computation of $s \models \Phi$ for all $s \in S$ like for PCTL model checking we do this by computation of the satisfaction sets. Therefore:

$$\begin{aligned} Sat(true) &= S \\ Sat(defect) &= \{s_3\} \\ Sat(\neg defect) &= \{s_0, s_1, s_2\} \end{aligned}$$

Now we need to compute

$$\underline{Pr}^{\mathcal{C}}(true \mathbf{U}^{[0,29]} defect) = \sum_{i=0}^{\infty} e^{r \cdot 29} \frac{(r \cdot 29)^i}{i!} \cdot \mathbf{P}^{unif(\mathcal{C})} \cdot \underline{Sat}(defect)$$

By applying uniformization to \mathcal{C} we get

$$\mathbf{P}^{(unif\mathcal{C})} = \frac{1}{9} \cdot \begin{pmatrix} 3 & 6 & 0 & 0 \\ 5 & 0 & 4 & 0 \\ 0 & 5 & 2 & 2 \\ 0 & 0 & 5 & 4 \end{pmatrix}, r = 0.9$$

If we now compute the probability by using those values we get

$$\underline{Pr}^{\mathcal{C}}(true \mathbf{U}^{[0,29]} defect) = (0.743028, 0.763895, 0.818737, 1)^T$$

By usage of this result we can compute the remaining satisfaction sets.

$$\begin{aligned} Sat(P_{\geq 0.75}(true \mathbf{U}^{[0,29]} defect)) &= \{s_1, s_2, s_3\} \\ Sat(\Phi) &= \{s_1, s_2\} \end{aligned}$$

Therefore $s_1 \models \Phi$ and $s_2 \models \Phi$ and s_0, s_3 are not satisfying Φ .

4 Efficient computation of poisson probabilities

While computing the transient probabilities for a Continuous-time Markov Chain we get to the point where we need to compute an infinite sum of the form $\sum_{i=0}^{\infty} e^{-\lambda} \cdot \frac{\lambda^i}{i!}$ called poisson probability. This sum could naively be computed by increasing the upper bound k up to point, such that $\sum_{i=0}^k e^{-\lambda} \cdot \frac{\lambda^i}{i!} \leq 1 - \varepsilon$, where ε is the wanted accuracy.

This naive approach of computing an infinite sum is quite expensive because for large λ a lot of negligible small values are added up at the beginning. To make this more clearly take a look at Figure 4.1 where it can be seen that the mass of the values from 0 to 30 are negligible small. So we need to find a efficient way of truncating the sum to an interval of relevant values. In 1988 Bennett L. Fox and Peter W. Glynn published an algorithm which gives such a truncation [6].

This section will give an intuition how the algorithm by Fox and Glynn works and give a graphical comparison to a naive implementation with respect to computation-time.

The proofs for the two used corollaries are very technical and not necessary for understanding the algorithm and are therefore included at the end of this thesis.

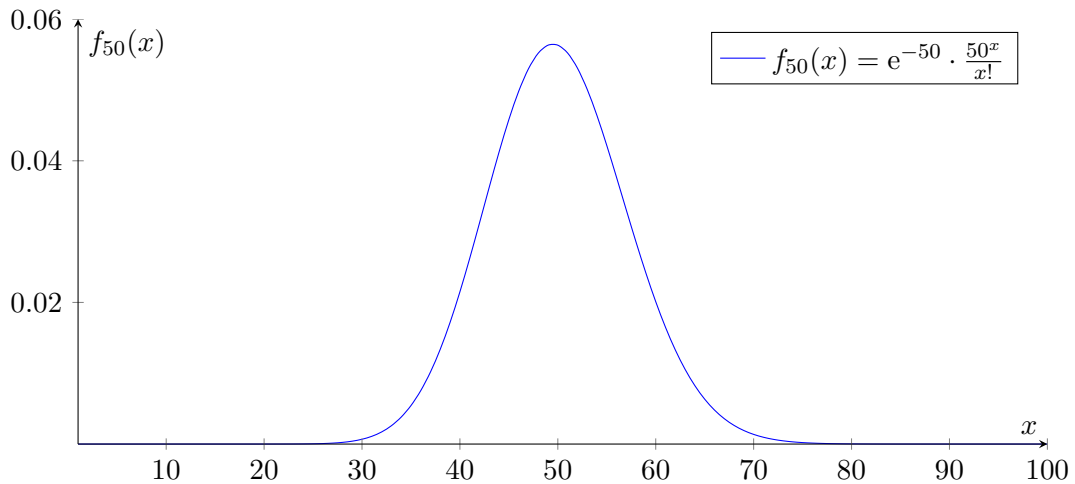


Figure 4.1: Poisson distribution for $\lambda = 50$

4.1 Intuition behind the Fox Glynn algorithm

Basically the algorithm developed by Fox and Glynn computes a left and a right truncation point L, R in a way that we can truncate the summation to $\sum_{i=L}^R e^{-\lambda} \cdot \frac{\lambda^i}{i!}$. The algorithm also guarantees that the mass outside of the left and right truncation point is lower than a given accuracy bound ε . By using this computation a great amount of computation-steps can be saved since every step before the left truncation point can be dropped. This amount of dropped calculations can be quite large for greater λ , this gets clear by looking at the graph in Figure 4.1.

4.2 Finding the truncation points

For finding a left and right truncation point we need to be able to compute the mass outside of the truncation points to guarantee that the mass outside is at most the given accuracy bound.

As the probability that exactly i events happen when λ are expected can be computed by $p_\lambda(i) = \frac{e^{-\lambda} \cdot \lambda^i}{i!}$ [7]. It follows that the probability of at most i events can be expressed as $T_\lambda(i) = \sum_{j=0}^{\lfloor i \rfloor} p_\lambda(j)$ and the probability of at least i events is analogously $Q_\lambda(i) = \sum_{j=\lceil i \rceil}^{\infty} p_\lambda(j)$.

With $T_\lambda(L)$ and $Q_\lambda(R)$ we can compute the mass of the left and right tail if we are given a left and a right truncation point.

To find those truncation points efficiently Fox and Glynn gave two corollaries in their paper [6] that basically arise by approximations on $T_\lambda(i)$ and $Q_\lambda(i)$.

Corollary 1. If $\lambda \geq 2$ and $\frac{1}{2\sqrt{2\lambda}} \leq k \leq \frac{\sqrt{\lambda}}{2\sqrt{2}}$

$$Q_\lambda \left(\lceil m + k\sqrt{2\lambda} + 3/2 \rceil \right) \leq \frac{a_\lambda \cdot d(k, \lambda) \cdot \exp\left(\frac{-k^2}{2}\right)}{k\sqrt{2\pi}}$$

With $a_\lambda = \left(1 + \frac{1}{\lambda}\right) \cdot \sqrt{2} \cdot \exp\left(\frac{1}{16}\right)$ and $d(k, \lambda) = 1 / \left(1 - \exp\left(-\frac{2}{9} \cdot \left(k \cdot \sqrt{2\lambda} + \frac{3}{2}\right)\right)\right)$. Which gives an approximation to the mass of the right tail.

Corollary 2. If $\lambda \geq 2$ and $k \geq \frac{1}{\sqrt{2\lambda}}$

$$T_\lambda \left(\lfloor m - k\sqrt{\lambda} - 3/2 \rfloor \right) \leq \frac{b_\lambda \cdot \exp\left(\frac{-k^2}{2}\right)}{k\sqrt{2\pi}}$$

Where $b_\lambda = \left(1 + \frac{1}{\lambda}\right) \cdot \exp\left(\frac{1}{8\lambda}\right)$.

Which gives an approximation to the mass of the left tail.

4 Efficient computation of poisson probabilities

Now we can use these corollaries to compute the truncation points where we want the mass of the left tail and the mass of the right tail each to be at most $\frac{\varepsilon}{2}$. To do so, we can simply apply the corollaries to get upper bounds on the mass of the tails.

The **right truncation point** R is computed by taking the lower bound on k from **Corollary 1** and then repeatedly increasing k by one until the right hand side of the equation in **Corollary 1** is smaller than $\frac{\varepsilon}{2}$. Now that k is known $Q_\lambda(\lceil m + k\sqrt{2\lambda} + 3/2 \rceil) \leq \frac{a_\lambda d(k, \lambda) e^{-k^2/2}}{k\sqrt{2\pi}} \leq \frac{\varepsilon}{2}$ yields the right truncation point $R = \lceil m + k\sqrt{2\lambda} + 3/2 \rceil$.

The **left truncation point** L is computed by taking the lower bound on k from **Corollary 2** and repeatedly increasing k until the right hand side of the equation in **Corollary 2** is also smaller than $\frac{\varepsilon}{2}$. Plugging k into $T_\lambda(\lfloor m - k\sqrt{\lambda} - 3/2 \rfloor) \leq \frac{a_\lambda d(k, \lambda) e^{-k^2/2}}{k\sqrt{2\pi}} \leq \frac{\varepsilon}{2}$ we get our left truncation point $L = \lfloor m - k\sqrt{\lambda} - 3/2 \rfloor$.

It suffices to increase k by one in every iteration, as $e^{k^2/2}$ is the dominating part of both approximations and thus the approximations converge very fast. Even for large λ and a reasonable accuracy-bound the parameter k is found that fast that no further optimizations are needed here.

Example:

Now as an example, we will compute the truncation points for $\lambda = 500$ and $\varepsilon = 1 \cdot 10^{-6}$.

First we compute the **left truncation point**. From the preliminaries of **Corollary 2** we get $\frac{1}{2} \cdot k \geq \frac{1}{\sqrt{2 \cdot 500}}$ so let $k = 1$. $\frac{b_\lambda \cdot \exp\left(\frac{-1^2}{2}\right)}{1\sqrt{2\pi}} = 0,24$ so we need to increase k by 1. Doing this repeatedly we find that for $k = 5$ it holds $T_\lambda(\lfloor 500 - 5\sqrt{500} - 3/2 \rfloor) \leq \varepsilon/2$, this gives us our **left truncation point** $L = 385$.

Now for the **right truncation point** we get $k = 1$ by applying the restrictions for k from **Corollary 1** and again need to increase k up to $k = 5$ until the right hand side of the equation is smaller than $\varepsilon/2$. This yields $Q_\lambda(\lceil 500 + 5\sqrt{2 \cdot 500} + 3/2 \rceil) \leq \varepsilon/2$, which gives us our **right truncation point** $R = 660$.

Now the computation of $\sum_{i=0}^{\infty} e^{-500} \cdot \frac{500^i}{i!} \geq 1 - \varepsilon$ can be reduced to $\sum_{i=385}^{660} e^{-500} \cdot \frac{500^i}{i!}$.

Note that finding the left and right truncation point only needed 10 iterations in total.

4.3 Improvement of CTMC Model Checking by usage of Fox & Glynn's algorithm

As the algorithm by Fox Glynn efficiently computes poisson probabilities we can use it to improve the computation of transient probabilities in CTMC Model Checking. This results in two independent ways of decreasing the runtime.

One improvement is that the time needed to compute the truncation of the sum is reduced significantly compared to a naive approach, as seen in Figure 4.2.

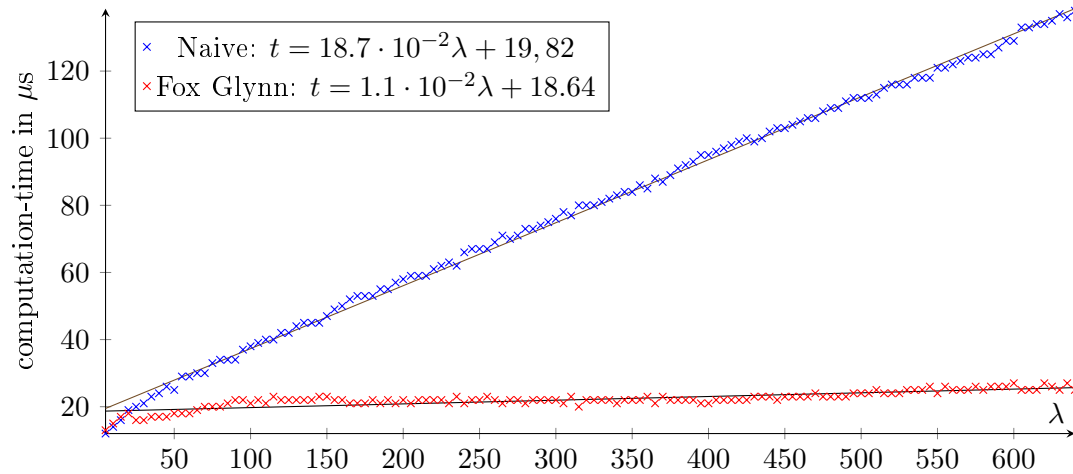


Figure 4.2: Comparison of computation time of Fox Glynn vs. a naive implementation

Further, the right truncation point should be nearly the same for a naive truncation and the truncation from the algorithm. Thus only the left truncation point is improving the computation. As every computation before the left truncation point can be omitted, using the algorithm by Fox and Glynn saves $\mathcal{O}(L \cdot N)$ multiplications, where L is the left truncation point and N is the number of states in the system.

In conclusion using the algorithm of Fox and Glynn to optimize the computation of transient probabilities improves the computation-time drastically for reachability probabilities with a large time-bound and systems with a high maximal exit-rate as $\lambda = r \cdot t$. Also for huge systems Fox and Glynn's algorithm yields a good improvement.

5 Implementation

This section will explain what improvements were made in my implementation, in addition to the usage of Fox and Glynn's algorithm, to increase the performance of the CTMC Model Checker

All path formulas without time-bound can be reduced to DTMC model checking. Thus the main part of the implementation was to make the computation of time-bounded reachability probabilities more efficiently. Basically the improvements reduce the transition matrix used in the computation and lower the number of necessary multiplications.

To improve the computation of reachability events we do not compute the transient probabilities with the whole transition matrix. Instead we use a reduced transition matrix to avoid computing probabilities for states where the probabilities can be obtained by graph analysis. The reduction consists of two parts.

First the set of states that are not able to reach a target-state via valid paths is determined by a reachability analysis of the graph. As the probability for those states is zero they are excluded from further computations. Note that the exit-rates of the remaining states must not be changed to guarantee the correctness of the state-to-state probabilities, thus the system resulting from this modification is not a CTMC any more.

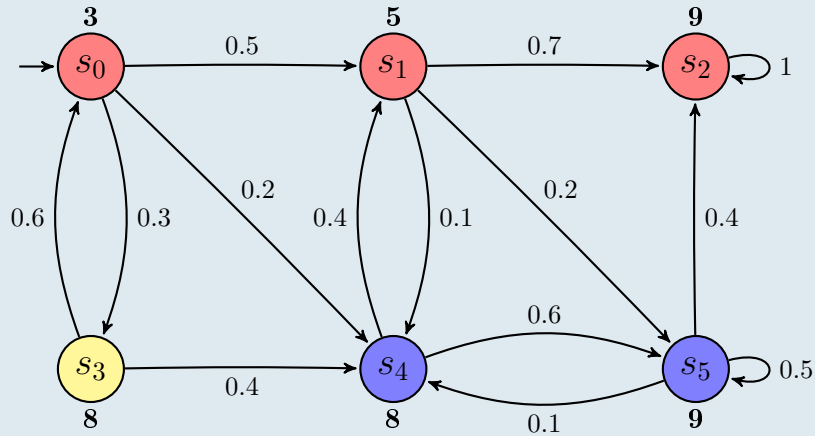
Afterwards all target-states are made absorbing in order to reduce the computation of reachability probabilities to the computation of transient probabilities. Now by definition 3.3.2 all target-states are weak bisimilar to each other and thus can be reduced to a single state.

Apart from the reduction of the used transition matrix the number of matrix multiplications performed can be reduced to improve the runtime. As seen before the truncation of the poisson probability directly affects the number of necessary multiplications. Thus lower truncation points yield a faster computation. The truncation points obtained by usage of the algorithm by Fox and Glynn are directly affected by the distribution parameter $\lambda = r \cdot t$. Therefore a reduction of the uniformization rate r results in a reduced number of necessary multiplications. To reduce the uniformization rate we only compute the uniformization for states in the reduced system obtained in the previous step and ignore the exit-rate of the target state. By doing so the exit-rates of all target states and states with probability zero are not included in the computation of the uniformization rate, which may yield a lower rate.

Example:

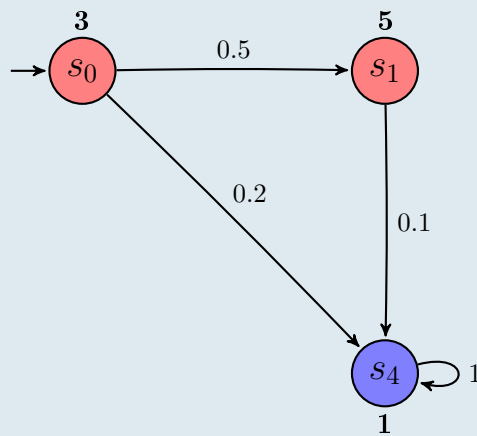
This example will stepwise apply the improvements.

Assume we want to compute $Pr(\mathbf{F} \mathbf{U}^{[0,\infty)} \mathbf{G})$ on the following system where the exit-rates are written next to the corresponding states.



In the first step we do a graph analysis and discover that s_2 and s_3 are not able to fulfil the formula at all. Thus they are excluded from further computations.

Now the target states s_4 and s_5 are made absorbing and are now weakly bisimilar. Therefore we reduce them to only a single state. The resulting system is shown below.



As seen above number of transitions is reduced from 14 to 4 and the maximal exit-rate is reduced from 9 to 5.

6 Benchmark against MRMC and Prism

The previous chapter dealt with improvements of CTMC model checking. This chapter will compare the implementation of these improvements for StoRM with MRMC [9] and PRISM [11] (where the sparse engine of PRISM was used) with respect to the computation time and storage consumption.

The first interesting point is the usage of memory during the execution of the model checkers.

Figure 5.1 shows a comparison of RAM-usage which looks similar for all formulae and models.

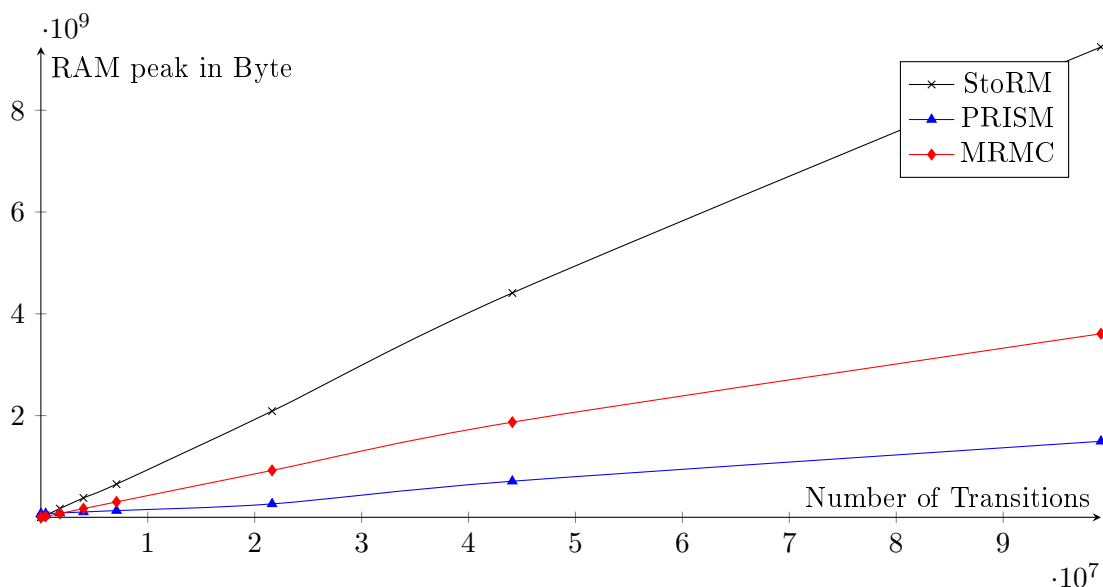


Figure 6.1: Maximal RAM-usage while checking $\diamond^{<100} \text{minimum}$ against the 'Workstation cluster' model [11]

As seen in Figure 6.1 StoRM needs around 6 times more space than PRISM. The main reason for this is a copy of the transition matrix used to transform the rate matrix into a probability matrix. This copy doubles the memory usage. Further StoRM uses a slightly different representation of sparse matrices which needs more storage as the representations of sparse matrices in MRMC and PRISM.

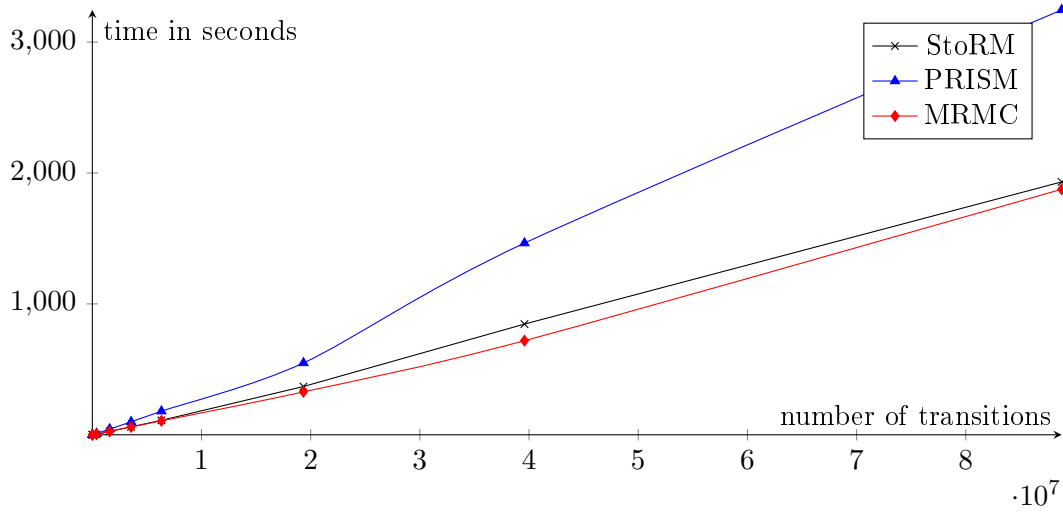


Figure 6.2: Time needed for checking $\diamond^{<100} \text{minimum}$ against the 'Workstation cluster' model [11]

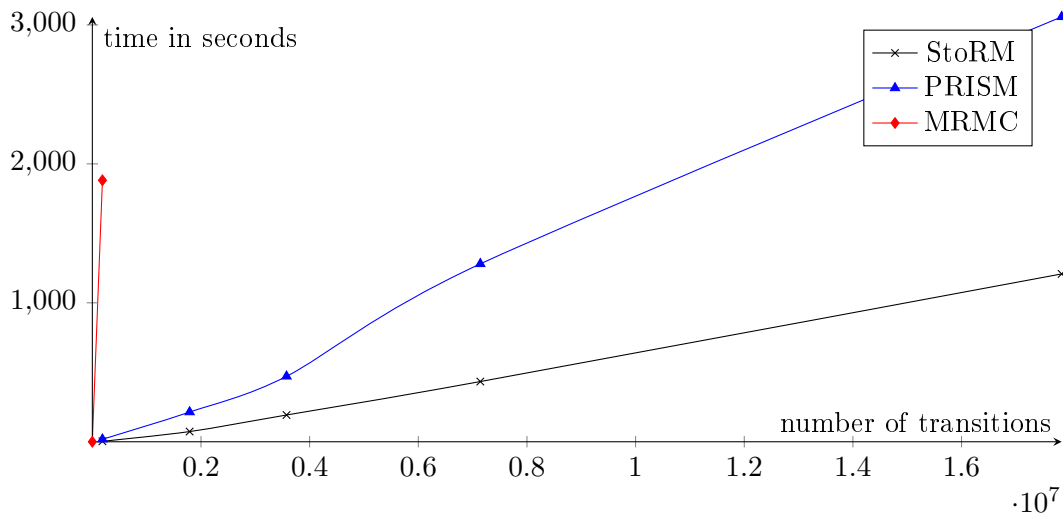


Figure 6.3: Time needed for checking $\diamond^{432000} \text{down}$ against the 'Embedded control system' model [11]

6 Benchmark against MRMC and Prism

The other interesting characteristic for benchmarking is the time needed to check certain properties.

For time-bounded properties it depends very much on the model and the formula which Model Checker is faster. As seen in Figure 6.2 StoRM gets better if matrix-vector multiplications on large matrices need to be performed. Also as seen in Figure 6.3 StoRM is a lot faster than PRISM when it comes to many matrix-vector multiplications (around 36.000 for the shown property).

The main reason for for the lower computation-times is that the matrix used for multiplications is smaller than in PRISM and MRMC due to the reduction of the system as explained before. Further, for the tested models StoRM performed around 10% less matrix multiplications than PRISM, which is caused by the reduction of the maximal exit-rate.

For some reason MRMC failed to compute the maximal exit-rate for the 'Embedded control system'. Independent of the size of the model MRMC computed maximal exit-rates of around 90, where as the real maximal exit-rate only is around 0.085 causing MRMC to perform around 1000 times as many matrix multiplications as StoRM and PRISM.

In conclusion StoRM needs much more memory than MRMC and PRISM and With respect to the runtime StoRM is faster than MRMC and PRISM if it comes to many multiplications but due to the memory overhead StoRM is outrun by both of them if only a few multiplications are needed.

7 Conclusion

In this bachelor thesis we gave an explanation of how to formalize reachability properties over a Discrete-time Markov Chain and how to compute the probabilities of those properties by using cylinder sets.

Afterwards, we have shown how to formalize time-bounded reachability properties over a Continuous-time Markov Chain and explained why the explicit computation of those properties is quite complex. Further, we gave a way to reduce the computation of time-bounded reachability probabilities to the computation of transient probabilities and how to compute those in a numerically stable way.

Additionally, we explained how to compute poisson probabilities more efficiently by truncating the sum by applying of the corollaries introduced by Fox and Glynn [6]. Along, we also introduced a way of optimizing the computation of reachability probabilities in a CTMC by reducing the number of multiplications needed and reducing the transition matrix.

As seen in the benchmarks the model checker with the given improvements was faster than MRMC and PRISM for large systems or a high number of matrix multiplications but due to the superior memory overhead StoRM is slower than the other two model checkers if it comes to small systems or a low number of necessary multiplications.

8 Proofs

This section will give the proofs to the corollaries from 4.1. This is just a more detailed version of the proofs by David N. Jansen [7]

First a little conclusion of the used notations.

- $p_\lambda(i) = \frac{e^{-\lambda} \cdot \lambda^i}{i!}$: The probability that exactly i events happen when λ are expected
- $Q_\lambda(i) = \sum_{j=\lceil i \rceil}^{\infty} p_\lambda(j)$: The probability that at least $\lceil i \rceil$ events happen when λ are expected
- $T_\lambda(i) = \sum_{j=0}^{\lfloor i \rfloor} p_\lambda(j)$: The probability that at most $\lfloor i \rfloor$ events happen when λ are expected
- $\varphi(x) = \frac{\exp(x^2/2)}{\sqrt{2\pi}}$: The probability density of the normal distribution
- $\bar{\Phi}(x) = \int_x^{\infty} \varphi(t) dt$: The complementary cumulative distribution of the normal distribution
- $a_\lambda = (1 + \frac{1}{\lambda}) \cdot \sqrt{2} \cdot \exp(\frac{1}{16})$: A helper variable
- $b_\lambda = (1 + \frac{1}{\lambda}) \cdot \exp(\frac{1}{8\lambda})$: A helper variable
- $d(k, \lambda) = 1 / \left(1 - \exp\left(-\frac{2}{9} \cdot \left(k \cdot \sqrt{2\lambda} + \frac{3}{2}\right)\right) \right)$: A helper Function

Basically we want to proof the following two corollaries [6] .

Corollary 1. If $\lambda \geq 2$ and $\frac{1}{2\sqrt{2\lambda}} \leq k \leq \frac{\sqrt{\lambda}}{2\sqrt{2}}$

$$Q_\lambda(\lceil m + k\sqrt{2\lambda} + 3/2 \rceil) \leq \frac{a_\lambda d(k, \lambda) e^{-k^2/2}}{k\sqrt{2\pi}}$$

Corollary 2. If $\lambda \geq 2$ and $k \geq \frac{1}{\sqrt{2\lambda}}$

$$T_\lambda(\lfloor m - k\sqrt{\lambda} - 3/2 \rfloor) \leq \frac{a_\lambda d(k, \lambda) e^{-k^2/2}}{k\sqrt{2\pi}}$$

To do so we need the following propositions.

Proposition 1. Assume $\lambda > 0$. If $n > \lambda - 1$ and $l \geq 1$, then

$$Q_\lambda(n) \leq (1 - ((\frac{\lambda}{n+1})^l)^{-1} \cdot \sum_{k=\lceil n \rceil}^{\lceil n \rceil + l - 1} p_\lambda(k)$$

Proposition 2. Let $\lambda, i \geq 1$ and $m = \lfloor \lambda \rfloor$. Then,

1.

$$p_\lambda(m - i) \leq \frac{1}{\sqrt{2\pi m}} e^{-\frac{i(i-1)}{2\lambda}},$$

2.

$$p_\lambda(m + i) \leq \frac{1}{\sqrt{2\pi m}} e^{-\frac{i(i-1)}{2\lambda} + \frac{i(i-1)(2i-1)}{12\lambda^2}},$$

Proposition 3 Suppose $\lambda \leq 2$. if $2 \leq i \leq \frac{(\lambda+3)}{2}$, then,

$$Q_\lambda(n) \leq e^{\frac{1}{8\lambda}} \left(1 + \frac{1Aswehave}{\lambda}\right) \sqrt{2} \left(1 - \left(\frac{\lambda}{m+i+1}\right)^m\right)^{-1} \cdot \bar{\Phi}\left(\frac{i - \frac{3}{2}}{\sqrt{2\lambda}}\right)$$

Proposition 4. Suppose $\lambda \leq 2$. if $2 \geq i \geq \frac{(\lambda+3)}{2}$, then,

$$Q_\lambda(n) \leq \frac{a_\lambda}{1 - e^{-\frac{2}{9}i}} \bar{\Phi}\left(\frac{i - \frac{3}{2}}{\sqrt{2\lambda}}\right)$$

Proposition 5. Suppose $\lambda \geq 2$ and $i \geq 2$. Then

$$T_\lambda(m - i) \leq b_\lambda \cdot \bar{\Phi}\left(\frac{i - \frac{3}{2}}{\sqrt{\lambda}}\right)$$

Proposition 6. If $x > 0$, then $\bar{\Phi}(x) \lesssim \frac{\varphi(x)}{x}$ with error less than $\frac{\varphi(x)}{x^3}$, i.e. $\bar{\Phi}(x) \leq (1 + \frac{1}{x^2}) \frac{\varphi(x)}{x}$.

First applying Proposition 6. to Proposition 4. and 5. and then reparameterizing the bounds with the substitutions $(i - 3/2)/\sqrt{2} = k\sqrt{\lambda}$ and $i - 3/2 = k\sqrt{\lambda}$ gives the corollaries.

So we need to proof that proposition 1 to 6 hold, in order to proof the correctness of the corollaries.

lemma 1 For $k/\lambda > -1$ it holds

$$\ln(1 - k/\lambda) \leq -k/\lambda$$

8 Proofs

Proof. Let $f(x) = x - \ln(1+x)$, then $f'(x) = 1 - \frac{1}{1+x}$. As $x > -1$ is non-decreasing for all $x > -1$ and therefore $\ln(1-x) \leq -x$ for $x > -1$. Substituting $x = k/\lambda$ yields the lemma. \square

Proposition 1. Assume $\lambda > 0$. If $n > \lambda - 1$ and $l \geq 1$, then

$$Q_\lambda(n) \leq \left(1 - \left(\frac{\lambda}{n+1}\right)^l\right)^{-1} \cdot \sum_{k=\lceil n \rceil}^{\lceil n \rceil + l - 1} p_\lambda(k)$$

Proof. By definition it holds

$$Q_\lambda(n) = \sum_{k=\lceil n \rceil}^{\infty} p_\lambda(k).$$

An index-shifting of the summation results in

$$\begin{aligned} Q_\lambda(n) &= \sum_{k=\lceil n \rceil}^{\lceil n \rceil + l - 1} p_\lambda(k) + \sum_{k=\lceil n \rceil + l}^{\infty} p_\lambda(k) \\ &= \sum_{k=\lceil n \rceil}^{\lceil n \rceil + l - 1} p_\lambda(k) + \lambda^l \sum_{k=\lceil n \rceil}^{\infty} p_\lambda(k) \cdot \frac{k!}{(k+l)!} \end{aligned}$$

Further

$$\begin{aligned} Q_\lambda(n) &= \sum_{k=\lceil n \rceil}^{\lceil n \rceil + l - 1} p_\lambda(k) + \lambda^l \sum_{k=\lceil n \rceil}^{\infty} p_\lambda(k) \cdot \frac{k!}{(k+l)!} \\ &\leq \sum_{k=\lceil n \rceil}^{\lceil n \rceil + l - 1} p_\lambda(k) + \lambda^l \sum_{k=\lceil n \rceil}^{\infty} p_\lambda(k) \cdot \frac{1}{n+1} \\ &= \sum_{k=\lceil n \rceil}^{\lceil n \rceil + l - 1} p_\lambda(k) + \left(\frac{\lambda}{n+1}\right)^l Q_\lambda(n). \end{aligned}$$

Since $n+1 > \lambda$ and $l \geq 1$ we can now solve for $Q_\lambda(n)$.

This results in the formula of Proposition 1. \square

Proposition 2. Let $\lambda, i \leq 1$ and $m = \lfloor \lambda \rfloor$. Then,

1.

$$p_\lambda(m-i) \leq \frac{1}{\sqrt{2\pi m}} \exp\left(-\frac{i(i-1)}{2\lambda}\right),$$

2.

$$p_\lambda(m+i) \leq \frac{1}{\sqrt{2\pi m}} \exp\left(-\frac{i(i-1)}{2\lambda} + \frac{i(i-1)(2i-1)}{12\lambda^2}\right).$$

Proof. Let $m = \lfloor \lambda \rfloor$. We first show that statement 1. holds.

$$\begin{aligned} p_\lambda(m-i) &= p_\lambda(m) \cdot \left(\frac{m}{\lambda}\right) \cdot \left(\frac{m-1}{\lambda}\right) \cdots \left(\frac{m-i+1}{\lambda}\right) \\ &\leq p_\lambda(m) \cdot 1 \cdot \left(1 - \frac{1}{\lambda}\right) \cdots \left(1 - \frac{i-1}{\lambda}\right) = p_\lambda(m) \cdot \exp\left(\sum_{k=0}^{i-1} \ln\left(1 - \frac{k}{\lambda}\right)\right) \end{aligned}$$

By applying lemma 1, we get:

$$p_\lambda(m-i) \leq p_\lambda(m) \cdot \exp\left(\sum_{k=0}^{i-1} -\frac{k}{\lambda}\right) = p_\lambda(m) \cdot \exp\left(-\frac{i(i-1)}{2\lambda}\right).$$

We can now use the following Stirling formula-type inequality to approximate the formula $p_\lambda(m)$ [4, p.54].

$$m! > \sqrt{2\pi m} \cdot m^m \cdot e^{-m} \cdot e^{\frac{1}{12m+1}}$$

substituting ($b = \lambda - m$) and using the approximation for $m!$ yields:

$$\begin{aligned} p_\lambda(m) &= \frac{e^{-\lambda} \cdot \lambda^m}{m!} \leq \frac{e^{-\lambda} \cdot \lambda^m}{\sqrt{2\pi m} \cdot m^m \cdot e^{-m} \cdot e^{\frac{1}{12m+1}}} \leq \frac{e^{-\lambda} \cdot \lambda^m}{\sqrt{2\pi m} \cdot m^m \cdot e^{-m}} \\ &= \frac{1}{\sqrt{2\pi m}} \cdot e^{-b} \cdot \left(1 + \frac{b}{m}\right)^m \leq \frac{1}{\sqrt{2\pi m}} \cdot e^{-b} \cdot e^b \leq \frac{1}{\sqrt{2\pi m}} \\ \Rightarrow p_\lambda(m-i) &\leq p_\lambda(m) \cdot \exp\left(-\frac{i(i-1)}{2\lambda}\right) \leq \frac{1}{\sqrt{2\pi m}} \exp\left(-\frac{i(i-1)}{2\lambda}\right) \end{aligned}$$

Now we need to show that statement 2. holds.

Since $m \geq \lambda - 1$ it holds:

$$\begin{aligned} p_\lambda(m+i) &= p_\lambda(m) \cdot \frac{\lambda^i}{(m+1)(m+2)\cdots(m+i)} \\ &\leq p_\lambda(m) \cdot \frac{\lambda^i}{\lambda(\lambda+1)\cdots(\lambda+i-1)} = p_\lambda(m) \cdot \exp\left(-\sum_{k=0}^{i-1} \ln\left(1 + \frac{k}{\lambda}\right)\right) \\ &\leq p_\lambda(m) \cdot \exp\left(-\sum_{k=0}^{i-1} \left(\frac{k}{\lambda} - \frac{k^2}{2\lambda^2}\right)\right) \leq \frac{1}{\sqrt{2\pi m}} \exp\left(-\frac{i(i-1)}{2\lambda} + \frac{i(i-1)(2i-1)}{12\lambda^2}\right) \end{aligned}$$

□

Proposition 3. Suppose $\lambda \leq 2$. if $2 \geq i \geq \frac{\lambda+3}{2}$, then,

$$Q_\lambda(n) \leq \exp\left(\frac{1}{8\lambda}\right) \cdot \left(1 + \frac{1}{\lambda}\right) \cdot \sqrt{2} \cdot \left(1 - \left(\frac{\lambda}{m+i+1}\right)^m\right)^{-1} \cdot \Phi\left(\frac{i - \frac{3}{2}}{\sqrt{2\lambda}}\right)$$

8 Proofs

Proof. First apply Proposition 1. with $l = m$ and then do a shift in the sum to get:

$$\begin{aligned} Q_\lambda(m+i) &\leq \left(1 - \left(\frac{\lambda}{m+i+1}\right)^m\right)^{-1} \cdot \sum_{k=\lceil m+i \rceil}^{\lceil m+i \rceil + m - 1} p_\lambda(k) \\ &= \left(1 - \left(\frac{\lambda}{m+i+1}\right)^m\right)^{-1} \cdot \sum_{k=\lceil i \rceil}^{m+\lceil i \rceil - 1} p_\lambda(m+k) \end{aligned}$$

For $i \leq k \leq m+i-1$, it holds:

$$-\frac{k(k-1)}{2\lambda} + \frac{(k-1)k(2k-1)}{12\lambda} \leq -\frac{k(k-1)}{2\lambda} \cdot \left(1 - \frac{m+i}{3\lambda} + \frac{1}{2\lambda}\right).$$

Let β be $\beta = 1 - \frac{m+i}{3\lambda} + \frac{1}{2\lambda}$.

By using Proposition 2. for the $p_\lambda(m+k)$ part and applying the statement from above we get

$$Q_\lambda(m+i) \leq \left(1 - \left(\frac{\lambda}{m+i+1}\right)^m\right)^{-1} \cdot \frac{1}{\sqrt{2\pi m}} \cdot \sum_{k=\lceil i \rceil}^{m+\lceil i \rceil - 1} \exp\left(\frac{-k(k-1)}{2} \cdot \beta\right).$$

For the latter sum it holds (if $i \geq 2$)

$$\sum_{k=\lceil i \rceil}^{m+\lceil i \rceil - 1} \exp\left(\frac{-k(k-1)}{2} \cdot \beta\right) \leq \int_{i-1}^{\infty} \exp\left(\frac{-k \cdot (k-1) \cdot \beta}{2\lambda}\right) dk$$

substituting $t = (k - \frac{1}{2})\sqrt{\lambda/\beta}$ yields:

$$\begin{aligned} &\int_{(i-\frac{3}{2})\sqrt{\beta/\lambda}}^{\infty} \exp\left(\frac{-\left(t \cdot \sqrt{\lambda/\beta} + \frac{1}{2}\right)\left(t \cdot \sqrt{\lambda/\beta} - \frac{1}{2}\right)}{2\lambda}\right) \cdot \sqrt{\lambda/\beta} dt \\ &= \sqrt{\lambda/\beta} \cdot \exp\left(\frac{\beta}{8\lambda}\right) \cdot \sqrt{2\pi} \cdot \bar{\Phi}\left(\left(i - \frac{3}{2}\right) \sqrt{\beta/\lambda}\right) \end{aligned}$$

Furthermore, it holds $\beta \leq 1$ since $m+i \geq i \geq 2$ and since $i \leq (\lambda+3)/2$, it follows that $\beta \geq 1 - \frac{\lambda+(\lambda+3)/2}{3\lambda} + \frac{1}{2\lambda} = \frac{1}{2}$. Hence, $1/\sqrt{\beta} \leq \sqrt{2}$, and $\exp\left(\frac{\beta}{8\lambda}\right) \leq \exp\left(\frac{1}{8\lambda}\right)$ therefore it holds

$$\bar{\Phi}\left(\left(i - \frac{3}{2}\right) \sqrt{\beta/\lambda}\right) \leq \bar{\Phi}\left(\left(i - \frac{3}{2}\right) \cdot \frac{1}{\sqrt{2\lambda}}\right)$$

combining these inequalities and the previously obtained $\sqrt{\lambda/m} \leq 1 + \frac{1}{\lambda}$ we get

$$\begin{aligned} Q_\lambda(m+i) &\leq \left(1 - \left(\frac{\lambda}{m+i+1}\right)^m\right)^{-1} \cdot \frac{1}{\sqrt{2\pi m}} \cdot \sqrt{\lambda/\beta} \cdot \exp\left(\frac{\beta}{8\lambda}\right) \cdot \sqrt{2\pi} \cdot \bar{\Phi}\left(\left(i - \frac{3}{2}\right) \sqrt{\beta/\lambda}\right) \\ &\leq \left(1 - \left(\frac{\lambda}{m+i+1}\right)^m\right)^{-1} \cdot \sqrt{\lambda/m} \cdot \sqrt{2} \cdot \exp\left(\frac{1}{8\lambda}\right) \cdot \bar{\Phi}\left(\left(i - \frac{3}{2}\right) \cdot \sqrt{\frac{1}{2\lambda}}\right) \\ &\leq \left(1 - \left(\frac{\lambda}{m+i+1}\right)^m\right)^{-1} \cdot \left(1 + \frac{1}{\lambda}\right) \cdot \sqrt{2} \cdot \exp\left(\frac{1}{8\lambda}\right) \cdot \bar{\Phi}\left(\frac{i - \frac{3}{2}}{\sqrt{2\lambda}}\right) \end{aligned}$$

□

Proposition 4. Suppose $\lambda \geq 400$. if $2 \leq i \leq \frac{\lambda+3}{2}$ and $m = \lfloor \lambda \rfloor$, then,

$$Q_\lambda(m+i) \leq \frac{a_\lambda}{1 - \exp\left(-\frac{2}{9}i\right)} \cdot \bar{\Phi}\left(\frac{i - \frac{3}{2}}{\sqrt{2\lambda}}\right)$$

Proof. Recall that $a_\lambda = \left(1 + \frac{1}{\lambda}\right) \cdot \sqrt{2} \cdot \exp\left(\frac{1}{16}\right)$. For $\lambda \geq 2$, $\exp\left(\frac{1}{8\lambda}\right) \leq \exp\left(\frac{1}{16}\right)$. Using proposition 3 we only need to show that

$$\left(1 - \left(\frac{\lambda}{m+i+1}\right)^m\right)^{-1} \leq \left(1 - \exp\left(-\frac{2}{9}i\right)\right)^{-1}.$$

Since $m+1 \geq \lambda$, it follows that $\lambda/(m+i+1) \leq \lambda/(\lambda+i) = 1 - (i/(\lambda+i))$. Further by Lemma 1 it holds

$$\begin{aligned} 1 - \frac{i}{\lambda+i} &\leq \exp\left(-\frac{i}{\lambda+i}\right) \\ \Rightarrow \frac{\lambda}{m+i} &\leq \exp\left(-\frac{i}{\lambda+i}\right) \\ \Rightarrow \left(\frac{\lambda}{m+i}\right)^m &\leq \exp\left(-\frac{i \cdot m}{\lambda+i}\right) \end{aligned}$$

The function $f(x) = (x-1)/(x+1)$ is non-decreasing on $[0, \infty)$ so $f(x) \geq f(2) = \frac{1}{3}$ for $x \geq 2$. This, for $\lambda \geq 2$ $(\lambda-1)/(\lambda+1) \geq 1/3$, and since $m = \lfloor \lambda \rfloor$ it also holds $m \leq \frac{1}{3}(\lambda+1)$. By $i \leq (\lambda+3)/2$ it follows

$$m \cdot (\lambda+i)^{-1} \geq m \cdot \left(\lambda + \frac{\lambda+3}{2}\right)^{-1} \geq \left(\frac{1}{3} \cdot (\lambda+1)\right) \cdot \left(\frac{2}{3} \cdot (\lambda+1)\right)^{-1} = \frac{2}{9}.$$

Since we are using this proposition only for $\lambda \geq 400$ we can improve the bound to $m \cdot (\lambda+1)^{-1} \geq \frac{266}{401}$ using $m \geq f(400)(\lambda+1) = \frac{399}{401}(\lambda+1)$.

The relation then yields

$$\left(\frac{\lambda}{m+i+1}\right)^m \leq \exp\left(-\frac{266}{401}i\right) \leq \exp\left(-\frac{2}{9}i\right).$$

for $\lambda \geq 400$. from which proposition 4 follows immediately. □

Proposition 5. Suppose $\lambda \geq 2$ and $i \geq 2$. Then

$$T_\lambda(m-i) \leq b_\lambda \cdot \bar{\Phi} \left(\frac{i - \frac{3}{2}}{\sqrt{\lambda}} \right)$$

Proof. By applying an indexshift to the basic definition of $T_\lambda(i)$ we get:

$$T_\lambda(m-i) = \sum_{k=\lceil i \rceil}^m p_\lambda(m-k)$$

By Proposition 2 it follows

$$T_\lambda(m-i) \leq \frac{1}{\sqrt{2\pi m}} \cdot \sum_{k=\lceil i \rceil}^m \exp \left(-\frac{k(k-1)}{2\lambda} \right)$$

Since $f(x) = -x(x-1)/2\lambda$ is non-increasing on $[\frac{1}{2}, \infty)$, for $x \geq 3/2$ it holds

$$\begin{aligned} \exp \left(-\frac{k(k-1)}{2\lambda} \right) &\leq \int_{x-1}^x \exp \left(-\frac{u(u-1)}{2\lambda} \right) du \\ &= \int_{x-1}^x \exp \left(\frac{-(u-1/2)^2 + (1/2)^2}{2\lambda} \right) du \\ &= \exp \left(\frac{(1/2)^2}{2\lambda} \right) \int_{x-1}^x \exp \left(-\frac{(u-1/2)^2}{2\lambda} \right) du \end{aligned}$$

Thus, if $i \geq 2$,

$$T_\lambda(m-i) \leq \frac{1}{\sqrt{2\pi m}} \cdot \exp \left(\frac{1}{8\lambda} \right) \int_{(i-1)}^{\infty} \exp \left(-\frac{(u-\frac{1}{2})^2}{2\lambda} \right) du$$

substituting $t = \frac{u-\frac{1}{2}}{\lambda}$ yields

$$\begin{aligned} T_\lambda(m-i) &\leq \frac{1}{\sqrt{2\pi m}} \cdot \exp \left(\frac{1}{8\lambda} \right) \cdot \int_{(i-\frac{3}{2})/\lambda}^{\infty} \exp \left(-\frac{t^2}{2} \right) \sqrt{\lambda} dt \\ &= \frac{1}{\sqrt{2\pi m}} \cdot \exp \left(\frac{1}{8\lambda} \right) \cdot \sqrt{\lambda} \cdot \sqrt{2\pi} \cdot \bar{\Phi} \left(\frac{i - \frac{3}{2}}{\sqrt{\lambda}} \right) = \exp \left(\frac{1}{8\lambda} \right) \cdot \sqrt{\frac{\lambda}{m}} \cdot \bar{\Phi} \left(\frac{i - \frac{3}{2}}{\sqrt{\lambda}} \right) \end{aligned}$$

By Lemma 1. $\sqrt{\lambda/m} = \sqrt{1 + (b/m)} = \sqrt{1 + (b/m)} \leq 1 + b/(2m)$. For $\lambda \geq 2$ it holds:

$$\lambda \leq \lfloor \lambda \rfloor + 1 \leq \lfloor \lambda \rfloor + 2 - 2/\lambda = \lfloor \lambda \rfloor + (2/\lambda) \cdot (\lambda - 1) \leq \lfloor \lambda \rfloor + 2/\lambda \cdot \lfloor \lambda \rfloor$$

with $b = \lambda - m$. It follows $b = \lambda - \lfloor \lambda \rfloor \leq 2\lfloor \lambda \rfloor/\lambda = 2m/\lambda$, thus $b/(2m) \leq 1/\lambda$.

With $\sqrt{\lambda/m} \leq (1 + 1/\lambda)$ it follows:

$$\begin{aligned} T_\lambda(m-i) &\leq \exp \left(\frac{1}{8\lambda} \right) \cdot \sqrt{\frac{\lambda}{m}} \cdot \bar{\Phi} \left(\frac{i - \frac{3}{2}}{\sqrt{\lambda}} \right) \\ &\leq \exp \left(\frac{1}{8\lambda} \right) \cdot \left(1 + \frac{1}{\lambda} \right) \cdot \bar{\Phi} \left(\frac{i - \frac{3}{2}}{\sqrt{\lambda}} \right) \end{aligned}$$

□

Proposition 6. If $x > 0$, then $\bar{\Phi}(x) \lesssim \frac{\varphi(x)}{x}$ with error less than $\frac{\varphi(x)}{x^3}$, i.e. $\bar{\Phi}(x) \leq (1 + \frac{1}{x^2})\frac{\varphi(x)}{x}$

This Proposition will only be used for the range of 3 to at most 10 in a practical implementation. Within this range the expression φ/x already overestimates $\bar{\Phi}$ by more than 5% [7].

Bibliography

- [1] Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert Brayton. Model-checking continuous-time markov chains, 2000.
- [2] Christel Baier and Joost-Pieter Katoen. Principles of model checking, 2008.
- [3] S. Donatelli, J. Hillston, and M. Ribaud. A comparison of performance evaluation process algebra and generalized stochastic petri nets, 1995.
- [4] William Feller. An introduction to probability theory and its applications, 1968.
- [5] Richard Fernow. Introduction to experimental particle physics, 1986.
- [6] Bennett L. Fox and Peter W. Glynn. Computing poisson probabilities, 1988.
- [7] David N. Jansen. Understanding fox and glynn's "computing poisson probabilities", 2011.
- [8] Joost-Pieter Katoen, Holger Hermanns, Boudewijn Haverkort, and Christel Baier. Model checking continuous-time markov chains by transient analysis, 2000.
- [9] 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, 2010.
- [10] Marta Kwiatkowska, Norman Gethin, and Parker David. Stochastic model checking, 2007.
- [11] Marta Kwiatkowska, Norman Gethin, and Parker David. Prism 4.0: Verification of probabilistic real-time systems, 2011.
- [12] M. Ajmone Marsan. Stochastic petri nets: An elementary introduction, 2005.
- [13] Ali Movaghar and Abdollahi Azgomi. A modelling tool for hierarchical stochastic activity networks, 2005.
- [14] H. Allen Orr. The distribution of fitness effects among beneficial mutations, 2003.
- [15] Anna Philippou, Insup Lee, and Oleg Sokolsky. Weak bisimulation for probabilistic systems, 2000.
- [16] Aviv Regev, Ekaterina M Panina, William Silverman, Luca Cardelli, and Ehud Shapiro. Bioambients: an abstraction for biological compartments, 2004.

Measured Values

Used Model	Parameter	Checked Formula	NNZ	StoRM		PRISM		MRMC	
				RAM (byte)	Time (ms)	RAM (byte)	Time (ms)	RAM (byte)	Time (ms)
Working Cluster Model	5	true $U^{\{100\}}$!minimum	5393	5365760	171	68008000	15	1520000	16
Working Cluster Model	10	true $U^{\{100\}}$!minimum	19553	6459392	272	68480000	397	2032000	96
Working Cluster Model	50	true $U^{\{100\}}$!minimum	449633	42389504	2714	74408000	9610	20560000	3683
Working Cluster Model	100	true $U^{\{100\}}$!minimum	1779233	171585536	26577	85964000	43819	77208000	25805
Working Cluster Model	150	true $U^{\{100\}}$!minimum	3988833	382582784	60653	106012000	98599	171792000	60422
Working Cluster Model	200	true $U^{\{100\}}$!minimum	7078433	653373440	110482	133756000	180181	303624000	106452
Working Cluster Model	350	true $U^{\{100\}}$!minimum	21627232	2087862272	368782	264712000	548887	923164000	328002
Working Cluster Model	500	true $U^{\{100\}}$!minimum	44096033	4409761792	845760	710512000	1464310	1871448000	718978
Working Cluster Model	750	true $U^{\{100\}}$!minimum	99144032	9242472448	1931887	1495068000	3245969	3608544000	1875786

Number of Matrix Multiplications performed for checking the above formula

Storm: 5543

Prism: 5750

MRMC: 4449

Used Model	Parameter	Checked Formula	NNZ	StoRM		PRISM		MRMC	
				RAM (byte)	Time (ms)	RAM (byte)	Time (ms)	RAM (byte)	Time (ms)
Working Cluster Model	5	minimum $U^{\{20,50\}}$ premium	5393	5443584	75	68044000	90	1532000	9
Working Cluster Model	10	minimum $U^{\{20,50\}}$ premium	19553	6688768	115	68876000	191	2168000	30
Working Cluster Model	50	minimum $U^{\{20,50\}}$ premium	449633	44593152	1258	74768000	3149	21128000	1320
Working Cluster Model	100	minimum $U^{\{20,50\}}$ premium	1779233	171552768	8508	87820000	13653	79508000	9932
Working Cluster Model	150	minimum $U^{\{20,50\}}$ premium	3988833	382550016	20852	113756000	34201	176088000	21453
Working Cluster Model	200	minimum $U^{\{20,50\}}$ premium	7078433	664526848	39681	147392000	61530	312512000	38050
Working Cluster Model	350	minimum $U^{\{20,50\}}$ premium	21627233	2087866368	123036	269396000	211209	939012000	116704
Working Cluster Model	500	minimum $U^{\{20,50\}}$ premium	44096033	4409753600	282820	539040000	433556	1876588000	269665
Working Cluster Model	750	minimum $U^{\{20,50\}}$ premium	99144033	8775946204	741611	1102852000	940187	3624424000	695361

Number of Matrix Multiplications performed for checking the above formula

Storm: 2469

Prism: 2606

MRMC: 2449

Used Model	Parameter	Checked Formula	StoRM		PRISM		MRMC	
			NNZ	RAM (byte)	Time (ms)	RAM (byte)	Time (ms)	RAM (byte)
Working Cluster Model	5	true $U^{\wedge}\{20\}$!minimum	5393	5353472	62	67940000	45	not computable
Working Cluster Model	10	true $U^{\wedge}\{20\}$!minimum	19553	6426624	75	67048000	88	not computable
Working Cluster Model	50	true $U^{\wedge}\{20\}$!minimum	449633	41893888	326	73404000	1406	not computable
Working Cluster Model	100	true $U^{\wedge}\{20\}$!minimum	1779233	171565056	1461	82608000	6018	not computable
Working Cluster Model	150	true $U^{\wedge}\{20\}$!minimum	3988833	383893504	6423	99728000	14529	not computable
Working Cluster Model	200	true $U^{\wedge}\{20\}$!minimum	7078433	654397440	14877	120896000	26256	not computable
Working Cluster Model	500	true $U^{\wedge}\{20\}$!minimum	44096033	4408959232	338652	464016000	170138	not computable

Number of Matrix Multiplications performed for checking the above formula

Storm: 1037 Prism: 1158 MRMC: 0

Used Model	Parameter	Checked Formula	StoRM		PRISM		MRMC		
			NNZ	RAM (byte)	Time (ms)	RAM (byte)	Time (ms)	RAM (byte)	Time (ms)
Working Cluster Model	5	true $U^{\wedge}\{20\}$!minimum	5393	5365760	71	68008000	55	1520000	12
Working Cluster Model	10	true $U^{\wedge}\{20\}$!minimum	19553	6459392	100	68480000	125	2032000	23
Working Cluster Model	50	true $U^{\wedge}\{20\}$!minimum	449633	42389504	723	74408000	2409	20560000	926
Working Cluster Model	100	true $U^{\wedge}\{20\}$!minimum	1779233	171585536	6975	85964000	10844	77208000	6307
Working Cluster Model	150	true $U^{\wedge}\{20\}$!minimum	3988833	382582784	16218	106012000	25681	171792000	14056
Working Cluster Model	200	true $U^{\wedge}\{20\}$!minimum	7078433	653373440	31176	133756000	48945	303624000	24542
Working Cluster Model	350	true $U^{\wedge}\{20\}$!minimum	21627233	2087862272	109812	264712000	159392	923164000	81947
Working Cluster Model	500	true $U^{\wedge}\{20\}$!minimum	44096033	4409761792	279349	710512000	364000	1871448000	166383
Working Cluster Model	750	true $U^{\wedge}\{20\}$!minimum	99144033	9242472448	815837	1495068000	1020000	3608544000	428744

Number of Matrix Multiplications performed for checking the above formula

Storm: 1233 Prism: 1302 MRMC: 1002

Used Model	Parameter	Checked Formula	StoRM		PRISM		MRMC		
			NNZ	RAM (byte)	Time (ms)	RAM (byte)	Time (ms)	RAM (byte)	Time (ms)
Embedded control system	50	true $U \leq (432000)$ down	185856	25362432	3515	70164000	18011	10368000	1881979
Embedded control system	500	true $U \leq (432000)$ down	1791006	222339072	73409	105732000	213730	out of bound	out of bound
Embedded control system	1000	true $U \leq (432000)$ down	3574506	456560640	192612	99656000	470924	out of bound	out of bound
Embedded control system	2000	true $U \leq (432000)$ down	7141506	860065792	433388	155416000	1280231	out of bound	out of bound
Embedded control system	5000	true $U \leq (432000)$ down	17842506	2123247616	1207924	223928000	3058025	out of bound	out of bound

Number of Matrix Multiplications performed for checking the above formula

Storm: 37344 Prism: 38348 MRMC: 9093300