**THÈSE DE DOCTORAT**
**DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN**

Présentée par

Monsieur Benoît BARBOT

**Pour obtenir le grade de**

**DOCTEUR DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN**

Domaine :
**Informatique**

**Sujet de la thèse :**

# Accélération pour le Model Checking Statistique

Thèse présentée et soutenue à Cachan le 20 novembre 2014 devant le jury composé de :

| | | |
|---|---|---|
| Pieter-Tjerk DE BOER | Assistant Professor | Rapporteur |
| Serge HADDAD | Professeur | Co-directeur de thèse |
| Patrice MOREAUX | Professeur | Examinateur |
| David PARKER | Lecturer | Examinateur |
| Nihal PEKERGIN | Professeur | Examinatrice |
| Claudine PICARONNY | Maître de Conférences | Co-directrice de thèse |
| Gerardo RUBINO | Professeur | Rapporteur |

Laboratoire Spécification et Vérification
ENS de Cachan, UMR 8643 du CNRS
61, avenue du Président Wilson
94235 CACHAN Cedex, France

English title : Acceleration for Statistical Model Checking

# Abstract

In the past decades, the analysis of complex critical systems subject to uncertainty has become more and more important. In particular the quantitative analysis of these systems is necessary to guarantee that their probability of failure is very small. As their state space is extremly large and the probability of interest is very small, typically less than one in a billion, classical methods do not apply for such systems.

Model Checking algorithms are used for the analysis of probabilistic systems, they take as input the system and its expected behaviour, and compute the probability with which the system behaves as expected. These algorithms have been broadly studied. They can be divided into two main families: Numerical Model Checking and Statistical Model Checking. The former computes small probabilities accurately by solving linear equation systems, but does not scale to very large systems due to the space size explosion problem. The latter is based on Monte Carlo Simulation and scales well to big systems, but cannot deal with small probabilities.

The main contribution of this thesis is the design and implementation of a method combining the two approaches and returning a confidence interval of the probability of interest. This method applies to systems with both continuous and discrete time settings for time-bounded and time-unbounded properties.

All the variants of this method rely on an abstraction of the model, this abstraction is analysed by a numerical model checker and the result is used to steer Monte Carlo simulations on the initial model. This abstraction should be small enough to be analysed by numerical methods and precise enough to improve the simulation. This abstraction can be build by the modeller, or alternatively a class of systems can be identified in which an abstraction can be automatically computed.

This approach has been implemented in the tool Cosmos, and this method was successfully applied on classical benchmarks and a case study.

# Résumé

Ces dernières années, l'analyse de systèmes complexes critiques est devenue de plus en plus importante. En particulier, l'analyse quantitative de tels systèmes est nécessaire afin de pouvoir garantir que leur probabilité d'échec est très faible. La difficulté de l'analyse de ces systèmes réside dans le fait que leur espace d'état est très grand et que la probabilité recherchée est extrêmement petite, de l'ordre d'une chance sur un milliard, ce qui rend les méthodes usuelles inopérantes.

Les algorithmes de Model Checking quantitatif sont les algorithmes classiques pour l'analyse de systèmes probabilistes. Ils prennent en entrée le système et son comportement attendu et calculent la probabilité avec laquelle les trajectoires du système correspondent à ce comportement. Ces algorithmes de Model Checking ont été largement étudiés depuis leurs créations. Deux familles d'algorithmes existent : le Model Checking numérique réduit le problème à la résolution d'un système d'équations. Il permet de calculer précisément des petites probabilités mais souffre du problème d'explosion combinatoire; le Model Checking statistique est basé sur la méthode de Monte-Carlo qui se prête bien à l'analyse de très gros systèmes mais qui ne permet pas de calculer de petites probabilités.

La contribution principale de cette thèse est le développement d'une méthode combinant les avantages des deux approches et qui renvoie un résultat sous forme d'intervalles de confiance. Cette méthode s'applique à la fois aux systèmes discrets et continus pour des propriétés bornées ou non bornées temporellement.

Cette méthode est basée sur une abstraction du modèle qui est analysée à l'aide de méthodes numériques, puis le résultat de cette analyse est utilisé pour guider une simulation du modèle initial. Ce modèle abstrait doit à la fois être suffisamment petit pour être analysé par des méthodes numériques et suffisamment précis pour guider efficacement la simulation. Dans le cas général, cette abstraction doit être construite par le modélisateur. Cependant, une classe de systèmes probabilistes a été identifiée dans laquelle le modèle abstrait peut être calculé automatiquement.

Cette approche a été implémentée dans l'outil Cosmos et des expériences sur des modèles de référence ainsi que sur une étude de cas ont été effectuées, qui montrent l'efficacité de la méthode.

# Remerciements

Je remercie mes encadrants de thèse Claudine Picaronny et Serge Haddad pour leur patience et leur gentillesse, ainsi que pour les conseils et les encouragements qu'ils m'ont prodigués tout au long de ces trois années. Grâce à leurs qualités scientifiques et humaines, ils m'ont permis de développer mon esprit critique et mon autonomie et de découvrir les différentes facettes du métier d'enseignant-chercheur. Je remercie en particulier Serge pour m'avoir fait rencontrer un grand nombre de chercheurs de différents horizons géographiques et scientifiques et pour m'avoir encouragé à travailler avec eux.

Je tiens également à remercier les deux rapporteurs de cette thèse Pieter-Tjerk De Boer et Gerardo Rubino qui ont accepté d'évaluer ce manuscrit et m'ont fait part de remarques très pertinentes pour la suite de mes travaux, ainsi que Patrice Moreaux, David Parker et Nihal Pekergin pour avoir accepté de participer à mon jury de thèse.

Je remercie tous les membres du LSV qui m'ont chaleureusement accueilli et avec qui j'ai pu discuter et échanger sur mes travaux pendant ces trois ans. Je remercie Paolo Ballarini, Hilal Djafri et Nihal Perkergin, pour leurs collaborations sur l'outil Cosmos qui m'ont permis de participer à l'implémentation d'un outil de vérification. Je remercie également Elvio Amparore, Marco Beccuti, Susanna Donatelli et Giuliana Franceschinis ainsi que les membres du département informatique de l'université de Turin pour m'avoir accueilli dans leur laboratoire lors de mon séjour en Italie. Je tiens à mentionner que participer au projet CosyVerif qui m'a sensibilisé au problème de la visibilité de la recherche a été une exprérience très enrichissante. Merci à tous les participants de ce projet de m'avoir intégré dans leur équipe et en particulier à Maximilien Colange, Clément Desmoulin et Alban Linard avec qui j'ai le plus étroitement collaboré.

Je tiens à remercier tous les membres du LSV qui ont rendu l'ambiance si conviviale et en particulier ceux avec qui j'ai collaboré sur des enseignements Cesar Rodriguez, David Baelde, Claudine Picaronny et Micheèle Sebag. Un grand merci à l'équipe administrative et technique qui m'a permis de travailler dans des conditions optimales. Je remercie ceux qui ont partagé mon bureau : Aiswarya, Benjamin, Patrick, Jérémie, Asalé et Baptiste pour l'ambiance studieuse et amicale qu'ils y ont instauré. Merci aussi à tous les

nageurs du LSV qui m'ont aidé à garder la forme, j'ai en particulier une pensée pour Alban, Virginie, Paul, Thida, Sophie, Claudine et Francis. Je remercie tous les doctorants du LSV pour leurs camaraderie. Je tiens à remercier chaleureusement Aiswarya et Benjamin dont les 'chamailleries' ont animé ma vie au LSV, ainsi que Cesar que je n'ai jamais pu convaincre des bienfaits de la cuisine maison.

J'aimerai enfin remercier toute ma famille, mes parents pour m'avoir motivé, soutenu et donné le goût d'apprendre, Paul pour m'avoir le premier initié à l'informatique et Antoine pour être un si bon conteur. Pour finir je remercie Elisa pour son aide, son amour et pour m'avoir aidé à garder confiance en moi.

Benoît Barbot
Oxford, Novembre 2014

# Contents

**Abstract** **3**

**Résumé** **4**

**Remerciements** **5**

**Contents** **7**

**1 Introduction** **11**
1.1 Model Checking . . . . . 12
1.2 Rare events . . . . . 12
1.3 Classical Methods for Probability Estimation . . . . . 13
1.4 Contributions . . . . . 15
1.5 Organization . . . . . 16

**I Preliminaries** **17**

**2 Technical Background** **18**
2.1 Introduction . . . . . 18
2.2 Models . . . . . 18
2.2.1 Discrete Event Dynamic System . . . . . 18
2.2.2 Discrete Time Markov Chain . . . . . 21
2.2.3 Continuous Time Markov Chain . . . . . 30
2.2.4 Higher Level Model . . . . . 35
2.3 Statistical Methods . . . . . 42
2.3.1 Monte Carlo Method . . . . . 42
2.3.2 Confidence interval . . . . . 43
2.3.3 Gaussian Confidence Interval . . . . . 46
2.3.4 Chernoff-Hoeffding Bounds . . . . . 48
2.3.5 Confidence Interval for Binomial law . . . . . 48
2.3.6 Hypothesis testing . . . . . 49
2.4 Model Checking . . . . . 50
2.4.1 Specification . . . . . 50

2.4.2 Numerical Model Checking . . . . . 52
2.4.3 Statistical Model Checking . . . . . 53
2.4.4 Comparison Between Numerical and Statistical Model Checking . . . . . 53

**3 Rare Events** **56**
3.1 Introduction . . . . . 56
3.2 Rare-Event Problem . . . . . 56
3.3 Splitting . . . . . 59
3.4 Importance Sampling . . . . . 63
3.4.1 Definition . . . . . 63
3.4.2 Zero Variance Importance Sampling . . . . . 65
3.4.3 Infinite Variance Importance Sampling . . . . . 67
3.4.4 Distribution of $W_{s_0}$ . . . . . 68
3.4.5 Efficiency of Importance sampling . . . . . 69
3.4.6 State of the Art on Importance Sampling . . . . . 69
3.5 Conclusion . . . . . 73

**II Theoretical Contributions** **74**

**4 Guaranteed Variance Reduction** **75**
4.1 Introduction . . . . . 75
4.2 A New Approach for Importance Sampling . . . . . 76
4.2.1 Principle of the Method . . . . . 76
4.2.2 Reduced Model . . . . . 77
4.2.3 Guaranteed Variance Reduction . . . . . 79
4.2.4 Structural Guarantee . . . . . 81
4.3 Reachability Analysis for Markov Chain . . . . . 87
4.3.1 General Importance Sampling . . . . . 87
4.3.2 Importance Sampling with Guaranteed Variance . . . 90
4.4 Time-Bounded Reachability for DTMC . . . . . 93
4.4.1 Challenge . . . . . 93
4.4.2 Adapting Reachability Analysis . . . . . 93
4.4.3 Algorithmic Considerations . . . . . 97
4.5 Time-Bounded Reachability for CTMC . . . . . 103
4.5.1 Transient Analysis . . . . . 104
4.5.2 Guaranteed Variance for CTMC . . . . . 106
4.5.3 More on Simulation. . . . . 109
4.6 From Model Checking to Reachability . . . . . 111
4.6.1 From Until Formula to Reachability . . . . . 111
4.6.2 From Model Checking Against Finite State Automaton to Reachability . . . . . 112
4.7 Conclusion . . . . . 113

**5 Patterns for Stochastic Bounds** **114**
5.1 Introduction . . . 114
5.2 Framework . . . 115
5.2.1 Syntax . . . 115
5.2.2 Operational Semantic . . . 117
5.3 Symmetries . . . 124
5.4 Coupling . . . 128
5.5 Application to Guaranteed Variance Reduction . . . 134
5.6 Conclusion . . . 137

**III Applications** **139**

**6 Cosmos** **140**
6.1 Introduction . . . 140
6.2 Description of the Tool . . . 141
6.2.1 Architecture . . . 141
6.2.2 The HASL Logic . . . 144
6.2.3 Statistical Procedures . . . 148
6.3 Integration of Importance Sampling . . . 150
6.3.1 Distribution Parameters . . . 150
6.3.2 State-Space Generation and Numerical Computation . 152
6.3.3 Fox-Glynn Algorithm and Uniformization . . . 153
6.4 CosyVerif . . . 153
6.4.1 Description . . . 153
6.4.2 Personal Contributions . . . 154
6.5 Stochastic Symmetric Net for Cosmos . . . 156
6.5.1 Extension of HASL . . . 157
6.5.2 Implementation of SSN in Cosmos . . . 158
6.5.3 Runtime Comparison of Symmetric Net Simulator vs Petri net Simulator . . . 160
6.6 Comparison Between Statistical Model Checkers . . . 162
6.6.1 Expressiveness Comparison . . . 162
6.6.2 Runtime Comparison of Statistical Model Checkers . . 164
6.7 Importance Sampling Benchmark . . . 168
6.7.1 Global overflow in tandem queues . . . 168
6.7.2 Local Overflow in Tandem Queues . . . 172
6.7.3 Bottleneck in Tandem Queues . . . 174
6.7.4 Parallel Random Walk . . . 174
6.7.5 The Dining Philosophers . . . 176
6.8 Conclusion . . . 177

**7 Signaling Cascade** **179**
7.1 Introduction . . . 179
7.2 Biological Background . . . 180
7.3 Petri net modeling . . . 182
7.4 Experiments . . . 184
7.4.1 Maximal peak of the output signal . . . 185
7.4.2 Conditional maximal signal peak . . . 189
7.4.3 Signal propagation . . . 190
7.5 Conclusion . . . 191

**8 Conclusion and Perspectives** **193**
8.1 Conclusion . . . 193
8.2 Perspectives . . . 194

**List of Publication** **195**

**Bibliography** **196**

# Chapter 1

# Introduction

When one is interested in the analysis of a given system, the usual approach consists in building a mathematical model of this system and in performing the analysis on the model rather than on the initial system. The benefits of such an approach are multiple and can be seen in many fields:

In engineering, modelling can very useful during the whole lifetime of a product. For instance, during the design of a system, the model can be modified to explore several design options or formal analysis and simulation can be conducted to ensure the correct behavior of the system. As these tests do not require to build any actual prototype, they are indeed much cheaper and faster than the traditional ones. During the exploitation of the system, the model can also be used to help maintenance or understand fault so that the next generation will be more effective. The use of models has been successful in several fields of engineering, as for example in mechanical engineering, electrical or software engineering.

In research,in most scientific fields, models are nowadays used to help with the comprehension of a system. In particular, models are used to understand or predict a specific behavior of a system without dealing with its whole complexity. In fact, the complete analysis could be either too expensive or too difficult but also unnecessary. The difficulty is then to abstract the system in a "smart" way in order to build a minimal pertinent mathematical model of the system for the property of interest.

Although applied in different fields and on different systems, similar trends can be found among the methods used for system analysis. The most common ones are *simulation* where the model is executed as the real system so that observations can be conducted on the simulated system; *verification* where all the states of a modeled system are explored to ensure that no faulty state can be reached; *synthesis* where parts of a model constrained by other parts are automatically build; and *evaluation* where the efficiency of a model is estimated.

The study and development of efficient methods for system analysis

constitutes therefore the object of an active research in computer science. Within this thesis, we will focus on one particular method, namely *Model Checking*.

## 1.1 Model Checking

In *Model Checking* [31], one is interested in checking if a software or hardware system meets its specification. In this case, the system is modeled as a transition system while its specification is modeled as a formula in a given logic. An exhaustive exploration of the state space can then be performed in order to determine whether or not a faulty state is reached.

This approach has been broadly studied for the last decades, leading to the development of several effective tools. Historically, model checking has been first applied to non-probabilistic programs using several specification languages like LTL [74] or CTL [30]. However, some systems are subject to unsettled behaviors which may be governed by probabilistic laws. In those systems, standard model checking is not very helpful as it can return only a boolean (this state is reached) but gives no information about the frequency of this event to actually occur. In order to get access to this information, one then has to use *quantitative* model checking instead.

Two main sources of randomness can be identified:

- A system can explicitly contain randomness. For example, a network protocol often includes a randomized algorithm to break symmetry [69].
- Randomness may also be used to model unknown parts of a system. For instance, one can model the physical part of a system or its interaction with its environment with probabilities. In the case of a Cyber-physical system, a deterministic algorithm is then entangled with a physical part interacting with the environment which is modeled as a stochastic process [22].

Quantitative model checking stands therefore at the junction between model checking and performance evaluation. It has been applied to a broad range of probabilistic systems. However, some particular cases still requires special attention as for instance rare events.

## 1.2 Rare events

Rare events are events that appear with a very small probability, like the simultaneous failure of two redundant components in a critical system or an overflow in a stable telecommunication system. Indeed human intuition is far from correct when it comes to the estimation of such small probabilities. For example, the report on the Challenger space shuttle accident by Richard

Feynmann [32] starts with the following sentences *"It appears that there are enormous differences of opinion as to the probability of a failure with loss of vehicle and of human life. The estimates range from roughly 1 in 100 to 1 in 100,000."*, Richard Feynmann notes that even experts on a system cannot estimate accurately small probabilities and are prone to underestimation. In the context of mathematical finance, Nassim Nicholas Taleb developped the *Black Swan* theory in [83] claiming that rare events play a crucial role and that their probability of occurrences are often underestimated to the point that they are not even considered by experts of the domain. It is therefore crucial to have reliable methods for computing small probabilities of failure.

## 1.3 Classical Methods for Probability Estimation

To estimate probabilities, they are two main approaches: numerical analysis and Monte Carlo simulation. In numerical analysis, the problem of computing the probability for a model to have a specific behavior may for instance be transformed into a system of linear equations. This system is then solved with classical linear algebra techniques. The main drawback of these methods is that the time and memory consumptions greatly increase when models become too complex, either because of the size of the state space which is known as the *state space explosion problem*, or because it features complex probability distributions. In Figure 1.1 experiments on a classical communication system composed of several queues in tandem are reported. This example is a well-known benchmark for rare event methods and is formally defined and studied in this manuscript. The state space of the system greatly increases with the number of queues. The time for estimating a probability of overflow before returning to an idle state in systems of variable sizes is plotted. On this figure, the time for numerical computation becomes intractable for large models. This situation is not specific to this example and very large system are intractable due to memory or time limitations.

An alternative to numerical computations is Monte Carlo simulation. In this setting, the system is simulated a large number of times and an estimation of the probability of interest is computed by counting the number of trajectories of the system that contains the behavior of interest. In the model-checking setting, this method is also known as statistical model checking. In Figure 1.1, a simulation-based method is used to estimate a probability close to 1 with a constant number of trajectories. This method takes almost constant time on the tandem queues system with respect to the size of the models. The simulation time mainly depends on the number of branchings in the system. It is constant in the tandem queue example and, most of the time, increases less than logarithmically with the system size. The simulation time also depends on the length of the trajectories which may not be directly linked to the size of the system. Moreover, simulation-based

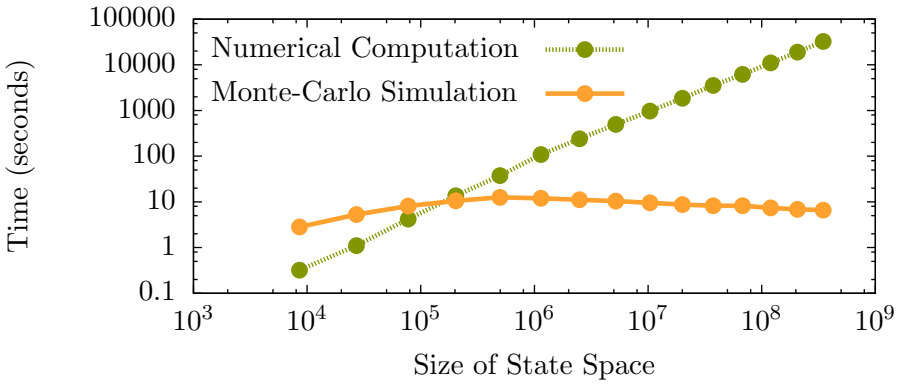



Figure 1.1: Numerical and simulation based computation time for increasing state space on a tandem queue example

methods can deal with models featuring complex probability distributions as long as algorithms to sample these distributions are available.

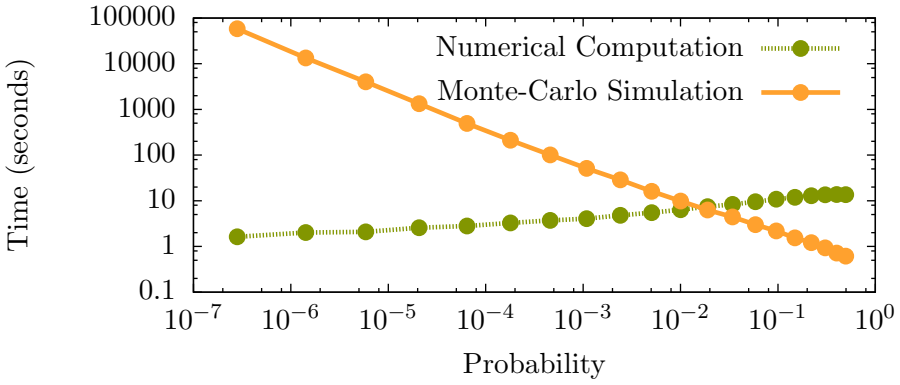



Figure 1.2: Numerical and simulation based computation time for decreasing probability on a tandem queue example

Unfortunately, simulation-based methods do not handle efficiently the estimation of rare-event probability. Figure 1.2 shows the time required for the estimation of a small probability such that the estimation error is a constant fraction of the probability of interest which can be formalized using confidence intervals. On this example, when the probability goes to zero, the simulation time becomes intractable whereas the time required by numerical method is almost constant.

The limitations of numerical methods and Monte-Carlo simulation have been heavily studied. In particular, several methods to deal with the rare-event problem have been developed, the two main ones being *splitting* and

*importance sampling*. These two methods modify how the simulation is conducted by making copies of trajectories in the case of splitting or biasing probability distributions in the case of importance sampling. They have been successfully applied in various domains like nuclear physic or queuing network. However two main limitations remain to apply them to model-checking problems: they require manual setting and they can only provide asymptotical guarantee on their result, in some particular cases they can even worsen the problem.

## 1.4 Contributions

This thesis presents contributions to simulation techniques in three directions:

- A theoretical method to use importance sampling which guarantees the result by providing a conservative confidence interval not relying on some asymptotical hypotheses has been developed [IV, V, VI]. This method relies on an abstraction of the system under study which is small enough to be analyzed numerically and which is used to build an efficient importance-sampling algorithm. It combines the advantages of both numerical and simulation-base methods for the estimation of probabilities of occurrence of rare events in large systems. This method can be used by manually defining an abstraction of the model. A theoretical framework in which the abstraction of the model is automatically computed is also presented.

- Software developments on the tool Cosmos have been conducted. Cosmos is a statistical model checker which takes Petri nets as input for models, and performs quantitative model checking using the HASL logic. The theoritical methods for handling rare events that we have developed have been integrated in this tool.

  The input language of the tool has been extended to a broader class of Petri nets, namely Stochastic Symmetric Nets [I].

  Recurrent difficulties with verification tools appear when one wants to exchange data between tools or interact with them in a uniform way. The CosyVerif platform [II] addresses these issues and provides a common interface to various tools including Cosmos.

- Several experiments using the rare-event methods have been conducted as well as one case study on a biological system [III] to assess the efficiency of the proposed methods.

## 1.5 Organization

In a first part, Chapter 2 recalls the formal notation and statistical results used throughout this thesis as well as classical algorithms used for numerical analysis of models. This chapter also recalls some properties of coupling relations. Chapter 3 presents in more details the rare-event problem for Monte Carlo simulations as well as classical methods to circumvent it, namely splitting and importance sampling. In the second part, theoretical results are presented. Chapter 4 presents a reliable theoretical method to estimate small probabilities for big systems. This method describes an algorithm based on an abstraction of the model. The design of this abstraction is a manual step. In Chapter 5, a framework is described in which abstractions can automatically be computed. The third part presents more applied results. Chapter 6 presents software contributions and benchmarks. In this chapter, the efficiency of the methods presented in Chapter 4 is demonstrated, as well as the efficiency of the statistical model checker Cosmos. Chapter 7 presents a biological case study where the methods for rare events previously described are used. Finally Chapter 8 contains the conclusions and perspectives of this thesis.

# Part I

# Preliminaries

# Chapter 2

# Technical Background

## 2.1 Introduction

This chapter presents known results used all along this thesis. They are presented in three parts:

- In a first part, we present the different stochastic models that we will consider later. We recall their semantics and some useful related properties.
- In the second part, we collect classical statistical results for estimating the expected value of a random variable. In particular, we introduce the notion of confidence interval and recall different methods to compute some.
- The last part is devoted to model-checking : we present logics to specify properties of stochastic systems and algorithms to verify them.

## 2.2 Models

### 2.2.1 Discrete Event Dynamic System

A general setting to describe probabilistic system is to suppose that the state of the system evolves in discrete steps, which implies that, in any time interval, the number of reached state is countable. This does not allows to model hybrid systems where the state of the system evolves continuously.

Discrete events system are modeled formally by defining successive states of a system as sequence of random variables. The time between successive events is also described by a sequence of random variables.

**Definition 1 (Discrete Event Dynamic System (DEDS))**

A discrete event dynamic system is a tuple: $(S, S_0, E, (E_n)_0^\infty, (T_n)_0^\infty)$ with:

- $S$ is a discrete set of *states*,
- $S_0$ is the random variable of the *initial state* taking value in $S$,
- $E$ is a set of *events*,
- $\delta : S \times E \to S$ is the *transition* function.
- $(E_n)_0^\infty$ is a sequence of random variables over the set of events $E$. A realization of $(E_n)_0^\infty$ is sequence of events in the system. The sequence of states is inductively defined as $S_{n+1} = \delta(S_n, E_n)$ from the initial state $S_0$.
- $(T_n)_0^\infty$ is a sequence of stopping times which are random variables taking values in $\mathbb{R}^+$ representing time interval. The variable $T_0$ is the time before the first event. Variable $T_n$ denotes the time between event $E_{n-1}$ and $E_n$.

**Running example.** *A tandem queues system is used in this section to illustrate the different model formalisms. This system is shown on Figure 2.1. It consists in two queues : event $\lambda$ represents the arrival of a client in the first queue. Once served in the first queue, the client goes to the second queue, represented by event $\rho_1$. Once served in the second queue, the client leaves the system with event $\rho_2$. The first event can occur at any time while the two others require the corresponding queue to contain at least one client.*

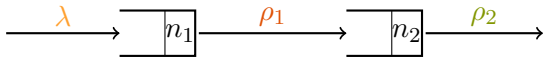



Figure 2.1: Tandem Queues System

*Each of the three events occurs according to a probability distribution. The first event is distributed according to a Poisson process and the two services $\rho_1$ and $\rho_2$ are distributed with exponential distributions. This system is described as **M/M/1** with two queues in Kendall's notation.*

*As the distribution are Markovian at any time the system is fully described by the number of clients in the two queues. The pair $(n_1, n_2)$ is thus used to describe the state of the system where there are $n_1$ clients in the first queue and $n_2$ clients in the second queue. As the queues are not bounded, the state space of the system is $\mathbb{N} \times \mathbb{N}$.*

*Figure 2.2 shows a realization of a DEDS specifying the tandem queues system. The initial state is* $(1, 0)$ *and the set of events is* $\{\lambda, \rho_1, \rho_2\}$*. Its formal definition is given in next section.*

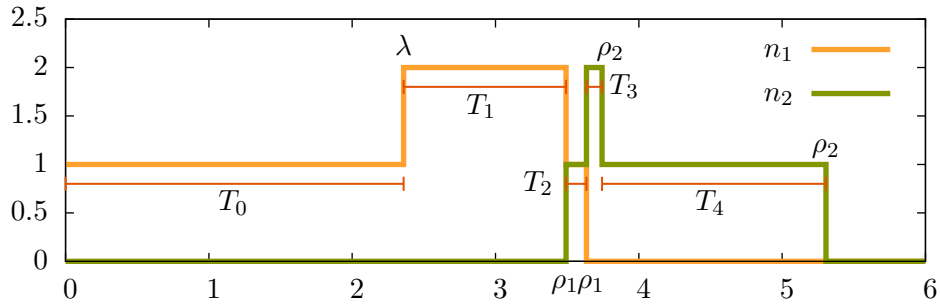



Figure 2.2: Example of a realization of a DEDS

**Remark 1** *A DEDS can be defined without the sequence of events but with the sequence of successive states* $(S_n)_0^\infty$*.*

A *Zeno trajectory* in DEDS is a realization of a DEDS for which $\sum_{i=0}^{\infty} T_i$ converges. Systems in which the probability of Zeno paths is not null are more difficult to analyzed and often avoided. The model of Markov chains which is heavily used in this thesis is not subject to the problem of Zeno path.

Several analysis can be performed on such a system. Stationary analysis studies whether the system reaches an equilibrium and in this case the behavior of the system in this equilibrium.

**Definition 2 (Stationary distribution)**

Let denoted by $\big(\pi(s)\big)_{s\in S}$ the probability distribution of states of the DEDS after an infinite time. It is defined when it exists as

$$\pi(s) = \lim_{n\to\infty} \mathbb{P}(S_n = s)$$

Transient analysis studies the behavior of a system in some fixed bounded time.

**Definition 3 (Transient distribution)**

Let $\big(\pi_\tau(s)\big)_{s\in S}$ be the distribution of state at time $\tau$. It exists as soon as the set of Zeno paths has null probability in the system. It is defined as:

$$\pi_\tau(s) = \sum_{n=0}^{\infty} \mathbb{P}\big(S_n = s \mid T_n \leq \tau < T_{n+1}\big)\mathbb{P}\big(T_n \leq \tau < T_{n+1}\big)$$

As the transient distribution depends of the initial state, $\pi_{\tau,s}$, with $s \in S$ is defined as the distribution at time $\tau$ of a DEDS where the initial state have been replaced by $s$.

### 2.2.2 Discrete Time Markov Chain

A discrete time Markov chain (DTMC) is a stochastic process fulfilling the discrete Markov property. The discrete Markov property ensures that the process is memoryless after each event, that is all future stochastic behaviors only depend of the current state.

**Definitions**

**Definition 4 (Discrete Markov Property for DEDS)**

Let $\mathcal{D} = (S, S_0, E, (E_n)_0^\infty, (T_n)_0^\infty)$ be a DEDS. The DEDS $\mathcal{D}$ has the discrete Markov property if for all $n \in \mathbb{N}$:

$$\forall s \in S,\ \mathbb{P}(S_{n+1} = s \,|\, S_0, T_0, S_1, T_1, \ldots, S_n, T_n) = \mathbb{P}(S_{n+1} = s \,|\, S_n)$$

$$\text{and } \forall t \in \mathbb{R}^+,\ \mathbb{P}(T_{n+1} > t \,|\, S_0, T_0, S_1, T_1, \ldots, T_n, S_{n+1}) = \mathbb{P}(T_{n+1} > t \,|\, S_{n+1})$$

In DTMC, the passing of time is represented only by the successive states of the system. The time between events is irrelevant and thus assumed to be always equal to one time unit. Consequently the state space of a DTMC is countable. Thus it can be represented as a possibly infinite directed graph where arcs are labeled by probabilities. This can be defined formally by the following:

**Definition 5 (Discrete Time Markov Chain (DTMC))**

A Discrete Time Markov Chain (DTMC) $\mathcal{C}$ is a tuple $(S, s_0, \mathbf{P})$ with:

- $S$ is a possibly infinite set of *states*,

- $s_0 \in S$ is the *initial state*,
- $\mathbf{P} = \big(\mathbf{P}(s, s')\big)_{(s,s') \in S \times S}$ is the *transition probability matrix*, This matrix $\mathbf{P}$ is stochastic which means that all its elements are non negative and the sum of each line is one, i.e. $\forall s \in S, \sum_{s' \in S} P(s, s') = 1$.

**Remark 2** *The function $\boldsymbol{P}$ is called a matrix despite the fact that it has infinite rows and columns when $S$ is infinite.*

DTMC takes as semantics a DEDS with the same state space, where the random variable $S_0$ takes for unique value $s_0$. The sequence of states is defined as follows:

$$\forall n > 0, \forall s \in S,\ \mathbb{P}(S_n = s) = \sum_{s' \in S} \mathbb{P}(S_{n-1} = s')\mathbf{P}(s', s).$$

The sequence of times $(T_n)$ is equal to 1 for all $n$. By definition DTMCs fulfill the Markov property.

The *embedded graph* of a Markov chain is defined as a directed graph whose vertices are the states of the Markov chain and the transitions of this graph are $s \to t$ where $\mathbf{P}(s, t) > 0$. Furthermore transitions of the graph may be labeled by their probabilities.

**Example 1** *Figure 2.3 shows a simple DTMC where the state space is the set $\{1, 2, 3\}$, the initial state is 3 and the transition probability matrix is*

$$P = \begin{pmatrix} 0.3 & 0.7 & 0 \\ 0 & 0 & 1 \\ 0.2 & 0.8 & 0 \end{pmatrix}.$$

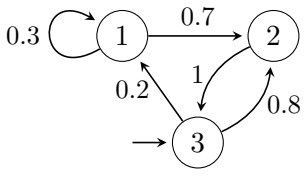



Figure 2.3: A simple DTMC

**Running example.** *The figure 2.4 represents a DTMC of a tandem queues system. The number of clients in the first queue is represented on the horizontal axis and the number of clients in the second one is represented on*

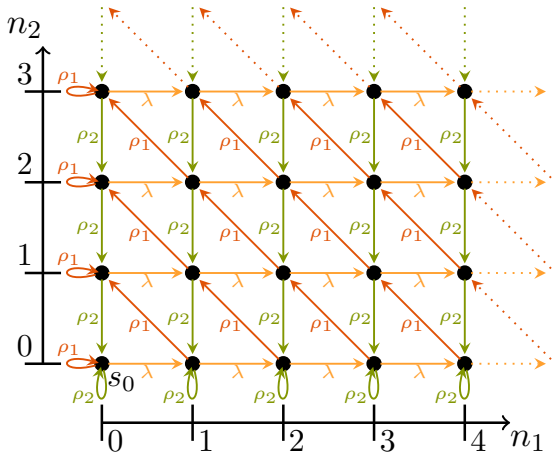



Figure 2.4: DTMC for the tandem queues

*the vertical axis. In the initial state $s_0$, the two queues are empty. Here this model is presented as a discrete one and thus the probability distribution of events is replaced by choices between events. Given a state $s = (n_1, n_2)$, a new client comes in the first queue with probability $\lambda$, a client leaves the first queue for the second one with probability $\rho_1$ and a client leaves the second queue and exits with probability $\rho_2$ ($\lambda + \rho_1 + \rho_2 = 1$).*

*The transition probability matrix is as follows:* $\forall s = (n_1, n_2) \in S$,

$$\begin{array}{ll} & \boldsymbol{P}\big((n_1,n_2),(n_1+1,n_2)\big) = \lambda \\ \textit{if } n_1 > 0 & \boldsymbol{P}\big((n_1,n_2),(n_1-1,n_2+1)\big) = \rho_1 \\ \textit{if } n_2 > 0 & \boldsymbol{P}\big((n_1,n_2),(n_1,n_2-1)\big) = \rho_2 \\ \textit{if } n_2 > 0 & \boldsymbol{P}\big((0,n_2),(0,n_2)\big) = \rho_1 \\ \textit{if } n_1 > 0 & \boldsymbol{P}\big((n_1,0),(n_1,0)\big) = \rho_2 \\ & \boldsymbol{P}\big((0,0),(0,0)\big) = \rho_1 + \rho_2 \\ \textit{otherwise} & \boldsymbol{P}\big((n_1,n_2),(n_1',n_2')\big) = 0 \end{array}$$

For our purpose, we enrich the definition of Markov chains with events. This definition should not be confused with the more classical definition of *labeled Markov chain*. In a labeled Markov chain, a labeling function assigns labels to states of the Markov chain. Here events are added on the transitions of the chain. It is possible to define *enriched labeled Markov chain* as the two extensions do not interfere one with the other.

**Definition 6**

An *enriched* discrete time Markov chain $\mathcal{C}$ is defined by:

- a set of states $S$ including an initial state $s_0$;

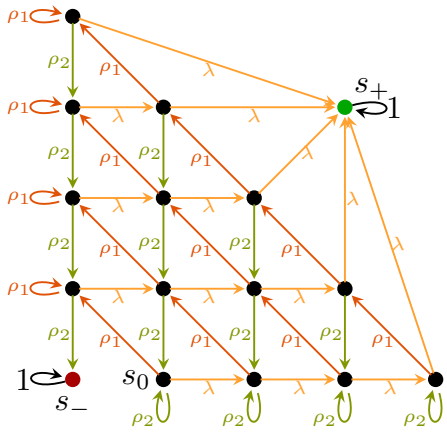



Figure 2.5: DTMC with absorbing states

- a finite set of events $E$;
- a successor function $\delta : S \times E \to S$;
- a function $p : S \times E \to [0, 1]$ such that for all $s \in S$, $\sum_{e \in E} p(s, e) = 1$.

We define the transition probability matrix $\mathbf{P}$ of size $S \times S$ by:

$$\forall s, s' \in S,\ \mathbf{P}(s, s') = \sum_{\delta(s,e)=s'} p(s, e)$$

**Running example.** *The tandem queues example has three events: $a$ is the arrival of a client, $e_1$ is the end of a service in queue 1 and $e_2$ is the end of a service in queue 2. For each state $s$, $p(s, a) = \lambda$, $p(s, e_1) = \rho_1$ and $p(s, e_2) = \rho_2$.*

In the context of simulation we will use bounded DTMC (BDTMC) as chain equipped with two special states. The terminology will be explained later.

**Definition 7 (BDTMC)**

A bounded DTMC is a tuple $(S, s_0, \mathbf{P}, s_-, s_+)$ where:

- $(S, s_0, \mathbf{P})$ is a DTMC.
- $s_- \in S$ and $s_+ \in S$ and these two states are absorbing.
  i.e. $\mathbf{P}(s_-, s_-) = \mathbf{P}(s_+, s_+) = 1$.

- The only bottom strongly connected components are $\{s_-\}$ and $\{s_+\}$.

This definition is equivalent to say that states $s_-$ and $s_+$ are reached with probability one and are absorbing. This definition is independent from the one of enriched DTMC and thus it makes sense to consider enriched bounded DTMC. The figure 2.5 shows a BDTMC. A BDTMCis *ergodic* as the only recurrent states are $s_+$ and $s_-$ and there exists $p \in [0,1]$, the probability to reach $s_+$ such that $\pi(s_+) = p$, $\pi(s_-) = 1 - p$.

### Methods

**Qualitative analysis** Qualitative analysis of a DTMC is done by analyzing its embedded graph. From the embedded graph of any DTMC, the *tree decomposition in strongly connected components* can be computed. A Strongly Connected Component (SCC) is a subset of the graph for which their exists a path between all pairs of states and maximal for this property. The tree decomposition in strongly connected components is computed by taking as root the SCC containing the initial state. Then inductively a SCC $A$ is the parent of a SCC $B$ if there exists a path from a state of $A$ to a state of $B$. SCC which are not connected to the root does not appear in the tree decomposition. SCC which are leaves of the tree are called *Bottom Strongly Connected Components (BSCC)*.

BSCC play a crucial role in qualitative analysis. For any state $s$ in a BSCC, the probability that $s$ is visited infinitely often by a trajectory of the Markov chain is strictly greater than zero whereas for any state which is not in a BSCC this probability is equal to zero.

A state $s$ of a SCC is called *aperiodic* if the gcd of lengths of all cycles on $s$ is 1. Finite Markov chains are *ergodic* if all BSCC are aperiodic. A SCC is aperodic if it contains an aperiodic state.

**Computing reachability probability** Let $\pi_s$ denote the stationary distribution of a DTMC where the initial state has been replaced by $s$. The stationary distribution of a DTMC $\mathcal{C}$ exists and is unique as soon as $\mathcal{C}$ is ergodic. In the particular case of a BDTMC, $\pi_s(s_+)$ is the probability to reach $s_+$ from a state $s$ and is denoted by $\mu(s)$. This stationary distribution can additionally be defined by the following equation:

**Definition 8 (Reachability)**

Given a DTMC $\mathcal{C}$ and a set of states $S_+ \subset S$, we define the vector of probabilities $\mu \in [0,1]^S$ as the smallest solution of the following system of

equations:

$$\mu(s) = \begin{cases} 1 & \text{if } s \in S_+ \\ \sum_{s' \in S} \mathbf{P}(s, s')\mu(s') & \text{if } s \notin S_+ \end{cases}$$

This definition allows to computes effectively the value of the stationary distribution $\mu$ by solving a linear system of equations.

**Computing time-bounded reachability probability** For any delay $u \in \mathbb{N}$, we define $\mu_u(s)$ as the probability to reach the state $s_+$ in $u$ time units starting from state $s$, that is defined as

$$\mu_u(s) = \mathbb{P}(S_u = s_+)$$

with $S_0 = s$. The vector $\mu$ whose components are $\mu_u(s)$, for $s \in S$ is called the time-bounded reachability vector with time-bound $u$. Similarly to $\mu$ a more effective definition of $\mu_u$ can be stated:

**Definition 9 (Time-Bounded Reachability for DTMC)**

Given a DTMC $\mathcal{C}$, a set of states $S_+ \subset S$ and a positive integer $u$ we define the vector of probabilities $\mu_u \in [0,1]^S$ as:

$$\mu_u(s) = \begin{cases} 1 & \text{if } s \in S_+ \\ 0 & \text{if } u = 0 \wedge s \notin S_+ \\ \sum_{s' \in S} \mathbf{P}(s, s')\mu_{u-1}(s') & \text{if } u > 0 \wedge s \notin S_+ \end{cases}$$

This definition can be used to compute effectively time-bounded reachability using matrix-vector multiplications.

**Coupling Methods** The *coupling* method [68] is a classical method for comparing two stochastic processes. It can be applied in various contexts (establishing ergodicity of a chain, stochastic ordering, bounds, etc.). In this thesis coupling methods are used to establish an order on the set of states of a DTMC toward the reachability of a state $s_+$. A coupling between two Markov chains is also a Markov chain whose state space is a subset of the product of the two spaces. This subset is called *the coupling relation*. A coupling must satisfy that that its projection on any of its components behaves like the original corresponding chain. This means that the product chain contains the behaviors of its two components.

Depending of the context for which the coupling is used, additional constraints are imposed. For our needs, we only define the coupling of a chain with itself and we add a constraint on the set of states to characterize an order on the DTMC states. Higher states in this order have higher probability to reach a state $s_+$.

**Definition 10**

Let $\mathcal{C} = (S, \mathbf{P})$ be a Markov chain and $s_+ \in S$ be an absorbing state of $\mathcal{C}$. A coupling of $\mathcal{C}$ with itself is a DTMC $\mathcal{C}^{\otimes} = (S^{\otimes}, \mathbf{P}^{\otimes})$ such that :

- $S^{\otimes} \subseteq S \times S$
- $\forall s \neq t \in S\ ,\ \forall (s, s') \in S^{\otimes},\ \mathbf{P}\ (s\ , t) = \sum_{t' \in S} \mathbf{P}^{\otimes}((s, s'), (t, t'))$ and $\forall s' \neq t' \in S,\ \forall (s, s') \in S^{\otimes},\ \mathbf{P}(s', t') = \sum_{s' \in S}\ \mathbf{P}^{\otimes}((s, s'), (t, t'))$
- $\forall (s, t) \in S^{\otimes},\ s = s_+ \Rightarrow t = s_+$

The set $S^{\otimes}$ defines a coupling relation with a reachability goal $s_+$.

The following proposition allows to compare transient and stationary reachability probabilities without any numerical computation. Recall that $\mu(s)$ denotes the probability to reach the state $s_+$ in $\mathcal{C}$ starting from state $s$.

**Theorem 1**

Let $\mathcal{C}^{\otimes}$ be a coupling of $\mathcal{C}$, with a reachability goal $s_+$ Then, for all $(s, s') \in S^{\otimes}$, we have:

$$\mu(s) \leq \mu(s')$$

$$\forall u > 0,\ \mu_u(s) \leq \mu_u(s')$$

*Proof:*
Let $\sigma$ be a finite random trajectory ending in a bottom strongly connected component in the coupled chain starting from $(s, s')$. We define the random variables $\mathbf{1}_{s_+}, \mathbf{1}'_{s_+}$ by:

- $\mathbf{1}_{s_+} = 1$ if the first component of the ending state of $\sigma$ is $s_+$.
- $\mathbf{1}'_{s_+} = 1$ if the second component of the ending state of $\sigma$ is $s_+$.

Let $(s_f, s'_f) \in S^{\otimes}$ be the final state of $\sigma$.
Using that $\forall (s, t) \in S^{\otimes},\ s = s_+ \Rightarrow t = s_+$ we have

$$s_f = s_+ \Rightarrow s'_f = s_+$$

Then for all $\sigma$:

$$\mathbf{1}_{s_+}(\sigma) \leq \mathbf{1}'_{S_+}(\sigma)$$

By taking expected values, we have:

$$\mathbb{E}(\mathbf{1}_{s_+}) \leq \mathbb{E}(\mathbf{1}'_{s_+})$$

Which can be rewritten as: $\mu(s) \leq \mu(s')$.
By taking conditional expectation over the length of the path we obtain:

$$\forall u > 0,\ \mathbb{E}(\mathbf{1}_{s_+} \big|\ |\sigma| \leq u) \leq \mathbb{E}(\mathbf{1}'_{s_+} \big|\ |\sigma| \leq u)$$

This can be rewritten as: $\forall u > 0,\ \mu_u(s) \leq \mu_u(s')$.

□

The following proposition specifies a special kind of coupling adapted to our needs. We point a local property, at the level of transitions, which whenever satisfied makes a relation a coupling. Precisely: For any pair of states $(s, s')$ in the relation $S^{\otimes}$, for any events $e \in E$ we require that the pair $(\delta(s_1, e), \delta(s_2, e))$ is also in the relation, in other words, we require that the following diagrams holds.

$$\forall (s, s') \in S, \forall e \in E: \qquad \begin{array}{ccc} s & \xrightarrow{\ e\ } & t \\ {\scriptstyle S^{\otimes}}\big| & & \big|{\scriptstyle S^{\otimes}} \\ s' & \xrightarrow{\ e\ } & t' \end{array}$$

When this holds the relation $S^{\otimes}$ is a coupling relation.

**Theorem 2**

Let $\mathcal{C} = (S, E, \delta, p)$ be an enriched BDTMC with a reachability goal state $s_+$. Let $S^{\otimes} \subset S \times S$ be a relation on $\mathcal{C}$ such that:

1. $\forall e \in E, s_1, s_2 \in S^{\otimes},\ p(s_1, e) = p(s_2, e)$
2. $\forall (s_1, s_2) \in S^{\otimes}, e \in E, (t_1, t_2) \in S^2$
   $(\delta(s_1, e) = t_1 \wedge \delta(s_2, e) = t_2) \Rightarrow (t_1, t_2) \in S^{\otimes}$

We define $\delta^{\otimes}((s_1, s_2), e) = (\delta(s_1, e), \delta(s_2, e))$ and $p^{\otimes}((s_1, s_2), e) = p(s_1, e)$. Then the DTMC $\mathcal{C}^{\otimes} = (S^{\otimes}, E, \delta^{\otimes}, p^{\otimes})$ and $S^{\otimes}$ is a coupling relation on $\mathcal{C}^2$.

*Proof:*
The second hypothesis asserts that the chain is well defined. Furthermore, for $s, t$ in $S$ such that $s \neq t$, we have, using Definition 6:

$$\forall s' \in S,\ (s, s') \in S^{\otimes} \Rightarrow \sum_{t' \in S} \mathbf{P}^{\otimes}((s, s'), (t, t')) = \sum_{e | \delta(s,e) = s'} p(s, e) = \mathbf{P}\ (s\ , t).$$

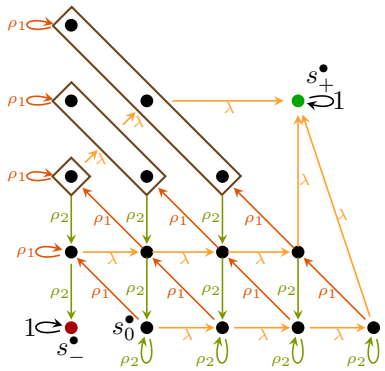



Figure 2.6: Reduced DTMC

The other claim of Definition 6 has a similar proof.

□

**Running example.** *Let us illustrate a coupling for the Markov chain represented in figure 2.6 and called* $\mathcal{C}^\bullet$*. This chain is obtained from the tandem queues example presented in figure 2.5 by lumping together states which have the same number of clients and at least* $R$ *clients in the second queue (in the figure* $R = 2$*). Its set of states* $S^\bullet$ *is obtained from*

$$\{(n_1, n_2) \mid 0 \leq n_1 + n_2 \leq N \wedge n_2 \leq R\}$$

*by merging all states* $(n_1, n_2)$ *such that* $n_1 + n_2 = N$*.* $s_+$ *corresponds to this merging and any pair* $(n_1, n_2)$ *such that* $n_1 + n_2 = N$ *denotes* $s_+$*. The goal set contains only* $s_+$*.* $s_- = (0, 0)$*. Let us define*

$$S^\otimes = \{((n_1, n_2), (n_1', n_2')) \mid n_1 + n_2 \leq n_1' + n_2' \wedge n_1 \leq n_1'\}$$

*and prove that* $S^\otimes$ *is a coupling relation. The first and the third conditions of proposition 2 are satisfied by construction. Let us check that for all couples* $((n_1, n_2), (n_1', n_2'))$ *in the relation, all successors are also in the relation. We examine the three events of the system:*

1. *For event* $a$*, the successor of* $((n_1, n_2), (n_1', n_2'))$ *is* $((n_1 + 1, n_2), (n_1' + 1, n_2'))$ *which is inside the relation.*

2. *For event* $e_1$*, the successor of* $((n_1, n_2), (n_1', n_2'))$ *is*
   $((n_1 - \mathbf{1}_{\{n_1>0 \wedge n_2<R\}}, n_2 + \mathbf{1}_{\{n_1>0 \wedge n_2<R\}}),$
   $(n_1' - \mathbf{1}_{\{n_1'>0 \wedge n_2'<R\}}, n_2' + \mathbf{1}_{\{n_1'>0 \wedge n_2'<R\}}))$
   *the first condition defining* $S^\otimes$ *is satisfied as the sums are unchanged.*

   - *If* $n_1 \leq n_1'$ *then* $n_1 - \mathbf{1}_{\{n_1>0 \wedge n_2<R\}} \leq n_1' - \mathbf{1}_{\{n_1'>0 \wedge n_2'<R\}}$ *and the second condition defining* $S^\otimes$ *is satisfied.*

- *Else* $n_1 = n'_1$ *and using the first condition we have* $n_2 \leq n'_2$*; then* $n'_1 > 0 \wedge n'_2 < R \Rightarrow n_1 > 0 \wedge n_2 < R$ *which implies:* $n_1 - \mathbf{1}_{\{n_1>0 \wedge n_2<R\}} \leq n'_1 - \mathbf{1}_{\{n'_1>0 \wedge n'_2<R\}}$*, so the second condition holds.*

3. *For event* $e_2$*, the successor of* $((n_1, n_2), (n'_1, n'_2))$ *is* $((n_1, n_2 - \mathbf{1}_{\{n_2>0\}}), (n'_1, n'_2 - \mathbf{1}_{\{n'_2>0\}}))$.
   *As the first component is unchanged, the second condition holds.*

   - *If* $n_1 + n_2 < n'_1 + n'_2$ *then* $n_1 + n_2 - \mathbf{1}_{\{n_2>0\}} \leq n'_1 + n'_2 - \mathbf{1}_{\{n'_2>0\}}$
   - *Else* $n_1 + n_2 = n'_1 + n'_2$ *and using the second condition* $n_2 \leq n'_2$ *then* $n_2 - \mathbf{1}_{\{n_2>0\}} \leq n'_2 - \mathbf{1}_{\{n'_2>0\}}$*, so the first condition holds.*

*Then* $S^{\otimes}$ *is a coupling relation.*

### 2.2.3 Continuous Time Markov Chain

#### Definitions

When the time between events of a Markov chain is not discrete, continuous distribution of the time are used and the continuous Markov property should hold between each event.

**Definition 11 (Continuous Markov Property for DEDS)**

Let $\mathcal{D} = (S, S_0, E, (E_n)_0^\infty, (T_n)_0^\infty)$ be a DEDS. The DEDS $\mathcal{D}$ has the continuous Markov property if it has the discrete Markov property and for all $n \in \mathbb{N}$:

$$\forall t, t' \in \mathbb{R}^+,\ \mathbb{P}(T_n > t + t' \,|\, T_n > t') = \mathbb{P}(T_n > t)$$

The discrete Markov property ensures that at each event the system is memoryless. The continuous Markov property ensures furthermore that between two events, the waiting time is also memoryless.

The only continuous distribution satisfying the continuous Markov property is the negative exponential distribution. Thus continuous time Markov chains are defined by adding a *rate* which is a positive real to each state. The waiting time in each state is distributed exponentially with this rate.

**Definition 12 (CTMC)**

A Continuous Times Markov Chain is a tuple: $(S, s_0, \mathbf{P})$ with:

- $S$ is a set of *states*,

- $s_0 \in S$ is the *initial state*,
- $\mathbf{Q} = (\mathbf{Q}(s, s'))_{(s,s') \in S \times S}$ is the *infinitesimal generator matrix*, all elements of this matrix are non negative except for the diagonal ones, which are negative. Moreover the sum of coefficients for each line is equal to zero, i.e. $\forall s \in S, \sum_{s' \in S} \mathbf{Q}(s, s') = 0 \Leftrightarrow \mathbf{Q}(s, s) = -\sum_{s' \neq s \in S} \mathbf{Q}(s, s')$.

Other (equivalent) definitions of Continuous time Markov chain exist. The infinitesimal generator matrix $\mathbf{Q}$ can be replaced by the *transition rate matrix* $\mathbf{R}$. For the same CTMC matrices $\mathbf{Q}$ and $\mathbf{R}$ are equal except on the diagonal where $\mathbf{R}$ can take arbitrary non negative values corresponding to loops. The infinitesimal generator can be split in two, a probability transition matrix $\mathbf{P}$ and an *exit rate function* $\mathbf{E} : S \to \mathbb{R}_{\geq 0}$. For two states $s, s' \in S$, the transition rate matrix $\mathbf{R}(s, s')$ is equal to $\mathbf{E}(s)\mathbf{P}(s, s')$

These three representations describe the same stochastic process but with different points of view. The infinitesimal generator is used in the *Kolmogorov* equation describing the behaviors of the chain. The rate transition matrix $\mathbf{R}$ is more used for modeling as the loops are treated like other transitions. Finally splitting the stochastic behavior in waiting time and probability transition matrix reveals the discrete behaviors of the system.

A CTMC takes its semantics as a DEDS with the same state space $S$, a unique initial state $s_0$,

$$\forall n > 0, \forall s \in S,\ \mathbb{P}(S_n = s) = \sum_{s' \in S \setminus \{s\}} \mathbf{Q}(s', s)\mathbb{P}(S_{n-1} = s'),$$

$$\forall n > 0,\ \mathbb{P}(T_n \leq x) = 1 - e^{\mathbf{Q}(S_n, S_n)x}.$$

Similarly to DTMCs, enriched CTMCs can be defined:

**Definition 13**

An *enriched* continuous time Markov chain $\mathcal{C}$ is a tuple $(S, s_0, E, \delta, \lambda)$ defined by:

- a set of states $S$ including an initial state $s_0$;
- a finite set of events $E$;
- a successor function $\delta : S \times E \to S$;
- a function $\lambda : S \times E \to \mathbb{R}^+$

We define the infinitesimal generator matrix $\mathbf{Q}$ of size $S \times S$ by:

$$\forall s \neq s' \in S,\ \mathbf{Q}(s, s') = \sum_{\delta(s,e)=s'} \lambda(s, e)$$

$$\forall s \in S,\ \mathbf{Q}(s, s) = - \sum_{\delta(s,e)\neq s} \lambda(s, e)$$

**Methods**

**Computing reachability probability** The embedded DTMC of a CTMC is defined as: $\mathcal{D} = (S, E, \delta, p)$ where $p$ is defined as follow:

$$\forall s \in S,\ e \in E,\ p(s, e) = \frac{\lambda(s, e)}{\sum_{s' \in S} \lambda(s', e)}.$$

The distribution of time $(T_n)$ in the DEDS is replaced by a sequence always equal to 1.

As reachability probabilities do not depend of the waiting time, reachability probabilities in a CTMC are equal to reachability probabilities in its embedded DTMC.

**Computing time-bounded reachability probability** Let $\mu_\tau$ the vector of probabilities to reach $s_+$ in $\tau$ time units be defined as $\mu_\tau(s) = \pi_{\tau,s}(s_+)$.

Using that the evolution of time is distributed with negative exponential distribution, the following definition for $\mu_\tau$ holds:

**Definition 14 (Bounded Reachability for CTMC)**

Given a CTMC $\mathcal{C}$, a set of states $S_+ \subset S$ and a positive real $\tau$ we define the vector of probabilities $\mu_\tau \in [0, 1]^S$ as the smallest solution to the following system of equations:

$$\mu_\tau(s) = \begin{cases} 1 & \text{if } s \in S_+ \\ \displaystyle\int_0^\tau \sum_{s' \in S, s' \neq s} \mathbf{Q}(s, s') e^{\mathbf{Q}(s,s)u} \mu_{\tau-u}(s') du & \text{if } s \notin S_+ \end{cases}$$

While in a CTMC the differential system can be numerically solved, a more efficient way to obtain these probabilities consists in *uniformising* the chain.

A chain is said to be *uniform* when for each state $s$, the exit rate $\lambda = \lambda_s$ is independent from $s$. When a chain is uniform the following holds.

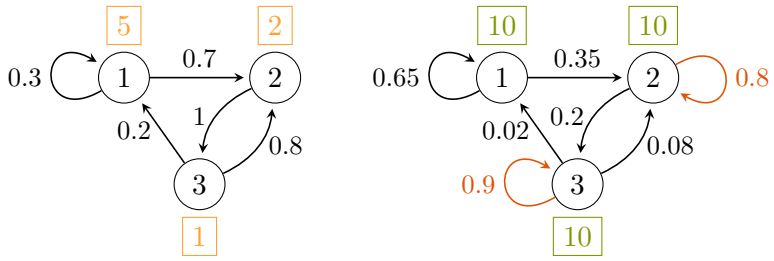



Figure 2.7: Two equivalent CTMCs: a non uniform one and its uniformisation

**Theorem 3 (Transient Distributions of Uniform Markov Chains)**

Given a uniform chain, the transient distribution $\pi_\tau$ is obtained by the following formula:

$$\pi_\tau(s) = \sum_{n \geq 0} \frac{e^{-\lambda\tau}(\lambda\tau)^n}{n!} \mathbf{P}^n(s_0, s)$$

Indeed using the uniform hypothesis, $\frac{e^{-\lambda\tau}(\lambda\tau)^n}{n!}$ is the probability that $n$ transitions take place in interval $[0, \tau]$ and $\mathbf{P}^n(s_0, s)$ is the probability to be in state $s$ after $n$ transitions. The term $\frac{e^{-\lambda\tau}(\lambda\tau)^n}{n!}$ is known as Poisson probability.

Given a non uniform chain with bounded rates, it is possible to transform it in a uniform chain with the same distribution $\pi_\tau$ (see figure 2.7). It consists in selecting some upper bound of the outgoing rates (say $\lambda$), consider $\lambda$ as the uniform transition rate and set a transition matrix $\mathbf{P}^{(u)}$ defined by:

$$\forall s \neq s' \in S \ \mathbf{P}^{(u)}(s, s') = \frac{\lambda_s}{\lambda} \mathbf{P}^{(u)}(s, s')$$

$$\mathbf{P}^{(u)}(s, s) = 1 - \sum_{s' \neq s} \mathbf{P}^{(u)}(s, s')$$

**Running example.** *Observe that uniformization does not require the CTMC to be finite. For instance, the rates of the infinite CTMC corresponding to the queuing tandem are bounded by $\rho_0+\rho_1+\rho_2$. Assuming that $\rho_0+\rho_1+\rho_2 = 1$, Figure 2.4 also represents the embedded DTMC of the uniform version of the tandem (with uniform rate 1).*

With the same argument, when the chain is uniform with rate $\lambda$, $\mu_\tau(s)$ fulfills:

$$\mu_\tau(s) = \sum_{n \geq 0} \frac{e^{-\lambda\tau}(\lambda\tau)^n}{n!} \mu_n(s)$$

This expression for $\mu_\tau$ is useful when it come to numerically compute the value of $\mu_\tau$ as [51] present efficient algorithms for truncating the infinite sum and computing Poisson probabilities with exact framing of the numerical errors.

**Coupling methods** For continuous time Markov chain, we can strengthen the result on coupling:

**Theorem 4**

Let $\mathcal{C} = (S, E, \delta, \lambda)$ be an enriched CTMC with absorbing state $s_+$. Let $S^\otimes \subset S \times S$ be a relation on $\mathcal{C}$ such that:

1. $\forall e \in E, s, s' \in S^\otimes$,
$$\lambda(s,e) > \lambda(s',e) \Rightarrow (\delta(s,e), s') \in S^\otimes$$
$$\text{and} \quad \lambda(s,e) < \lambda(s',e) \Rightarrow (s, \delta(s',e)) \in S^\otimes$$
2. $\forall (s,s') \in S^\otimes, e \in E, (t,t') \in S^2$
$(\delta(s,e) = t \wedge \delta(s',e) = t') \Rightarrow (t,t') \in S^\otimes$
3. $S^\otimes \cap (S_f \times S) \subset (S_f \times S_f)$

We define the product CTMC, $\mathcal{C}^\otimes = (S^\otimes, E^\otimes, \delta^\otimes, \lambda^\otimes)$ as follow:

- $E^\otimes = E \uplus \tilde{E}$, where $\tilde{E}$ is a copy of $E$.
- $\forall (s,s') \in S^\otimes$, $\delta^\otimes((s,s'),e) = (\delta(s,e), \delta(s',e))$
and $\delta^\otimes((s,s'),\tilde{e}) = \begin{cases} (\delta(s,e), s') & \text{if } (\delta(s,e), s') \in S^\otimes \\ (s, \delta(s',e)) & \text{if } (s, \delta(s',e)) \in S^\otimes \\ (s,s') & \text{otherwise} \end{cases}$
- $\forall (s,s') \in S^\otimes$, $\lambda^\otimes((s,s'),e) = min(\lambda(s,e), \lambda(s',e))$
and $\lambda^\otimes((s,s'),\tilde{e}) = |\lambda(s,e) - \lambda(s',e)|$

Then the CTMC $\mathcal{C}^\otimes$ is a coupling over $\mathcal{C}^2$ and

$$\forall \tau \in \mathbb{R}^+, \ \mu_\tau(s) \leq \mu_\tau(s')$$

*Proof:*
The second hypothesis asserts that the chain is well defined. Furthermore, for all $s, t$ in $S$ such that $s \neq t$, and for all $s' \in S$, such that $(s,s') \in S^\otimes$ we have, using definition 6:

$$\sum_{t' \in S} \mathbf{Q}^\otimes((s,s'),(t,t')) = \sum_{e \in E^\otimes | \delta^\otimes((s,s'),e) = (t,t')} \lambda^\otimes((s,s'),e)$$

We can split the sum over transitions of $E$ and transitions of $\tilde{E}$.

$$= \sum_{e \in E | \delta(s,e) = t} \lambda^\otimes((s,s'),e) + \sum_{\tilde{e} \in \tilde{E} | \delta(s,e) = t \wedge (\delta(s,e), s') \in S^\otimes} \lambda^\otimes((s,s'),\tilde{e})$$

Rates of transitions of $E$ can be replaced by theirs definitions.

Rates of transitions of $\tilde{E}$ are null if not in $(\delta(s,e),s') \in S^{\otimes}$

$$= \sum_{e\in E|\delta(s,e)=t} min(\lambda(s,e),\lambda(s',e)) + \sum_{\tilde{e}\in\tilde{E}|\delta(s,e)=t} max(\lambda(s,e)-\lambda(s',e),0)$$

The two sum can be merged together.

$$= \sum_{e\in E|\delta(s,e)=t} min(\lambda(s,e),\lambda(s',e)) + max(\lambda(s,e)-\lambda(s',e),0)$$

Finally we obtain.

$$= \sum_{e\in E|\delta(s,e)=t} \lambda(s,e) = \mathbf{Q}\ (s\ ,t)$$

The other claim for the coupling is symmetric.

The proof that

$$\forall\tau\in\mathbb{R}^{+},\ \mu_{\tau}(s) \leq \mu_{\tau}(s')$$

is obtain by rewriting the proof of Theorem 1 with conditional expectation on the time duration of paths.

□

### 2.2.4 Higher Level Model

Markov Chains are well suited for the theoretical analysis of probabilistic systems but not so well suited for modeling systems. To build a Markov chain modeling a system, one has to enumerate all states of the system and build the transition probability matrix. To ease the modeler's work, higher level models are used, which allow to express in a compact way complex systems. From such a compact representation, a Markov chain can be extracted and give a stochastic semantics to the system. In this thesis, we use stochastic Petri nets and stochastic symmetric nets as higher level models.

#### Stochastic Petri Net

Stochastic Petri net is a well known formalism for modeling some stochastic process. It relies on the usual formalism of Petri net, well adapted to the modeling of the control flow of concurent system. Let us first recall this formalism that have been heavily studied (see for example [26])

**Definition 15 (Petri net)**

A Petri net or Place/Transition net is a tuple $\mathcal{N} = (P, T, W^{-}, W^{+}, m_0)$ with $P \cap T = \emptyset$ where

- $P$ is a finite set of places,
- $T$ is a finite set of transitions,

- $W^- : P \times T \to \mathbb{N}$ is the *pre incidence matrix*,
- $W^+ : P \times T \to \mathbb{N}$ is the *post incidence matrix*,
- $m_0 \in \mathbb{N}^P$ is the initial marking.

We call a *marking* $m \in \mathbb{N}^P$ of a Petri net $\mathcal{N}$ a vector assigning an integer to each place of the net. A transition $t \in T$ is fireable in a marking $m$ if $\forall p \in P,\ m(p) - W^-(p,t) \geq 0$. The firing of a firable transition $t$ from a marking $m$ leads to marking $m'$ denoted $m \xrightarrow{t} m'$, where $m'$ is defined as $\forall p \in P,\ m'(p) = m(p) - W^-(p,t) + W^+(p,t)$. Let $\sigma = \sigma_1 \cdots \sigma_n \in T^*$ be a sequence of transitions, $\sigma$ is fireable from a marking $m$ leading to $m'$ and denoted by $m \xrightarrow{\sigma} m'$ if there exists a sequence of markings $m = m_1 \cdots m_{n+1} = m'$ such that for all $1 \leq k \leq n,\ m_k \xrightarrow{\sigma_k} m_{k+1}$. We denote by $Reach(\mathcal{N}, m_0)$ the set of reachable marking equals to $\{m \mid \exists \sigma \in T^* \ s.t.\ m_0 \xrightarrow{\sigma} m\}$. This set is in general infinite.

**Example 2** *Figure 2.8 represents a small Petri net where an action is split in two parts that evolve independently before synchronizing. The set of places is* $\{1,2,3,4\}$ *and the set of transitions* $\{a,b,c,d\}$*. The initial marking is* $(0,0,0,0)$*. The incidence matrices are:*

$$W^- = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \quad W^+ = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

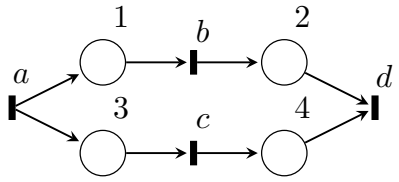



Figure 2.8: A simple Petri net

Stochastic behaviors are introduce in Petri net by adding notion of time to Petri net and by specifying passing of time by probability distribution. Let us first recall the different way time can be introduce in the behaviors of Petri net (see for example [13] for a more detailed comparison of Petri nets equipped with time) Two alternative approaches exist:

1. *Time Petri Net* [14] add clocks and guards on transitions of the net.

2. *Timed Petri Net* [1] add clocks to tokens and guards on input arcs of the net.

The first approach is more suitable to model a stochastic system, as it makes a correspondence between transitions of the net and transitions in the reachability graph. Thus stochastic Petri nets have been defined with probability distributions on the transitions.

Several choices remains on how the time is taken into account in transitions, when dealing with time Petri net all the choices are non deterministic:

- In *Duration semantics*, a transition is chosen among all the fireable ones, the tokens in the input places are removed and a waiting time is chosen with respect to the guard of the transition. When this time has elapsed the token are released in the output places. This semantic is not commonly used because marking reached by the net before the end of a transition may not belongs to the set of reachable marking of the plain Petri net.

- In *Delay semantics*, as soon as a transition is fireable a time is chosen and the transition is said to be *activated*. The transition is fired only when the sampled time is reached. This semantics preserves concurrence as several concurrent transitions can be activated at the same time, for each of them a time is chosen and only the transition with the smallest time is fired.

The second semantics is the most used one. Even with this semantics choices remains which are called policy.

- *Memory policy* determines the behaviors of transitions when a transition is activated, deactivated by concurrent transitions and activated again. Using *no memory policy* a time is chosen each time the transition is activated. Using *age memory policy* when a transition is deactivated the remaining time before firing is stored, when the transition is activated again, the firing time is set according to the remaining time before deactivation.

- *Server policy* determines if a transition for which there are enough tokens to fire it several times can effectively be activated several time. In *single server policy* such a behavior is forbidden. In *infinite server policy* the transition is activated as many times that there are enough tokens in the input places. In *multiple server policy* the maximal number of activations is fixed and specified for each transition.

In Stochastic Petri nets all choice of time are replaced by exponential probabilistic distribution and there is no guard on transitions. Moreover single server policy is used.

**Definition 16 (Stochastic Petri Net(SPN))**

A Stochastic Petri net is a tuple $\mathcal{N} = (P, T, W^-, W^+, m_0, \Lambda)$ where $(P, T, W^-, W^+, m_0)$ is a Petri net and $\Lambda : \mathbb{N}^P \times T \to \mathbb{R}$ is the rate function which associates a rate to each marking and transition.

Due to the choice of policy, stochastic Petri nets take naturally as semantics an enriched CTMC: From $\mathcal{N} = (P, T, W^-, W^+, m_0, \Lambda)$ a CTMC $\mathcal{C} = (S, s_0, E, \delta, \lambda)$ is defined as follow:

- $S = Reach(\mathcal{N}, m_0)$
- $s_0 = m_0$
- $E = T$
- $\delta(m, e) = m' \; s.t. \; m \xrightarrow{e} m'$
- $\lambda(m, e) = \Lambda(s, e)$

**Running example.** *Figure 2.9 shows a SPN representing the tandem queues system. Two places $(Q_1, Q_2)$ are used to encode the two queues and transitions encode events.*

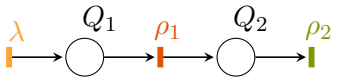



Figure 2.9: A SPN representing the tandem queues system

A common extension of SPN are generalized stochastic Petri nets which authorized to use *immediate transition* which follow Dirac distributions with parameter 0. Weight and/or priority are added to resolve concurrent firing of immediate transitions. As immediate transitions are memoryless the semantic of generalized Petri net is still a CTMC. Adding such immediate transitions is really convenient for modeling stochastic systems and may reduce the size of the reachable marking [57]. In this setting Zeno paths are possible if there is a cycle of immediate transitions. Such cycles can be detected by analyzing the structure of the net.

### Stochastic Symmetric Net

When a model contains several instantiations of a same pattern, it can be expressed in a compact way using Stochastic Symmetric Net (SSN). Intuitively instead of using indistinguishable tokens as in a Petri net, sets of *colors* are used to distinguish them. When the obtained system is unchanged when color classes are permuted, the lumped embedded Markov chain is

much smaller and can be computed directly [19]. Whether color classes can be permitted or not can be obtained by syntactic analysis of the system.

As the stochastic symmetric net formalism is an extension of the symmetric net formalism, we first recall this formalism. Symmetric net where also named *well formed net*. Compare to plain Petri nets which model the control flow of concurrent systems, symmetric nets can be used to model also data structure by distinguishing between tokens. In Petri net firing a transition is analogous to calling a function with no argument, in symmetric net a transition has a set of variables which are instantiated to some color. When the transition is fired it is analogous to calling a function that takes as arguments the bindings of colors to variables.

**Definition 17 (Symmetric Net)**

A symmetric net is a tuple $\mathcal{N} = (P, T, W^-, W^+, m_0, C, Domain, X)$ where

- $P$ and $T$ are set of places and transitions such that $P \cap T = \emptyset$.
- $C = \{C_1, \cdots, C_k\}$ is a set of color classes where each $C_i$ are pair-wise distinct and are finite set of colors. Each $C_i$ can furthermore be strictly ordered.
- $Domain : P \to C_{i_1}, C_{i_2}, \ldots, C_{i_k}$ where $\forall 0 \leq j \leq k,\ i_j$ is the domain function on places which assigns a sequence of color classes to each place.
- $Domain : T \to C_{i_1}, C_{i_2}, \ldots, C_{i_k}$ where $\forall 0 \leq j \leq k,\ i_j$ is the domain function on transitions which assign a sequence of color classes to each transition. This function specifies the domain of each color variable. For each transition a set $\{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$ of variables is defined. Each $x_i$ takes value in $C_i$.
- $W^-, W^+$ are the input and output arc functions, for all $p \in P$ and $t \in T$, $W^-(p,t)$ and $W^+(p,t)$ are functions that map valuations of colored variables in $Domain(t)$ to multisets of elements in $Domain(p)$.
- $X$ is the guard function. It associates to each transition a boolean function taking as input the valuation of color variables.

**Example 3** *The symmetric net depicted on Figure 2.10 is an extension of the plain Petri net of Figure 2.8. Sets of places and transitions are the same as for the plain Petri net. The set of color classes is the singleton* $\{C_1\}$. *The Domain function maps all places to the color class* $C_1$ *and, for each transition, assign variable* $x$ *to take values in* $C_1$. *All the guards are always*

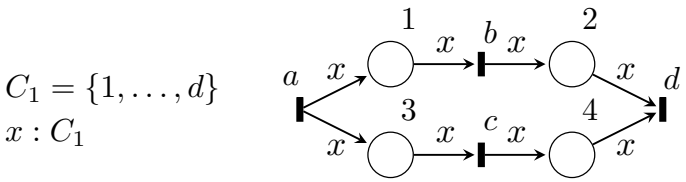



Figure 2.10: A simple example of symmetric net

*satisfied. Incidence matrices are as follows:*

$$W^- = \begin{pmatrix} 0 & 0 & 0 & 0 \\ x & 0 & 0 & 0 \\ 0 & 0 & x & 0 \\ 0 & x & 0 & x \end{pmatrix}, \quad W^+ = \begin{pmatrix} x & 0 & x & 0 \\ 0 & x & 0 & 0 \\ 0 & 0 & 0 & x \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Then stochastic symmetric net are defined by adding a distribution to each transition of the net which parameters may depends of the binding color of the transition.

**Definition 18 (Stochastic Symmetric Net(SSN))**

A stochastic symmetric net is a tuple $\mathcal{N} = (P, T, W^-, W^+, m_0, C, Domain, X, \Lambda)$ where

- $(P, T, W^-, W^+, m_0, C, Domain, x)$ is a symmetric net
- $\Lambda$ is the distribution function. It associates to each transition a distribution of probabilities and takes as input the valuation of color variables.

**Remark 3** *In this definition, the function $\Lambda$ associates to each transition a distribution which can be arbitrary. Usually SSN are defined with some restrictions on the available distributions. When one wants the resulting model to be Markovian only exponential distributions and immediate distributions are allowed, as generalized stochastic Petri nets.*

**Running example.** *Figure 2.11 shows a SSN representing a tandem queues system with $d$ queues. The unique color class $C_1$ is equal to $\{1, 2, \ldots, d\}$, there is only one place $Q$. The domain for the place and transitions is $C_1$. Color variables are declared globally instead of locally to each transition to simplify notation, thus there is only one variable $x$ taking value in $C_1$.*

*In this example the pattern of queues is factorized in one place. It allows to parametrize the model by the color class $C_1$. This example is not symmetric by every permutation of colors in the color class, thus for this*

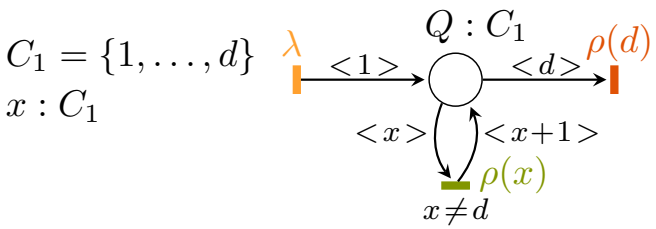



Figure 2.11: SSN representing a tandem queues system with $d$ queues

*example one cannot use the symmetric representation to analyze the system more efficiently.*

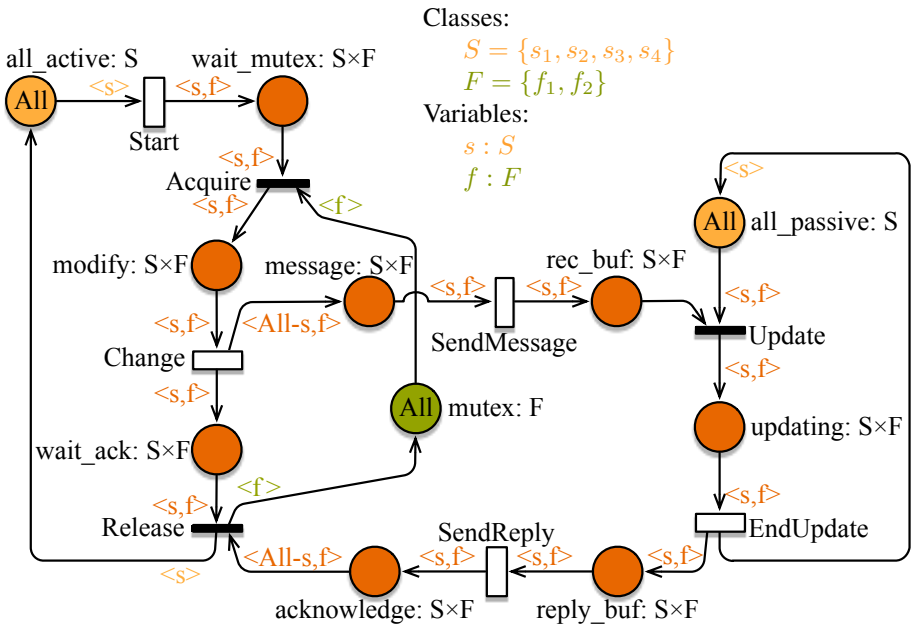



Figure 2.12: SSN representing the distributed database example

**Example 4** *Figure 2.12 shows an SSN representing a distributed database example. This example contains two color classes $S$ for site and $F$ for file. All thin plain transitions are immediate transitions, all other transitions are distributed exponentially with rate 1. Each site contains a copy of each file. This model implements a crude protocol to ensure the integrity of files across different databases using global mutexes for each file. The workflow is as follows:*

1. *If a user on site $s$ wants to use a file $f$ it takes transition* `Start`*. Then he waits for the common mutex on this file to be available.*

2. *As soon as the mutex is available it is acquired with transition* `Acquire`*. Then the user can start to edit the file.*

3. *When modifications to the file are finished, transition* `Change` *is fired and messages are sent to all sites distinct from* $s$*.*

4. *Each message arrives to its site with transition* `SendMessage`*.*

5. *Each site starts to update files according to messages as soon as possible with transtion* `Update`*.*

6. *When the update is finished each site sends a reply with transition* `EndUpdate`

7. *Each reply arrives to the site that initiates the change of the file with transition* `SendReply`*.*

8. *As soon as all replies, one from every site, are received the mutex of the file is released and the site returns to an idle state with transition* `Release`*.*

*In this SSN any permutation in* $S$ *and* $F$ *does not change the behavior thus this system can be analyzed efficiently using symbolic representation taking into account these symmetries. Symmetries are syntactically checked: no arc valuation distinguishes color tokens, the model contains no guard and the initial state is invariant by all permutations of colors.*

## 2.3 Statistical Methods

### 2.3.1 Monte Carlo Method

The Monte Carlo method computes an estimation of $\mathbb{E}(X)$ when $X$ is a random variable following an unknown distribution law $\mathcal{L}$, defined as the result of a computation on a trajectory of a probabilistic system such that $\mathbb{V}(X)$ is finite. If $X$ follows a known distribution up to a parameter, a Monte Carlo method can be used to estimate this parameter. A Monte Carlo method contains two distinct parts.

- The first part is a simulator that produces realizations of $X$. Ideally the simulator must produce independent realizations of this law. To do this a simulator requires a source of randomness that produces independent realizations of a known law. In general such a source of randomness is approximated by a pseudo random number generator producing real numbers chosen uniformly in $[0; 1]$ like Mersenne-Twister [70]. The produced samplings are not really randomly chosen and are not independent because they are produced by a deterministic algorithm

but the number of required samplings to distinguish such a sampling from a random sampling is huge.

The simulator takes as input a description of the system that one wants to simulate. The system is usually described as a stochastic process. For each simulated trajectory the algorithm returns a value, usually a real number, which is a realization of $\mathcal{L}$.

- The second part is a deterministic algorithm which takes as input a sequence of realizations and produce an estimation of the result that one wants to compute. This part is the Monte Carlo estimator. As the goal is to estimate an expectancy, the best estimator is the mean of all the values. The estimator is itself a random variable. After $N$ simulations of the system, the estimator posses $N$ independent identically distributed copies of $X$ noted $(X_i)_{i=1}^N$. The estimator is $Z = \frac{1}{N}\sum_1^N X_i$. Basic probabilistic property shows that this estimator is an unbiased estimator of $\mathbb{E}(X)$, meaning that its expected value is equal to $\mathbb{E}(X)$. Moreover it's variance is equal to $\frac{\mathbb{V}(X)}{N}$.

### 2.3.2 Confidence interval

The goal of a Monte Carlo method is to compute an estimation of $\mathbb{E}(X)$ for some variable $X$, when the exact value of this expected value can not be computed, one wants to know how far this approximation is from the actual value. It is not possible to guarantee that the numerical value we obtain is at a given distance to the actual value. That is after a number of sample, with a non-zero probability our simulation may have ignored a parts of the system. This leads to incorrect value for the estimation.

For example if one wants to check that a dice is well balanced, one can throw it 60 times; if the dice is indeed well balanced we can expect each face to show around 10 times. But even if one face never shows up, we can only suspect that the dice is biased, without certainty.

Intuitively, as the variance of $Z$ tends to 0 when $N$ tends to infinity, for a fixed $N$ and a fixed real $\delta > 0$, the variance of $Z$ is smaller than that of $X$; thus, a realization $\hat{p}$ of $Z$ will be more likely to be in the interval $[\mathbb{E}(X) - \delta; \mathbb{E}(X) + \delta]$ than a realization of $X$.

The notion of confidence interval allows to have some probabilistic guarantee on a result obtained using random algorithms. This guarantee takes the form of confidence on the fact that the actual value is close enough to the realization. In a more general setting a confidence interval is defined as follows:

**Definition 19 (Confidence Interval)**

Let $(X_i)_1^N$ be independent random variables following a common distribution including a parameter $\theta$. Let $0 < \gamma < 1$ be a confidence level. Then a confidence interval for $\theta$ with level at least $\gamma$ is given by two random variables $l(X_1, \ldots, X_N)$ and $u(X_1, \ldots, X_N)$ such that for all $\theta$:

$$\mathbb{P}\left[l(X_1, \ldots, X_N) \leq \theta \leq u(X_1, \ldots, X_N)\right] \geq \gamma$$

**Theorem 5 (Markov Inequality)**

Given a positive random variable $Y$ the following equality holds:

$$\forall \beta > 0,\ \beta \cdot \mathbb{P}\left[Y \geq \beta\right] \leq \mathbb{E}(Y)$$

In the context of Monte Carlo method the parameter that one wants to frame is $\mathbb{E}(X)$. An easy way to build a confidence interval without making any new assumption on the distribution law of $X$ is to use the inequality of Chebyshev.

**Theorem 6 (Chebyshev's inequality)**

Given a random variable $X$ such that $\mathbb{E}(X)$ and $\mathbb{V}(X)$ are finite. The following inequality holds:

$$\forall \alpha > 0,\ \mathbb{P}\left[|X - \mathbb{E}(X)| \geq \alpha\sqrt{\mathbb{V}(X)}\right] \leq \frac{1}{\alpha^2}$$

Using the Chebyshev inequality on the random variable $Z$ using $N$ samples, one can choose $\alpha$ such that $1 - \frac{1}{\alpha^2}$ is equal to the confidence level $1 - \varepsilon$. The inequality now states that for a realization $\hat{p}$ of $Z$ we have:

$$\mathbb{P}\left[\mathbb{E}(X) \in \left[\hat{p} - \sqrt{\frac{\mathbb{V}(X)}{N\varepsilon}}, \hat{p} + \sqrt{\frac{\mathbb{V}(X)}{N\varepsilon}}\right]\right] \geq 1 - \varepsilon$$

which is a confidence interval.

Several remarks can be made on this confidence interval:

- The speed of convergence of the Monte Carlo method with respect to $N$ is in $O\left(\frac{1}{\sqrt{N}}\right)$. This is not a fast convergence but the only hypothesis that we make is that the variance is finite.

- The width of the confidence interval depends highly on the variance of the distribution law. Classical methods to improve the speed of convergence of Monte Carlo algorithm are based on the reduction of this variance.
- The variance of the distribution is unknown. To use this confidence interval, one has to estimate or bound this variance. This fact contributes greatly to the rare event problem of Monte Carlo methods.

To obtain tighter confidence intervals or to reduce the require number of simulations, one can make stronger assumptions on the shape of the distribution of $X$ and use more advanced statistical results to build confidence intervals.

**Functionalities** Statistical procedures have different kinds of inputs and outputs:

- The output can be a boolean corresponding to the comparison between a probability and a threshold. Corresponding statistical techniques are called *hypothesis testing*.
- The input can be an expression involving the expectations of random variables and the output is an estimation given as a confidence interval framing the value.

**Probabilistic Guarantee** As the result is obtained by a statistical estimation, there are two kinds of probabilistic guarantees. Denoting by $p$ the confidence level,

- The probability that the result is wrong is upper bounded by $p$. Here the result is either the comparison of a probability with a threshold or the framing of a parameter inside a confidence interval.
- The probability that the result is wrong is *asymptotically* upper bounded by $p$. The limit can be taken when the number of samples goes to infinity or when the width of the confidence interval goes to 0. In this case only asymptotic confidence intervals are obtained.

**Static versus sequential sampling**

- A *static procedure* takes as input or initially computes the number of simulations before performing these simulations.
- A *sequential procedure* both performs successive simulations and estimates the stopping criterion which depends on the simulation outputs.

| Guarantee | Assumption on the law | | | | Process |
|---|---|---|---|---|---|
| | $\emptyset$ | Bounded | Normal | Bernouilli | |
| Asymptotic | Gaussian | C-H | Gaussian | C-P | Static |
| Exact | $\emptyset$ | | | | |
| Asymptotic | C-R | C-R | C-R | C-R | Sequential |
| Exact | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | |

Table 2.1: Summary of statistical method producing confidence intervals. C-P stands for Copper-Pearson, C-H stands for Chernoff-Hoeffding and C-R stands for Chow-Robbins

### 2.3.3 Gaussian Confidence Interval

In the general case, the classical approach is based on the hypothesis that the normal law is a good approximation of the simulated law. A discussion on the validity of this approximation for the estimation of a rare event is made in Section 3.2.

Thus, given a number of paths $N$ and a confidence level $1-\varepsilon$, the method produces a confidence interval.

**Theorem 7**

If $X$ follows a normal distribution a confidence interval can be computed for a confidence level of $1-\varepsilon$. Let $z$ be such that $2\phi(z)-1 = 1-\varepsilon$. The following equation holds:

$$\mathbb{P}\left(\mathbb{E}(X) \in \left[\hat{p} - z\sqrt{\mathbb{V}(X)}, \hat{p} + z\sqrt{\mathbb{V}(X)}\right]\right) = 2\phi(z) - 1$$

Where $\phi$ is the *error function* sometime written *erf* defined by

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{-u^2}{2}} du$$

As the mean of normally distributed random variables still follow a normal law, this result can be used with $N$ i.i.d. copies of $X$. We then obtain the following equality:

$$\mathbb{P}\left(p \in \left[\hat{p} - z\frac{\sqrt{\mathbb{V}(X)}}{\sqrt{N}}, \hat{p} + z\frac{\sqrt{\mathbb{V}(X)}}{\sqrt{N}}\right]\right) = 2\phi(z) - 1$$

To use this theorem one should find a suitable $z$. Fortunately such a $z$ only depends on the confidence level. The value $z$ is computed only once using implementation of the $\phi^{-1}$ function. Indeed, this function is well known

and precise approximation exists and are implemented in standard library of most programming language.

As mentioned for the Chebyshev's inequality for a fixed distribution of $X$ and a fixed confidence level, the confidence interval width is still proportional to $\frac{1}{\sqrt{N}}$. To apply this result one needs to know $\mathbb{V}(X)$. Fortunately there exists an unbiased estimator of $\mathbb{V}(X)$.

**Theorem 8**

Given $N$ i.i.d. random variables $(X_i)_{i=1}^N$. Let $V$ be a random variable equal to

$$\frac{1}{N-1}\sum_{i=1}^{N}(X_i - Z)^2$$

The random variable $V$ is an unbiased estimator of $\mathbb{V}(X)$

If $X$ does not follow a normal law but the sample is large enough and both the mean and the variance of $X$ are finite, we can use the central limit theorem:

**Theorem 9 (Central Limit Theorem)**

Let $(X_i)_1^N$ a sequence of i.i.d. random variables of finite expectancy $\mathbb{E}(X)$ and finite variance $\mathbb{V}(X)$ then when $N$ tends to the infinity the empirical mean converges in distribution to a normal law.

$$\sqrt{N}\left(\left(\frac{1}{N}\sum_{i=1}^{N}X_i\right) - \mathbb{E}(X)\right) \xrightarrow{d} \mathcal{N}(0, \mathbb{V}(X))$$

This method can be used in a static way by simulating $N$ samples and applying the Gaussian confidence interval method. Or one can use results from [21] to build a sequential procedure.

Let $x_1, x_2, \ldots$ an infinite sequence of realizations of $X$. We are interested in computing a confidence interval of a given width $2d$ with a confidence level $1-\varepsilon$, using the least number of terms in $(x_i)$. Let $a = \phi^{-1}\left(1 - \frac{\varepsilon}{2}\right)$ Let $N \geq 1$ the number of sample be the smallest integer such that:

$$\frac{1}{N}\sum_{i=1}^{N}\left(x_i - \overline{x}_N\right) + \frac{1}{N} \leq \frac{d^2}{a^2} \text{ where } \overline{x}_N = \sum_{i=1}^{n}\frac{x_i}{N}$$

The following theorem ensure that a Monte Carlo estimator with $N$ samples produce a confidence interval a width smaller than $2d$.

**Theorem 10 (Chows and Robbin)**

If $0 < \mathbb{V}(X) < \infty$ then

1. $\lim_{d\to 0} \frac{d^2 N}{a^2 \mathbb{V}(X)} = 1$ a.s.,
2. $\lim_{d\to 0} \mathbb{P}\left(p \in [\overline{x}_N - d, \overline{x}_N + d]\right) = 1 - \varepsilon$
3. $\lim_{d\to 0} \frac{d^2 \mathbb{E}(N)}{a^2 \mathbb{V}(X)} = 1$

Compare to Gaussian analysis, this theorem allows to sequentially compute a confidence interval of a given width with the additional requirement that the width of the confidence interval tends to zero.

### 2.3.4 Chernoff-Hoeffding Bounds

When the distribution of the random variable $X$ has a bounded support, Chernoff-Hoeffding's bounds [49] allow to compute a tighter conservative confidence interval.

**Theorem 11 (Chernoff-Hoeffding inequality)**

Let $X_1, \ldots, X_N$ be $N$ independent random variables. Assume there exists $(a_i)_{i=1}^N$ and $(a_i)_{i=1}^N$ such that $\forall i,\ \mathbb{P}(X_i \in [a_i, b_i]) = 1$ then

$$\mathbb{P}\left(\left|\frac{1}{N}\sum_{i=1}^{N} X_i - \mathbb{E}(X_i)\right| \geq z\right) \leq 2\exp\left(-\frac{2N^2 z^2}{\sum_{i=1}^{N}(b_i - a_i)^2}\right)$$

Which can be written as

$$\mathbb{P}\left(\mathbb{E}(X) \in \left[\hat{p} - z\sqrt{\frac{\sum_{i=1}^{N}(b_i - a_i)^2}{N}}, \hat{p} + z\sqrt{\frac{\sum_{i=1}^{N}(b_i - a_i)^2}{N}}\right]\right) \geq 1 - 2e^{-2z^2}$$

Written like this, the confidence interval can be compared to the Chebyshev one. The variance have been replaced by $\sum_{i=1}^{N}(b_i - a_i)^2$ which is an upper bound on the variance. The speed of convergence with respect to $N$ is also $\frac{1}{\sqrt{N}}$.

### 2.3.5 Confidence Interval for Binomial law

In the case where a random variable $X$ follows a binomial law for which we want to estimate the parameter $\theta$, that is $X$ takes values in $\{0, 1\}$ and $\theta = \mathbb{P}(X = 1)$, a static confidence interval can be computed on the

distribution. Let $x_1, x_2, \dots x_N$ be a sequence of $N$ independent realization of $X$. Let $x = \sum_{i=1}^{N} x_i$. A confidence interval for the binomial law with confidence level $1 - \varepsilon$ can be expressed as:

$$\left\{\theta \mid \mathbb{P}\big(\mathrm{Bin}(N,\theta) \leq x\big) > \frac{\varepsilon}{2} \wedge \mathbb{P}\big(\mathrm{Bin}(N,\theta) \geq x\big) > \frac{\varepsilon}{2}\right\}$$

where $\mathrm{Bin}(N,\theta)$ denotes a random variable following a binomial distribution with $N$ trials and a probability of success $\theta$. This confidence interval is known as Clopper-Pearson confidence interval [23]. It can be expresses using quantiles of the *Beta distribution* (denoted $B$) as:

$$\left[B\left(\frac{\varepsilon}{2}, x, N+1-x\right), B\left(1-\frac{\varepsilon}{2}, 1+x, N-x\right)\right]$$

The Beta distribution is a continuous function for which a good approximation exists. Therefor it is possible to numerically compute this confidence interval.

### 2.3.6 Hypothesis testing

When one is not interested in the actual value of a statistic but rather wants to decide whether this value is above or below a threshold, hypothesis testing methods can be used.

Let us focus on a binomial distribution of unknown parameter $p$. Given a threshold probability $p'$ we want to know whether $p > p'$ or $p < p'$ with high confidence. It is not possible to decide this when $p'$ is arbitrary closed to $p$ thus we define an indifference region: $]p_0, p_1[$ containing $p'$ and we defined two hypothesis:

- $H_0$ is the hypothesis that $p \leq p_0$,
- $H_1$ is the hypothesis that $p \geq p_1$.

Given two confidence level $\alpha$ and $\beta$ we want that the probability to accept hypothesis $H_1$ whereas $H_0$ holds is less than $\alpha$ and the probability to accept hypothesis $H_0$ whereas $H_1$ holds is less than $\beta$. If $p$ lies in the indifference region both hypothesis are acceptable.

Given $\alpha$, $\beta$, $p_0$ and $p_1$, the Sequential Probability Ratio Test (SPRT) [85] is an optimal sequential test for deciding whether $H_0$ or $H_1$ holds.

This test is as follows: after $N > 0$ samples, let $x$ be the number of successful samples. We define

$$\frac{p_{1,N}}{p_{0,N}} = \frac{p_1^x (1-p_1)^{N-x}}{p_0^x (1-p_0)^{N-x}}$$

There are three cases:

- If $\frac{p_{1,N}}{p_{0,N}} \leq \frac{\beta}{1-\alpha}$ then accept hypothesis $H_0$.

- If $\frac{p_{1,N}}{p_{0,N}} \geq \frac{1-\beta}{\alpha}$ then accept hypothesis $H_1$.
- Otherwise $\frac{1-\beta}{\alpha} < \frac{p_{1,N}}{p_{0,N}} < \frac{\beta}{1-\alpha}$. We cannot conclude without further simulation.

**Remark 4** *To avoid effectively computing* $\frac{p_{1,N}}{p_{0,N}}$ *which is likely to be numerically unstable for large value of* $N$*, the logarithm of this expression is computed:*

$$\log\left(\frac{p_{1,N}}{p_{0,N}}\right) = x\log\left(\frac{p_1}{p_0}\right) + (N-x)\log\left(\frac{1-p_1}{1-p_0}\right)$$

*and compared to* $\log\left(\frac{1-\beta}{\alpha}\right)$ *and* $\log\left(\frac{\beta}{1-\alpha}\right)$.

## 2.4 Model Checking

Model checking has been widely used since its introduction in [31]. It allows to formally check that a *system* satisfies a *specification* in a fully automated way. More often, a model checking algorithm checks if the specification of a system holds by exploring all its possible states. This requires the construction of the reachability graph. The simplicity of this approach allows automation whereas other methods of verification often require human intervention. But for large or infinite systems it is not possible to build and explore the reachability graph. Several techniques are then used to reduce the state space. For example it is possible to generate the state space on the fly or to use symbolic representation of the states space.

For probabilistic system one can be interested in the quantitative model checking. In this setting the probabilities that a trajectory satisfy a particular property is computed, or whether this probability is above some threshold.

### 2.4.1 Specification

There are two main families of specification languages for probabilistic systems. The first is based on the Linear Temporal Logic (LTL) and specifies accepted paths in the system. The probability for a path of a system to be accepted is specified, thus this logic does not allowed nested probabilistic operator. The other one is based on the Computational Tree Logic (CTL) where only state formula are considered. In this logic probabilistic operator and until operator can be nested but have to alternate.

#### PLTL

The Probabilistic Linear Temporal Logic (PLTL) allows to specify whether the probability of paths in a system satisfying an LTL formula. A PLTL

formula if of the form $\mathbb{P}_{\bowtie p}(\phi)$ where $\bowtie \in \{<, >\}$, $p \in [0,1]$ and $\phi$ is an LTL formula given by the following grammar:

$$\phi ::= q \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \phi \,\mathbf{U}^{I}\, \phi \mid \mathbf{X}^{I}\phi$$

Where $q$ is an atomic proposition of the system, $\neg, \vee$ and $\wedge$ have there usual meanings. LTL is a *path-based* logic, that is the semantics of all operators is defined on a path of the system. The $\mathbf{U}$ and $\mathbf{X}$ operator are equipped with an interval $I \subset \mathbb{R}^+$. The until operator means that along a path, the formula of the left member must hold until a state where the formula of the right member holds. The $\mathbf{X}$ operator means that in the next discrete step the formula must hold. The formal semantics is as follows: given a non-Zeno realization of a DEDS $\sigma = ((s_i)_{i=0}^{\infty}, (t_i)_{i=0}^{\infty})$ and a time $t$, let $n$ such that $\sum_{i=0}^{n} t_i \leq t < \sum_{i=0}^{n+1} t_i$

$$\begin{array}{ll}
\sigma, t \models q & \text{if } s_n \models q \\
\sigma, t \models \neg\phi & \text{if } \sigma, t \not\models \phi \\
\sigma, t \models \phi_1 \wedge \phi_2 & \text{if } \sigma, t \models \phi_1 \text{ and } \sigma, t \models \phi_2 \\
\sigma, t \models \phi_1 \vee \phi_2 & \text{if } \sigma, t \models \phi_1 \text{ or } \sigma, t \models \phi_2 \\
\sigma, t \models \mathbf{X}^{I}\phi & \text{if with } t' = \sum_{i=0}^{n+1} t_i,\ t' - t \in I \text{ and } \sigma, t' \models \phi \\
\sigma, t \models \phi_1 \,\mathbf{U}^{I}\, \phi_2 & \text{if there exists } t' \geq t \text{ s.t. } t' - t \in I \\
 & \qquad \text{and } \forall t'' \in [t, t'[,\ \sigma, t'' \models \phi_1 \text{ and } \sigma, t' \models \phi_2
\end{array}$$

Generalization of this logic consists in replacing the LTL formula by some kind of automaton specifying which are accepting paths.

### PCTL

The PCTL logic [43] allows to specify properties using nested probabilistic operator thus branching properties. The PCTL logic is *state based* logic but it contains path based subformula. It is described by the following grammar:

$$\begin{array}{l}
\psi ::= q \mid \neg\psi \mid \psi \vee \psi \mid \psi \wedge \psi \mid \mathbf{P}_{\bowtie p}(\phi) \\
\phi ::= \psi \,\mathbf{U}^{I}\, \psi \mid \mathbf{X}^{I}\psi
\end{array}$$

where $\psi$ denotes a state based formula and $\phi$ denotes a path based formula. A PCTL formula always starts by a state based fromula. LTL operators have their usual meaning: $\mathbf{P}_{\bowtie p}\big(\psi_1 \,\mathbf{U}\, \psi_2\big)$ specify whether the probability for $\psi_1 \,\mathbf{U}\, \psi_2$ to holds is above or below the threshold $p$ in the initial state. The semantics of path based operator is the same as in LTL. The formal semantics of state base operators is as follows, with $s \in S$,

$$\begin{array}{ll}
s \models q & \text{if } s \models q \\
s \models \neg\phi & \text{if } s \not\models \phi \\
s \models \phi_1 \wedge \phi_2 & \text{if } s \models \phi_1 \text{ and } s \models \phi_2 \\
s \models \phi_1 \vee \phi_2 & \text{if } s \models \phi_1 \text{ or } s \models \phi_2 \\
s \models \mathbf{P}_{\bowtie p}\big(\psi\big) & \text{if } \mathbb{P}\Big(\big((S_i)_{i=0}^{\infty}, (T_i)_{i=0}^{\infty}\big), 0 \models \phi \mid S_0 = s\Big) \bowtie p
\end{array}$$

A generalization of this logic is the Continuous Stochastic Logic (CSL) with an extra operator allowing to specify properties of the steady state probability.

### 2.4.2 Numerical Model Checking

Numerical models are best suited for PCTL type formula. Considering the formula as a tree, they are evaluated starting from the leaf. Given a sub-formula consisting of one probabilistic operator at the root and no nested probabilistic operator, the evaluation of **U** and **X** formulas can be reduced to time-bounded or time unbounded reachability problems as sub-formulas are state formulas. These reachability problems are solved using Definition 9, 14 or Theorem 3. Once the sub-formula is evaluated on each state it can be replaced by a proposition in the upper sub-formula until the root is reached. One can refer to the chapter 10 of [7] for details about model checking of probabilistic systems.

For a linear specification, the formula is translated into an automaton, the synchronized product of the stochastic system with the automaton is then built. This product is still a stochastic system, where the model checking problem reduced to time-unbounded reachability. If the formula is large the size of the automaton and the size or the product may become problematic.

Most numerical model checking algorithms rely on matrix vector multiplication, usually taking advantage of the sparsity of transition probability matrices. However these computations are difficult to parallelized. The tool Marcie [46] features a parallel implementation of this method but still a non negligible part of the computation cannot be performed in parallel. Due to Amdahl's law [5], the gain of parallel computing is limited.

Strong probabilistic hypotheses on the stochastic system to be analyzed are mandatory. The easiest case is to suppose that the system is *Markovian* and have a finite state space. Even with theses strong hypotheses the system of linear equations required by the computation can be huge. In that case, the computation of solutions becomes difficult and therefore requires iterative methods. Only on approximated solution of such a system is then obtained.

For a more complex linear specification like one given by a time automaton, efficient techniques exist only if there is at most one clock in the automaton see for example [28] and [11].

This approach can be adapted to deal with non determinism [84] and to tackle model checking problem on *Markov decision processes*. In this setting probabilistic transitions are alternated with non deterministic ones.

This approach has been efficiently implemented in several tools with different types of Markovian systems. For example Prism [64] can deal with *Discrete Time Markov Chains*, *Continuous Time Markov Chains*, *probabilistic automata* and *Markov decision processes*. Uppaal [12] deals with various models of automata. GreatSPN [20] and Marcie [46] deal with Stochastic

Petri nets. Others tools exist: MRMC [58], ...

### 2.4.3 Statistical Model Checking

The alternative to numerical approach when dealing with large systems is to use statistical methods. The statistical model checking relies on a *Monte Carlo* algorithm to estimate the probability of interest. Precisely, on a linear specification, a random variable $X$ is defined which takes 1 as value on a trajectory which satisfies the specification and 0 otherwise. Thus this variable follows a Benouilli law and we compute its *expected value* $\mathbb{E}(X)$.

This method is more adapted to linear specifications. The Monte Carlo algorithm simulates a large number of trajectories, say $N$, and counts the number of those trajectories satisfying the specification. An estimation of the probability is obtained as the ratio of the number of successful trajectories on the total number of trajectories. The random variable $Z$ is defined as the mean of $N$ independent copies of $X$:

$$Z = \frac{1}{N} \sum_{i=1}^{N} X_i$$

Statistical model checking can be parallelize very easily: it suffices to run several simulators of the system on different processors or different machines and take the mean result of trajectories of all simulators. Particular attention is required on the random number generator to ensure that all generated trajectories are independent one from each others. As the only sequential operation in this approach is to compute the mean value and possibly a confidence interval, there is almost no cost in using parallel computing.

Statistical method can be naturally extended to performance evaluation. Instead of computing a probability, the expected value of a random variable those values depend on trajectories of the system is computed.

### 2.4.4 Comparison Between Numerical and Statistical Model Checking

Statistical methods have several advantages compared to numerical ones.

1. The memory required to simulate one trajectory in the system is usually very small. Thus the memory requirement of a statistic model checker is very low. Whereas the memory is often the bottleneck of numerical methods.

2. As long as we have an operational semantics of a probability distribution (i.e. we have a probabilistic algorithm whose outputs follow this distribution law), the statistical method can be used to analyze any system using this law.

3. As all trajectories are independent, it is straightforward to parallelize the simulation whereas as we already mentioned, it is much more difficult for numerical computation.

4. Complex linear properties like the one expressed in the *Hybrid Automaton Stochastic Logic* (HASL) [9] can be efficiently evaluated.

However, the statistical method has several drawbacks:

1. If precise results are required, which means that we want a tight confidence interval, the time of computation can become big. In general the width of the confidence interval decreases according to the square root of the number of simulations. This means that in order to divide by two the width of the confidence interval we need to perform four time more simulations.

2. Difficulties can occur when the simulation is used to compute a stationary distribution. The difficulties lie in the fact that a stationary distribution is defined as a limit of distributions of states after an infinite number of time unit. As simulations are finite, the initial distribution introduces a bias which can be difficult to correct.

3. Statistical model checking is not well suited to evaluate logic based on state formula because it requires to perform simulations starting from each state which is too costly.

4. For properties expressed as nested CSL probabilistic operator, the difficulty relies on the fact that nested probabilistic operators require to estimate first the most nested operator and then to evaluate the formula bottom-up. For each nesting new simulations of the system is needed. Careful computations of confidence interval are required to take into account the errors at each level of nesting. Details can be found in [86].

5. Statistical model checking cannot evaluate unbounded until operator $\phi \mathbf{U} \psi$: if there is a bottom strongly connected component in the system where $\phi$ holds, as the simulator only evaluates finite trajectories, it cannot decide if the formula holds. A workaround is described in [44]. Tools rely on more pragmatic approaches: either the unbounded until is not available in the specification language (ex UPPAAL) or the simulator does not terminate when simulating a trajectory always satisfying $\phi$ (ex COSMOS).

6. Finally crude Monte Carlo estimator is unsuitable to compute very small probabilities as for the probability of *rare event*. This problem is the subject of the next chapter. Contrary to statistical model checker, numerical one are not subject to rare events. In some case, reachability

of a rare event may lead to computations using *ill-conditioned* transition probability matrices. In this case the convergence time of iterative method may increase.

| | Numerical | Statistical |
|---|---|---|
| Result | Exact (up to numerical error) | Probabilistic framing |
| Memory | Store transition matrix and a vector of probability | Store one state of the system |
| Parallelism | Difficult | Free (almost) |
| Models | Markovian | General distribution |
| Determinism | Handle non-determinism | Requires fully stochastic models |
| Logic | PCTL and PLTL (The second increase the state space) | PLTL, automata based logic, no nested operator |
| Unbounded until | Supported | Require hypothesis on the model |
| Stationary Distribution | Exact | Biased by initial distribution |
| Rare events | Some numerical instability | Intractable |

Table 2.2: Summary of comparisons between statistical and numerical model checking.

# Chapter 3

# Rare Events

## 3.1 Introduction

The main drawback of statistical methods is the rare-event problem. This problem occurs when the probability that a trajectory satisfies a formula is very small, say less than $10^{-9}$. The satisfaction of such formula is then called a *rare event* with respect to the model. In order for statistical methods to work in this case, a large number of trajectories must be generated to obtain an accurate estimation of the result. The time required to produce such a large number of trajectories makes the statistical approach in the rare-event setting unfit. To enhance Monte Carlo methods, two main families of algorithms tackling this problem have been developed, namely splitting and importance sampling.

In Section 3.2, the rare-event problem is studied in details. In Section 3.3, splitting methods are presented and importance sampling methods are discussed in Section 3.4.

## 3.2 Rare-Event Problem

Let us illustrate the rare-event problem with an example. Assume we want to estimate the probability of an event occurring with a probability smaller than $10^{-15}$ with a confidence level of 0.99. Suppose we are able to compute up to a billion trajectories. Let us describe the results of plain Monte Carlo simulation depending on all possible outcomes.

- With a large probability, approximately $1 - 10^{-6}$, the rare event does not occur in any trajectory. In this case, the corresponding confidence interval is $[0, 7\ 10^{-9}]$. This confidence interval contains the value we are looking for, but the width of this confidence interval is large. The precision of the result is thus very low.

- With a small probability, smaller than $10^{-6}$, the rare event occurs in one

trajectory and the corresponding confidence interval is $[7\ 10^{-10}, 2\ 10^{-9}]$. This confidence interval does not contain the expected value.

- With an even smaller probability, the rare event occurs in more than one trajectory and the confidence interval does not contain the expected value either.

In all cases, the number of trajectories required to get an accurate estimate of the probability with the plain Monte Carlo statistical method is intractable. In this example the result is not satisfactory.

By modeling the occurrence of a rare event by a Bernouilli variable $X$ with the unknown parameter $p$, another way to illustrate the rare-event problem is to look at the variance of the random variable $Z$ associated with the Monte Carlo estimator of the rare event. This random variable follows a binomial law whose variance can be computed by the following formula $\mathbb{V}(Z) = \frac{p(1-p)}{N}$ with $p = \mathbb{E}(X)$ the probability of the event to occur. When $p$ is small, the variance can be approximated by $\frac{p}{N}$ as the term in $O(p^2)$ is negligible. A confidence interval may be computed using for instance the Chebyshev Inequality. The confidence interval width is then proportional to the standard deviation up to a factor depending of the confidence level. As the standard deviation is the square root of the variance, the confidence interval width is proportional to $\frac{\sqrt{p}}{\sqrt{N}}$. When $p$ is small, the width of the confidence interval is big compared to $p$, making the estimation not accurate. Other more involved methods (like Clopper-Pearson confidence interval) producing a confidence interval reduce the factor depending of the confidence level. However the width of the confidence interval remains proportional to $\frac{\sqrt{p}}{\sqrt{N}}$.

Experimentally, this can be observed by simulating a parametrized system where the probability of the event of interest can be made arbitrary small using a parameter. Then, by estimating the value of this probability with a confidence interval of fixed relative width (*i.e.*, the ratio of the confidence interval width on the probability is always below some threshold) with a sequential method, one observes that the simulation time quickly increases when the probability tends to zero. Figure 3.1 illustrates this on a tandem queue system. When the probability becomes tiny, we observe that the simulation time explodes.

Most methods that compute confidence intervals use the variance, but this variance is unknown. Only estimations of the variance are available when one is computing the probability of a rare event. For instance, suppose that we want to estimate the probability of occurrence of a rare event with a probability less than $10^{-6}$ but not equal to zero. In general two approximations are usual for the computation of the confidence interval. The first one assumes that the estimator follows a normal law, its approximation being justified by the central limit theorem. The second one consists in

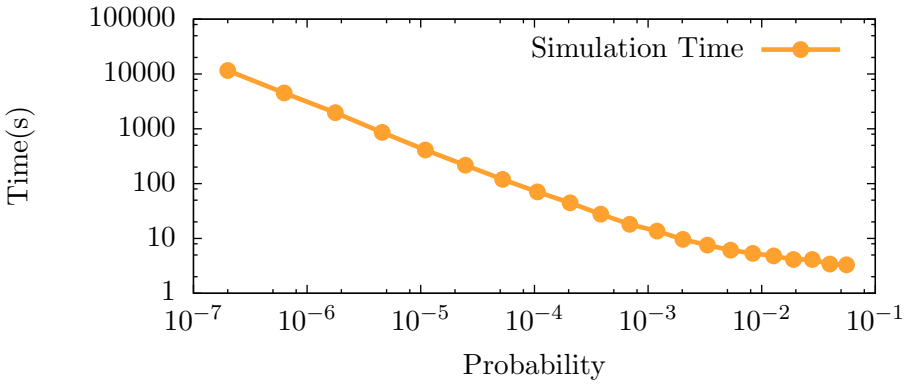



Figure 3.1: Simulation time as a function of the probability for fixed relative width on a tandem queues system.

replacing $\mathbb{V}(X)$ by its estimation in the computation of the confidence interval. This second approximation is not correct when dealing with a rare event as the estimator for $\mathbb{V}(X)$ is also subject to rare event problem. Suppose we are able to run a thousand simulations: in most cases, the event will not occur in these thousand simulations. Thus the empirical variance computed with this estimator is equal to 0. A direct application of Theorem 7 with any confidence level returns a confidence interval equal to the singleton $[0, 0]$. This result is a underestimation of the true value. Underestimation of the value of interest is a characteristic of rare events. It may occur even when specific methods to deal with rare events are used. Some empirical rules exist to indicate whether or not one can invoke the convergence to the normal law. For instance, one rule states that both $N\mathbb{E}(X)$ and $N(1-\mathbb{E}(X))$ should be greater than 5. Most of them do not apply to rare events as $\mathbb{E}(X)$ is too small.

Historically, this problem has been first studied in particle transmission problem [56], where one is interested in computing the probability for a single particle to get across a material. If the material is thick enough, the probability is very small but as the initial number of particles can be large, the number of particles going through can still be significant. In this context, a small number of particles crossing the material can cause radiation poisoning so that getting an accurate analysis is critical. This problem has then been analyzed in a large variety of contexts. In general, estimating a tiny probability is important when the impact of a single occurrence of the event is tremendous and/or the process that contains this event is iterated a large number of times.

- In biology, rare-event probability problems can occur for estimating the speed of some molecular reactions. As the number of molecules is huge, even small probabilities of reaction lead to observable consequences.

- In telecommunication, probabilities of overflow or loss of packet could be small but the number of packets handled by a telecommunication system could be huge.
- In insurance, the probability of a stock market to crash should stay small to avoid drastic consequences.
- The dependability of a mechanical system could be quantified with probabilities. For critical system, this probability should be small. The probability of failure should be small not just for a given short period of time but for the whole lifetime of a system which may be very long.

## 3.3 Splitting

A classical method to deal with rare events with Monte Carlo simulation is the *splitting* technique [65]. A detailed description can be found in Chapter 3 of [78] or in [37]. According to [56], Dr. Von Neumann proposed the splitting technique in the late forties. It is usually defined by a sequence of successively embedded subsets of the state space of a Markov chain $\gamma_k \subset \cdots \subset \gamma_0$, where $\gamma_k$ contains exactly the states satisfying the rare event and $\gamma_0$ contains all states of the system except a sink state. The probability of interest $p$ is the probability to reach a state of $\gamma_k$ before returning to the sink state. In more general settings, this method can be used to estimate the probability to reach $\gamma_k$ before an arbitrary stopping time. Each time a trajectory reaches the next subset for the first time, this trajectory is *split* into several independent trajectories. Each trajectories carry a weight which is also split uniformly in each subtrajectories. Trajectories which start in the initial state have a weight of 1. Let $n_i$ be the number of trajectories in which a trajectory reaching $\gamma_i$ is split, $n_0$ being the number of trajectory starting from the initial state. With $H_i$ being the number of trajectories reaching $\gamma_i$ ($H_0 = 1$), the following is an unbiased estimator of $p$:

$$\frac{H_k}{n_0 n_1 \cdots n_{k-1}}.$$

This can be seen by introducing $p_i = \mathbb{P}(\gamma_i | \gamma_{i-1})$ the probability that a trajectory entering for the first time $\gamma_{i-1}$ reaches a state of $\gamma_i$ before reaching

the sink state. Then the following equalities hold:

$$\begin{aligned}
\mathbb{E}\left(\frac{H_k}{n_0 n_1 \cdots n_{k-1}}\right) &= \sum_{i=0}^{n_0 n_1 \ldots n_k} \mathbb{E}\left(\frac{H_k}{H_{k-1} n_{k-1}} \Big| H_{k-1}\right) \frac{i\mathbb{P}(H_{k-1}=i)}{n_0 n_1 \cdots n_{k-2}} \\
&= \sum_{i=0}^{n_0 n_1 \ldots n_{k-1}} \mathbb{P}(\gamma_k|\gamma_{k-1}) \frac{i\mathbb{P}(H_{k-1}=i)}{n_0 n_1 \cdots n_{k-2}} \\
&= p_k \mathbb{E}\left(\frac{H_{k-1}}{n_0 n_1 \cdots n_{k-2}}\right) \\
&= \cdots \\
&= p_1 p_2 \ldots p_k = p
\end{aligned}$$

These equalities hold since $\frac{H_i}{H_{i-1} n_{i-1}}$ is an unbiased estimator of $\mathbb{P}(\gamma_i|\gamma_{i-1})$.

As trajectories coming close to the rare event are split, the probability to have a trajectory reaching the rare event is larger than in a crude Monte-Carlo simulation.

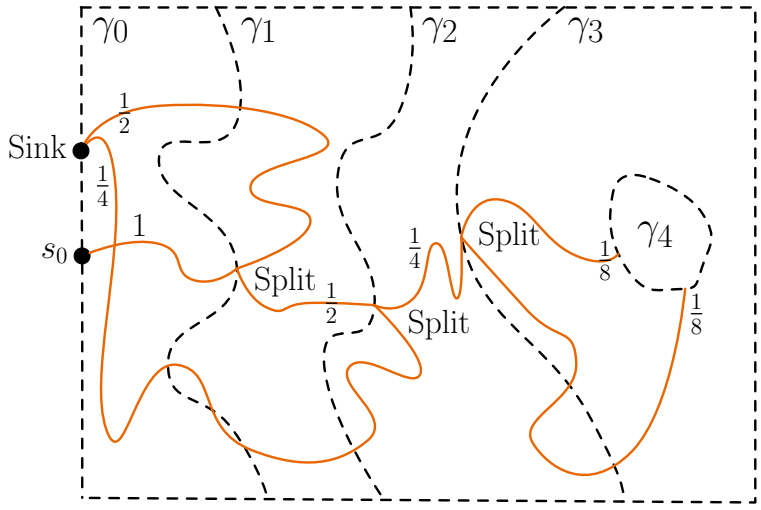



Figure 3.2: Evolution of trajectories using splitting

Figure 3.2 illustrates this idea. The sequence of embedded subsets is $\gamma_4 \subset \gamma_3 \subset \cdots \subset \gamma_0$. Starting from $s_0$, we want to compute the probability to reach $\gamma_4$ before hitting the sink state. Each time a trajectory enters a new subset for the first time, it is split into two trajectories.

**Example 5** *To illustrate the splitting method, a simple example can be built from a coin tossing problem. Suppose we want to estimate the probability for an unbiased coin to return head 25 times consecutively. This probability is* $\frac{1}{2^{25}} \approx 3\ 10^{-8}$. *Using Monte-Carlo simulation, the rare-event problem arises. This example can be formalized in a BDTMC where the state space*

*is* $\{0, 1, \ldots, 25, s_-\}$*, the initial state is* 0*, the accepting state is* 25*, and* $s_-$ *is the rejecting state. States* 25 *and* $s_-$ *are absorbing, each probability to go from states* $i$ *to state* $i+1$ *is* 0.5 *and the probability to go to* $s_-$ *is* 0.5*. The probability for a trajectory to reach state* 25 *is* $\frac{1}{2^{25}}$ *which is rare. A splitting technique can be used here. The embedded subsets are chosen as* $\gamma_0 = \{0, 1, \ldots, 25\}$ $\gamma_1 = \{5, 6, \ldots, 25\}$, $\gamma_2 = \{10, 11, \ldots, 25\}$,... $\gamma_5 = \{25\}$*. The number of replications (*$n_i$*) is constant and equals to* 50*.*

*In this setting, the probability* $p_1 = p_2 = \ldots p_5$ *can be computed and is equal to* $1 - \frac{1}{2^5} \approx 0.03$*. The probability of having at least one trajectory reaching* 25 *is crudely under-approximated by the probability that there is one trajectory reaching* $\gamma_{i+1}$ *from* $\gamma_i$ *for each* $1 \leq i \leq 5$ *:*

$$(1 - (p_1)^{50})^5 \approx 0.32$$

*The mean number of trajectories reaching the state* 25 *is* $(n_0 p_1)^5 \approx 9.3$*. These calculations show that using splitting, observing a trajectory reaching the target state is no longer a rare event and the mean number of trajectories to be generated is tractable. By comparison in plain Monte-Carlo, to obtain a probability of* 0.32 *to observe a trajectory reaching* 25*, a number of trajectories around* $12\ 10^6$ *is required.*

In a simplified setting, a simple expression for the variance of this estimator can be obtained. We make the assumption that for any subset of $\gamma_i$, the probability to reach $\gamma_{i+1}$ is independent of the state of entry in $\gamma_i$ and is independent of the trajectory history. This is equivalent to say that there is only one entry state for each subset and this state fully characterizes the stochastic process. In this setting, for all $i \leq k$, the random variable $H_i$ follows a binomial law with parameters $n_{i-1}H_{i-1}$ and $p_i$. The variance is computed as follows:

$$\begin{aligned}
\mathbb{V}(H_k) &= \mathbb{E}(\mathbb{V}(H_k|H_{k-1})) + \mathbb{V}(\mathbb{E}(H_k|H_{k-1})) \\
&= \mathbb{E}(n_{k-1}H_{k-1}p_k(1-p_k)) + \mathbb{V}(n_{k-1}H_{k-1}p_k) \\
&= n_0 p_1 n_1 p_2 \ldots n_{k-1} p_k (1-p_k) + n_{k-1}^2 p_k^2 \mathbb{V}(H_{k-1}) \\
&= \ldots \\
&= \sum_{i=1}^{k} n_i^2 p_{i+1}^2 \ldots n_{k-1}^2 p_k^2 n_0 p_1 \ldots n_{i-1} p_i (1-p_i) \\
&= p_1^2 \ldots p_k^2 n_0^2 \ldots n_{k-1}^2 \sum_{i=1}^{k} \frac{1-p_i}{n_0 p_1 \ldots n_{i-1} p_i}.
\end{aligned}$$

Thus

$$\mathbb{V}\left(\frac{H_k}{n_0 n_1 \ldots n_{k-1}}\right) = p^2 \sum_{i=1}^{k} \frac{1-p_i}{n_0 p_1 \ldots n_{i-1} p_i} \tag{3.1}$$

In the optimal case, for $0 < i < k$, $n_i$ are chosen such that $n_i p_{i+1} \approx 1$, thus the number of trajectories in each subset is roughly constant, the expression of the variance becomes: $p^2 \sum_{i=1}^{k}(1 - p_i)$. This function takes its minimal value for $p_1 = p_2 = p_k = \sqrt[k]{p}$ where it becomes $p^2 k(1 - \sqrt[k]{p})$, which converges to $-\ln(p)$ when $k$ goes to the infinity.

**Example 6** *The coin tossing problem fulfills the requirement for using Equation* (3.1)*. The variance is around* $1.53\ 10^{-15}$*. The number of trajectories required by the Monte-Carlo estimator to achieve the same variance is around* $19\ 10^6$*.*

The main difficulty in the splitting method is to choose the sequence of subsets. In general, an *importance function* [36] is defined on the states of the system and the sequence of subsets is defined as the sequence of sets of states where the importance function exceeds a threshold.

Another crucial parameter is the number of offsprings at each split. If it is too small no trajectory reaches the rare event and if it is too large the number of trajectories may become unmanageable. In an optimal setting, the number of offspring at each split is inversely proportional to the probability to reach the next subset. In this setting, the total number of trajectories in any subset is constant. This optimal setting is theoretical as the probability to reach the next subset is unknown. Different strategies exist to keep the number of trajectories large enough but still manageable, which dynamically compute the number of splits $n_i$. In this case, each trajectory is annotated by the number of siblings at each split.

Additionally to splitting trajectories on transitions coming closer to the goal, trajectories taking a transition going away from the goal may be killed, using a russian roulette, to reduce the number of paths to manage. It is also possible to get rid of the sequence of subsets and decide for each transition whether to split or kill the trajectory according to the evolution of the importance function by this transition. For example, the *RESTART* algorithm [37] is a splitting technique that aims at keeping the number of trajectories manageable and preventing death of all trajectories.

This method requires to deal with numerous parameters: the sequence of subsets, the number of splittings of the trajectories, etc. The number of parameters helps the user to find a good parametrization but in return makes the automation difficult. Moreover, variance reduction is not easy to guarantee theoretically.

As the system is unmodified by splitting methods, these methods can be used in a black box setting as long as it is possible to make a copy of the system. Such techniques are efficient when some expert knowledge on the system is available. In particular, to design the sequence of embedded subsets, one needs to know which part of the state space is close to the rare event. These methods work the best when the probability to reach

the rare event is uniformly distributed on the boundary of each set $\gamma_i$, like in the setting used for the computation of the variance. They have been successfully used in various settings, for example in the setting of particle transport (see for instance Chapter 10 of[78] ), in chemical physics [4], in telecommunication [3], in dependability [54], etc.

## 3.4 Importance Sampling

Importance sampling [40] is based on a different idea. It relies on the modification of the probability distribution in the system. This modification introduces a bias which will increase the probability that a rare event occurs. During the simulation of a trajectory, it is possible to keep track of the introduced bias with a so called *likelihood* ratio. By weighting the trajectories with their likelihood, it produces an unbiased estimator of the probability of the rare event. The difficulty is then to find a correct bias for the system. It is easy to prove that there exists an optimal modification of the system which computes the probability of the rare event with only one simulation, In this case, the estimator has a zero variance. But an erroneous modification of the system can, on the contrary, gives an estimator which is "worse" than the initial one. The efficiency of an importance sampling is often evaluated by looking at its variance: the smaller the variance is, the faster the convergence to the expected value will be.

In this thesis, we only consider importance sampling on DTMC. It is possible to define it for more general settings with arbitrary continuous distributions. In this case, the likelihood ratio is defined in terms of Radon-Nikodym derivative $d\mathbb{P}/d\tilde{\mathbb{P}}$ where $\mathbb{P}$ is the initial probability distribution and $\tilde{\mathbb{P}}$ is the biased one.

### 3.4.1 Definition

Recall that in a BDTMC, all trajectories starting from $s_0$ reach the state $s_+$ or state $s_-$ with probability 1. The problem we are focusing on in the following is that, "Given a BDTMC, what is the probability to reach $s_+$?". Formally, we introduce the random variable $V_{s_0}$ such that for all trajectories $\sigma = s_0, s_1, \ldots, s_n, s_\pm$ ending in $s_+$ or $s_-$ of a BDTMC,

$$V_{s_0} = \begin{cases} 1 & \text{if } \sigma \text{ ends in state } s_+ \\ 0 & \text{if } \sigma \text{ ends in state } s_-. \end{cases}$$

We are interested in estimating $\mathbb{E}(V_{s_0})$.

The formal definition of importance sampling is as follows:

**Definition 20 (Importance Sampling)**

Given a BDTMC $\mathcal{C} = (S, s_0, \mathbf{P}, s_-, s_+)$, and a biased transition probability matrix $\mathbf{P}'$ such that:

$$\forall s, t \in S,\ \mathbf{P}(s,t) > 0 \wedge t \neq s_- \Rightarrow \mathbf{P}'(s,t) > 0 \tag{3.2}$$

The random variable $W_{s_0}$ is defined such that for all trajectories $\sigma = s_0, s_1, \ldots, s_n, s_\pm$ ending in $s_+$ or $s_-$ of the biased BDTMC $(S, s_0, \mathbf{P}', s_-, s_+)$,

$$W_{s_0} = \begin{cases} \frac{\mathbf{P}(s_0,s_1)}{\mathbf{P}'(s_0,s_1)} \cdot \frac{\mathbf{P}(s_1,s_2)}{\mathbf{P}'(s_1,s_2)} \cdot \ldots \cdot \frac{\mathbf{P}(s_n,s_+)}{\mathbf{P}'(s_n,s_+)} & \text{if } \sigma \text{ ends in state } s_+ \\ 0 & \text{if } \sigma \text{ ends in state } s_- \end{cases}$$

A statistical method which estimates $W_{s_0}$ in order to estimate $V_{s_0}$ is called an importance sampling.

The value taken by the random variable $W_{s_0}$ when it reaches the state $s_+$ is the likelihood.

**Remark 5** *Condition 3.2 ensures that the introduced bias cannot remove transitions that have not $s_-$ as target. This is necessary to ensure that all trajectories in the original model are valid trajectories in the biased model.*

*This condition allows one to add new transitions in the bias distribution but trajectories that take such transitions in the bias model have a likelihood of zero and thus behave as trajectories reaching $s_-$ which reduces the probability to reach $s_+$.*

*Thus when one is building a biased model, only DTMC with the same underlying graph should be considered. This is not very restrictive as one can still choose all the probability transitions in this graph.*

The following proposition establishes the correctness of the method.

**Theorem 12**

Given a BDTMC $\mathcal{C} = (S, s_0, \mathbf{P}, s_-, s_+)$, and a biased transition probability matrix $\mathbf{P}'$, such that $\mathbf{P}'$ defines an importance sampling on $\mathcal{C}$ with associated random variable $W_{s_0}$ then

$$\mathbb{E}(W_{s_0}) = \mathbb{E}(V_{s_0}).$$

*Proof:*
In all states, the probability to reach $s_-$ or $s_+$ is equal to 1. Then thanks to Equation 8, the expected value of the random variable $V_s$ is the unique

solution of the following system of equations:

$$\begin{cases} \mathbb{E}(V_{s_-}) = 0 \\ \mathbb{E}(V_{s_+}) = 1 \\ \forall s \notin \{s_-, s_+\}\ \mathbb{E}(V_s) = \sum_{s' \neq s_-} \mathbf{P}(s, s')\mathbb{E}(V_{s'}) \end{cases} \tag{3.3}$$

We now write the corresponding system for $\mathbf{P}'$ with correction factor:

$$\begin{cases} \mathbb{E}(W_{s_-}) = 0 \\ \mathbb{E}(W_{s_+}) = 1 \\ \forall s \notin \{s_-, s_+\}\ \mathbb{E}(W_s) = \sum_{s' \neq s_- \wedge \mathbf{P}'(s,s') > 0} \mathbf{P}'(s, s') \left( \frac{\mathbf{P}(s, s')}{\mathbf{P}'(s, s')} \right) \mathbb{E}(W_{s'}) \end{cases}$$

Thanks to the restriction of Equation 3.2, the two systems are equal after simplification, and we have $\mathbb{E}(W_{s_0}) = \mathbb{E}(V_{s_0}) = p$.

□

A good choice of $\mathbf{P}'$ should reduce the variance of $W_{s_0}$ w.r.t. to the variance of $V_{s_0}$.

**Running example.** *As the tandem queue system is parametrized by the three probabilities $\lambda, \rho_1$ and $\rho_2$ an importance sampling can be built by changing these parameters to $\lambda', \rho'_1$ and $\rho'_2$. In this example we are interested in estimating the probability to reach a state with at least 5 clients before reaching the state where the system is empty. This probability is equal to 0.0089. An importance sampling is built by increasing the value of the probability of an incoming client $\lambda'$ and reducing the probability of a client leaving the system $\rho'_2$. Figure 3.3 shows the evolution of the variance when $\lambda'$ increases.*

*We observe that starting from $\lambda' = \lambda = 0.1$, the variance decreases when $\lambda'$ increases until a minimal value is reached, then the variance increases. This shows the difficulty to find a good importance sampling where the variance is significantly smaller than in the initial model.*

### 3.4.2 Zero Variance Importance Sampling

The following proposition shows that for any BDTMC, there exists a matrix $\mathbf{P}'$ which leads to a null variance. We denote the probability to reach $s_+$ starting from $s$ by $\mu(s)$.

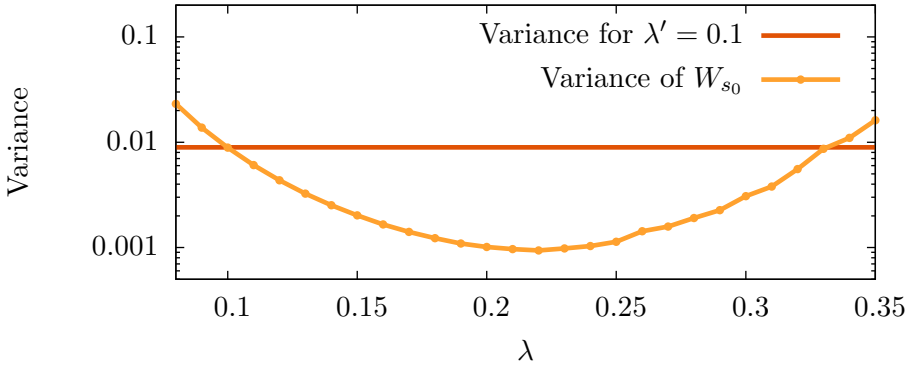



Figure 3.3: Evolution of the variance of $W_{s_0}$ for the tandem queue system

**Theorem 13**

Let $\mathbf{P}'$ be defined by:

- For all $s$ such that $\mu(s) \neq 0$, $\mathbf{P}'(s, s') = \frac{\mu(s')}{\mu(s)}\mathbf{P}(s, s')$
- For all $s$ such that $\mu(s) = 0$, $\mathbf{P}'(s, s') = \mathbf{P}(s, s')$

Then for all $s$, we have $\mathbb{V}(W_s) = 0$.

*Proof:*
If $\mu(s) = 0$ then all trajectories starting in $s$ end in $s_-$. Therefore the variance is null.

If $\mu(s) \neq 0$, thanks to the equation:

$$\mu(s) = \sum_{s'|\mu(s')>0} \mathbf{P}(s, s')\mu(s')$$

$\mathbf{P}'(s, -)$ is a distribution. A trajectory starting from a state $s$ with $\mu(s) > 0$ only visits states $s'$ with $\mu(s') > 0$, so it ends in $s_+$. Denoting by $s = u_0, \ldots, u_l = s_+$ such a trajectory, the value $L$ is equal to

$$\frac{\mu(u_0)}{\mu(u_1)}\frac{\mu(u_1)}{\mu(u_2)} \cdots \frac{\mu(u_{l-1})}{\mu(u_l)} = \mu(s).$$

□

This result has *a priori* no practical application since it requires the knowledge of $\mu$ for all states, whereas we only want to estimate $\mu(s_0)$. Several importance sampling methods including the one presented in this thesis are based on approximating the vector $\mu$ and plugging this approximation into the zero-variance importance sampling biased matrix.

### 3.4.3 Infinite Variance Importance Sampling

Unfortunately a poor choice of $\mathbf{P}'$ can lead to an importance sampling with infinite variance. Let us illustrate this on an example.

Let us define a BDTMC $\mathcal{C}$ with three states $s_0$, $s_+$ and $s_-$, where $s_0$ is the initial state, $s_+$ is an absorbing state and $s_-$ is not reachable from $s_0$. Let $\mathbf{P}$ be its transition probability matrix. $\mathbf{P}(s_0, s_0) = \frac{1}{2} = \mathbf{P}(s_0, s_+)$, $\mathbf{P}(s_+, s_+) = 1$ and $\mathbf{P}(s_+, s_0) = 0$. We are interested in the probability to reach $s_+$, this probability is indeed equal to 1.

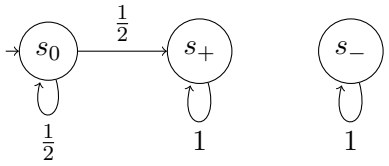



Figure 3.4: Markov Chain illustrating the infinite variance importance sampling

We can use an importance sampling on this Markov chain by defining a new transition probability matrix $\mathbf{P}'$, $\mathbf{P}'(s_0, s_0) = \frac{1}{4}$ , $\mathbf{P}'(s_0, s_+) = \frac{3}{4}$ and $\mathbf{P}' = \mathbf{P}$ for other transitions. Let $W_{s_0}$ be the random variable associated to this importance sampling. Let $L_n$ be the likelihood of a path reaching $s_+$ in $n$ steps.

$$L_n = \left( \prod_{i=1}^{n-1} \frac{\mathbf{P}(s_0, s_0)}{\mathbf{P}'(s_0, s_0)} \right) \frac{\mathbf{P}(s_0, s_+)}{\mathbf{P}'(s_0, s_+)} = \left( \prod_{i=1}^{n-1} \frac{1}{2}\frac{4}{1} \right) \frac{1}{2}\frac{4}{3} = \frac{2^n}{3}.$$

The probability of such a path is $p'_n$

$$p'_n = \left( \prod_{i=1}^{n-1} \mathbf{P}'(s_0, s_0) \right) \mathbf{P}'(s_0, s_+) = \left( \prod_{i=1}^{n-1} \frac{1}{4} \right) \frac{3}{4} = \frac{3}{4^n}.$$

We compute the variance of the random variable $W_{s_0}$.

$$\mathbb{V}(W_{s_0}) = \mathbb{E}(W_{s_0}^2) - \mathbb{E}^2(W_{s_0}) = \left( \sum_{n \geq 1} L_n^2 p_n \right) - 1$$

$$\mathbb{V}(W_{s_0}) = \left( \sum_{n \geq 1} \left( \frac{2^n}{3} \right)^2 \frac{3}{4^n} \right) - 1 = \left( \sum_{n \geq 1} \frac{1}{3} \right) - 1$$

As this series does not converge the variance of $W_{s_0}$ is infinite.

This example shows that a poor choice of importance sampling may lead to an infinite variance estimator. In this case, there is no guarantee that the estimator converges or even worse it could seem to converge to a value which is not the probability that we want to compute.

### 3.4.4 Distribution of $W_{s_0}$

The major drawback of importance sampling is that the random variable $V_{s_0}$ which follows a Bernoulli law is replaced by the random variable $W_{s_0}$ whose distribution is unknown. This is a problem for computing a confidence interval. Indeed all methods to compute a confidence interval rely either on the range of values taken by the random variable or on its variance. When the distribution is unknown there is no bound on the values taken by the random variable and the variance may be difficult to estimate.

When importance sampling is used, trajectories may use transitions whose probability is smaller in the biased model than in the initial one. Such transitions are unlikely but carry a large likelihood. The case of infinite variance is the worst instance of this phenomenon. It may happen with high probability that after a consequent number of simulations no such trajectories have been observed, thus the expected value of $W_{s_0}$ is underestimated. Moreover as the variance of $W_{s_0}$ is also underestimated, confidence intervals computed to frame $\mathbb{E}(W_{s_0})$ will be wrong. This problem is well known and frequently happens. Results of an importance sampling algorithm are usually taken as underestimations of the probability of interest.

**Running example.** *In Figure 3.3, an estimation of the variance of the random variable $W_{s_0}$ is plotted, the analysis can be enhanced by plotting the probability density function of the random variable $W_{s_0}$ for several values of $\lambda'$. Figure 3.5 shows such a plot for the tandem queue system. For $\lambda' = \lambda$, $W_{s_0} = V_{s_0}$ and thus it follows a Bernoulli law. For other values of $\lambda'$, the probability density function is continuous and when $\lambda'$ increases, the probability for a trajectory to have a large likelihood drastically increases.*

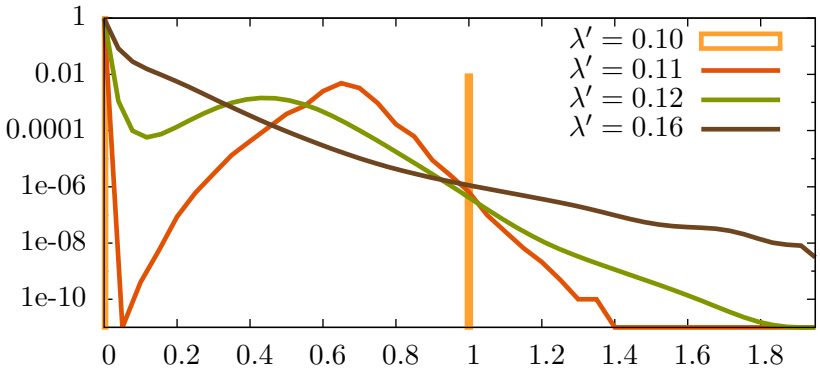



Figure 3.5: PDF of $W_{s_0}$ for increasing values of $\lambda'$

### 3.4.5 Efficiency of Importance sampling

To assert the quality of importance sampling, classical approaches consider that the model is parametrized by $\varepsilon$ and that the probability of the event of interest converges to 0 as $\varepsilon$ goes to 0. Then the behaviors of the importance sampling for the rare event is studied for value of $\varepsilon$ close to zero. One measure is the *relative error*:

**Definition 21**

The relative error of an estimator is half the width of the Gaussian confidence interval divided by the mean value.

An unbiased estimator $Z$ is said to have *Bounded Relative Error* (BRE) if the relative error remains bounded when $\varepsilon$ tends to 0. It is equivalent to say that the ratio $\frac{\mathbb{E}(Z^2)}{\mathbb{E}^2(Z)}$ is bounded. This property is very strong and difficult to check in practice. A weaker property is the asymptotic optimality. An unbiased estimator $Z$ is *asymptotically optimal* with respect to $\varepsilon$ if

$$\lim_{\varepsilon \to 0} \frac{\log \mathbb{E}(Z^2)}{\log \mathbb{E}(Z)} = 2.$$

These two efficiency properties can be strengthened to take into account higher moments of the estimator [15].

### 3.4.6 State of the Art on Importance Sampling

There are two main categories of importance sampling: state-dependent and state-independent importance samplings, depending of the level on which the definition of the importance sampling takes place.

#### State-dependent importance sampling

The state-dependent importance sampling is defined at the level of DTMC (or CTMC in a more general setting). If the DTMC is huge or infinite, it becomes difficult to manage a representation of the new distribution. Moreover it is unfeasible to manually define the biased transition probability matrix as there are as many parameters as transitions in the DTMC. Several works rely on decomposing transitions of the DTMC into transitions that bring the system closer to the rare event and those that make the system further from it. In [82], a threshold $\gamma$ is chosen, then in each state of the Markov chain, outgoing transitions are weighted such that, the sum of probabilities of all transitions that bring the system closer to the rare event is at least $1 - \gamma$. The partition of transitions in the set of transitions that bring the system closer to the rare event and the set of transitions that bring it further from

the rare event is manually defined. By making $\gamma$ small enough, the event of interest is no longer a rare event in the biased model.

Approximations of the zero-variance importance sampling by approximating the probability to reach the target state $\mu$ have been extensively studied for example in [55]. In [66], the zero-variance importance sampling is approximated by using most typical trajectories to reach the rare event. This method is extended in [75] to stochastic Petri nets by analyzing the structure of the underlying Markov chain.

Adaptive methods for learning the vector $\mu$ during the simulation to produce better and better approximations have also been studied [42]. The algorithm works as follows

1. Choose an initial probability vector $\mu_1$. Set $i = 1$.
2. Build a transition probability matrix $\mathbf{P}_i$ from the vector $\mu_i$ using the optimal importance sampling formula which is normalized to obtain a stochastic matrix:
$$\mathbf{P}_i(s, s') = \frac{1}{\sum_{s'' \in S} \frac{\mu_i(s'')}{\mu_i(s)} \mathbf{P}(s, s'')} \frac{\mu_i(s')}{\mu_i(s)} \mathbf{P}(s, s')$$
3. For each state $s$, estimate $\mu_{i+1}(s)$, the probability to reach $s_+$ from $s$ using the importance sampling induced by $\mathbf{P}_i$.
4. Set $i = i + 1$. Go back to step (2) until a stopping criterion is reached.

The process iterates until the obtained importance sampling is efficient enough. Under some regularity assumptions, it is proven in [61] that the sequence of vectors $(\mu_i)$ converges exponentially toward $\mu$.

### State-independent importance sampling

The state-independent importance sampling is an importance sampling defined on a high-level model using parameters. In high-level models, transitions of the Markov chain are not defined independently. All transitions of the Markov chain depend on a small number of parameters. For instance in the tandem queue example, there are three parameters which are the rate of arrival and the rate of the services. In the corresponding Markov chain, several transitions have the same rate. State-independent importance sampling consists in modifying the probabilities or rate of transitions at a high level and not for each state of the Markov chain independently.

Importance sampling where the probability of transition depends on the states but only relies on a few parameters can also be called state independent. They have the same properties as a real state-independent importance sampling. For example, consider a queue with an infinite server policy, that is the probability of a client to be served increases linearly with

the number of waiting clients. This system is only parametrized by the probability to serve a single client. An importance sampling in this setting can only change this parameter and modify all transition probabilities in a different way. However as all transition probabilities depend on the same parameter, the biasing of transition probabilities does not really depend on each state.

As the number of parameters is usually small, the associated importance sampling is also easy to define. Some of these methods are:

- Trying several values of each parameter and searching experimentally for good values. This is feasible only if the number of parameters is very small and their values are bound.
- The parameters may have some meaning to the modeler who can use his knowledge on the system to design the biased transition probability matrix.
- Optimization methods can be used to find the best parameter values that minimize the variance. Among optimization methods, the cross-entropy method is detailed below.

Unfortunately if no values of parameters characterize the optimal importance sampling which is frequent, there is no guarantee to find an efficient importance sampling. Indeed, the number of degrees of freedom is equal to the number of parameters whereas in the case of state-dependent importance sampling, the number of degree of freedom is the number of transitions in the Markov chain. The problem of finding an efficient importance sampling then relies on finding a correct parametrization of the system, and, when using iterative optimizing methods, on finding suitable initial parameters.

**Running example.** *We illustrate the limit of state-independent importance sampling in the tandem queue system which has been heavily studied. By choosing as parameters the three probabilities $\lambda, \rho_1$ and $\rho_2$, the optimal state-independent importance sampling is obtained by exchanging the value of $\lambda$ with the largest value among $\rho_1$ and $\rho_2$ [71]. It is referred in the following as PW importance sampling. This importance sampling is induced by results in large deviation theory. In this setting, an importance sampling is asymptotically efficient if the number of simulations required to obtain a given relative error grows less than exponentially with the overflow level $N$. In [80], it is shown that only PW importance sampling may be asymptotically efficient. In [39], it is proven that for some values of the parameters, PW importance sampling is not asymptotically efficient and its variance may even be infinite [24]. For some values of the parameters of the importance sampling, there is no efficient state-independent importance sampling whereas there are efficient state-dependent ones [29] which can be efficiently computed.*

### Cross-entropy method

Among optimization methods, the cross-entropy method has recently received a lot of attention [79, 22, 52]. In practice it relies on a parametrization of the model, even if it is defined at the level of the Markov chain. Among all the importance samplings defined by changing the value of the parameters, it computes the one that minimizes the distance to the optimal importance sampling using the *cross-entropy distance* [62]. Furthermore, this minimization is done iteratively by successive simulations of the model. An introduction to the cross-entropy method can be found in [25] where it is shown that this method can be used to solve also classical optimization problems unrelated to rare events, like the traveling salesman problem.

More formally in the context of BDTMC with an initial transition probability matrix $\mathbf{P}$, let call $f_{\mathbf{P}}$ the probability density function over paths of the Markov chain: $f_{\mathbf{P}}(s_0, s_1, s_2 \ldots, s_n) = \mathbf{P}(s_0, s_1)\mathbf{P}(s_1, s_2) \ldots \mathbf{P}(s_{n-1}, s_n)$. Let $\Omega$ be the set of all finite paths ending in $s_+$ or $s_-$. Recall that $X(\sigma)$ is a random variable which value is 1 for path ending in $s_+$ and zero otherwise. The probability density function of the zero-variance importance sampling is $f^*(\sigma) = \frac{f_{\mathbf{P}}(\sigma)X(\sigma)}{\mathbb{E}(X)}$. Given two probability transition matrices $\mathbf{P}_1$ and $\mathbf{P}_2$, the cross-entropy distance is defined as:

$$CE(f_{\mathbf{P}_1}, f_{\mathbf{P}_2}) = \int_{\sigma\in\Omega} f_{\mathbf{P}_1}(\sigma) \log \frac{f_{\mathbf{P}_1}(\sigma)}{f_{\mathbf{P}_2}(\sigma)} d\sigma.$$

This quantity is not formally a distance but is sufficient for our purposes. The idea of this method is to find a change of measure that minimizes the cross-entropy to the zero-variance importance sampling:

$$\begin{aligned} CE(f^*, f_{\mathbf{P}'}) &= \int_{\sigma\in\Omega} f^*(\sigma) \log \frac{f^*(\sigma)}{f_{\mathbf{P}'}(\sigma)} d\sigma \\ &= \int_{\sigma\in\Omega} (f^*(\sigma) \log f^*(\sigma) - f^*(\sigma) log(f_{\mathbf{P}'}(\sigma))) d\sigma \\ &= \int_{\sigma\in\Omega} f^*(\sigma) \log f^*(\sigma) d\sigma - \int_{\sigma\in\Omega} \frac{X(\sigma) f_{\mathbf{P}}(\sigma)}{\mathbb{E}(X)} \log(f_{\mathbf{P}'}(\sigma)) d\sigma \end{aligned}$$

By keeping only terms depending of $\mathbf{P}'$, minimizing the cross-entropy is equivalent to maximizing the right term:

$$\max_{\mathbf{P}'} \int_{\sigma\in\Omega} X(\sigma) f_{\mathbf{P}}(\sigma) \log(f_{\mathbf{P}'}(\sigma)) d\sigma$$

which can be rewritten as:

$$\max_{\mathbf{P}'} \int_{\sigma\in\Omega} f_{\mathbf{P}'}(\sigma) L_{\mathbf{P}'}(\sigma) X(\sigma) \log(f_{\mathbf{P}'}(\sigma)) d\sigma \tag{3.4}$$

where $L_{\mathbf{P}'} = \frac{f_{\mathbf{P}}}{f_{\mathbf{P}'}}$ is the likelihood corresponding to the change of measure with transition probability matrix $\mathbf{P}'$.

Using Equation (3.4), one can define an iterative algorithm as follows:

1. Choose an initial transition probability matrix $\mathbf{P}_1$. Set $i = 1$.
2. Simulate $n_i$ trajectories of the Markov chain using $\mathbf{P}_i$ as transition probability matrix, yielding a set of trajectories $\{\sigma_k\}_{k=1}^{n_i}$
3. Compute $\mathbf{P}_{i+1} = \max_{\mathbf{P}'} \sum_{k=1}^{n_i} L_{\mathbf{P}_i}(\sigma_k) X(\sigma_k) \log(f_{\mathbf{P}'}(\sigma_k))$
4. Set $i = i + 1$. Go back to step (2) until a stopping criterion is reached.

Suppose the system is parametrized by $\varepsilon < 1$ and that $\mathbb{E}(X)$ goes to 0 as $\varepsilon$ goes to 0. Let $\mathbf{P}^{ce}$ be the matrix obtained by the last iteration of the algorithm. Under the additional strong constraint that there exist two finite constants $K_1$ and $K_2$ such that

$$K_1 \leq CE(f^*, f_{\mathbf{P}^{ce}}) \leq K_2$$

for any $\varepsilon$, then the cross-entropy method is asymptotically optimal [77]

In order to implement such a method, a parametrization of the transition probability matrix is often necessary to make the optimization step tractable. Remark that in general the result of asymptotical optimality does not hold with a parametrization. As each intermediate step produces an importance sampling, the algorithm can be stopped as soon as the produced importance sampling is efficient enough. The initial transition probability matrix is crucial as the followings steps require that some trajectories reach $s_+$. Here the cross-entropy method is described in the setting of DTMC but it is also used in a more general setting.

## 3.5 Conclusion

The reliable estimation of a rare-event probability is crucial for the analysis of some stochastic systems. However Monte-Carlo simulations are ineffective to deal with this kind of probabilities. Methods that deal with the rare-event problem in Monte Carlo simulation have been intensively studied, mostly in some specific domain like computational physics. Recently, a new interest on statistical model checking stimulates research for methods dealing with the rare-event problem in more general settings.

The existing methods, mainly importance sampling and splitting, cannot produce confidence intervals for the value of interest except by using asymptotical results whose validity is difficult to assert in practice. Moreover all these methods are parametrized and often require the modeler to guess these parameters.

# Part II

# Theoretical Contributions

# Chapter 4

# Guaranteed Variance Reduction

## 4.1 Introduction

In a reachability probability problem, one wants to compute the probability $p$ to reach a final state $s_+$, from the initial state $s_0$ of a given model. When the size of the model becomes too large for numerical analysis and when the probability of interest $p$ is too small for direct statistical simulation, rare event acceleration methods become necessary. A possible approach to tackle this problem is to build an importance sampling scheme for which a statistical simulation becomes possible. In order to be useful, this importance sampling has to be built carefully to ensure a reduction of the variance.

In this chapter, we propose a method to design such efficient importance sampling. This method relies on the construction of an abstraction of the initial model, on which the distribution of the importance sampling is computed by numerical analysis. This abstraction is performed by lumping some states and possibly modifying distributions in order to obtain a smaller model with a similar behavior but more manageable for numerical computations. We call this model the *reduced model*.

In order to ensure that the variance is reduced within this approach, we define sufficient conditions on the reduced model in Section 4.2, together with methods to check that they are satisfied by a given reduced model. In the three following sections, we use such reduced models to define importance samplings allowing efficient estimations of either time-unbounded or time-bounded reachability probabilities. Section 4.3 deals with the case of unconstrained reachability property. Section 4.4 studies the time-bounded reachability property of a discrete time model. Section 4.5 analyzes the time-bounded reachability property of continuous time models by adapting the results obtained in the time-discrete case. Finally, Section 4.6 describes how model checking problems can be transformed into reachability problems

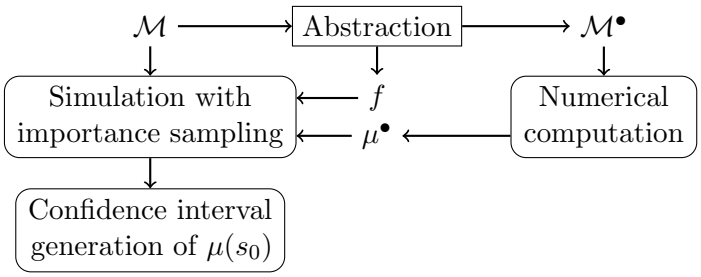



Figure 4.1: Schematic representation of the method

which expands the range of application of this method.

## 4.2 A New Approach for Importance Sampling

This section formalizes the idea of abstracting a model in a way that allows one to build an importance sampling with sufficient property to reduce the variance. We present two kinds of hypotheses on the construction ensuring importance samplings satisfying different interesting properties. These hypotheses are the same for models taking as semantics a DTMC or a CTMC and for time-bounded or unbounded reachability problems. In the following we consider only DTMCs. Similar definitions are given in Section 4.5 for CTMCs.

### 4.2.1 Principle of the Method

In this section, we consider a model $\mathcal{M}$ taking as semantics a DTMC with two absorbing states $s_+$ and $s_-$ reached with probability one. Our goal is to efficiently estimate the probability to reach $s_+$ with statistical methods whereas this probability is small.

The principle of the method is summarized in Figure 4.1. Roughly speaking, the computation is split in two parts. A crude approximation of the probability of interest is computed with a numerical computation, then using this approximation, a simulation with importance sampling computes a precise value. The numerical computation is performed on an abstraction $\mathcal{M}^\bullet$ of the model $\mathcal{M}$ using a function $f$ that maps states of $\mathcal{M}$ to states of $\mathcal{M}^\bullet$. As the transition probability matrix $\mathbf{P}'$ of the optimal importance sampling(see Section 3.4.2) is defined by

$$\mathbf{P}'(s,s') = \frac{\mu(s')}{\mu(s)}\mathbf{P}(s,s'),$$

where $\mu(s)$ is the probability to reach the state $s_+$ from $s$, we define in a

similar way the matrix by

$$\mathbf{P}'(s,s') = \frac{\mu^\bullet(f(s'))}{\mu^\bullet(f(s))}\mathbf{P}(s,s'),$$

where the probability vector $\mu$ has been replaced by $\mu^\bullet \circ f$. This vector $\mu^\bullet$ is defined as a reachability probabilities vector in an abstraction of the model $\mathcal{M}$ *i.e. the reduced model* $\mathcal{M}^\bullet$. The function $f$ maps states of the original model to states of the reduced model. Such a function is called a *reduction function*. Figure 4.2 illustrates the behavior of the function $f$. The aim is that the likelihood under this importance sampling will be close to $\mu^\bullet(f(s_0))$, whereas in the optimal case the likelihood is equal to $\mu(s_0)$. Moreover $\mu^\bullet(s^\bullet)$ is a reachability probability in a smaller DTMC, it can be computed numerically. However, some adaptations are required in $\mathbf{P}'$ due to the fact that $\mu^\bullet \circ f \neq \mu$, to obtain a transition probability matrix. These adaptations lead to different important samplings which are defined in Sections 4.3, 4.4 and 4.5.

Reduced models must achieve three possibly conflicting goals:

1. The size of the reduced model has to be significantly smaller than the size of the initial one in order to perform a numerical analysis on it.

2. The reachability probabilities of the event of interest with and without reduction ($\mu^\bullet(f(s_0))$ and $\mu(s_0)$) should be no more different than about three orders of magnitude. Otherwise, the importance sampling obtained from the reduced model will be useless because the event of interest will still be rare.

3. The random variable $W_{s_0}$ obtained from the importance sampling we produce should have a small variance. In particular, we want to ensure that we do not fall in the case described in Section 3.4.3 where the variance is infinite.

An example of reduction for the example of tandem queues is given in the next section. Chapter 5 and 6.7 present and discuss several examples of reduced models.

### 4.2.2 Reduced Model

**Definition 22 (Reduction of Model)**

Let $\mathcal{C}$ be a BDTMC with two absorbing states $s_+$ and $s_-$. A DTMC $\mathcal{C}^\bullet$ is called a *reduction* of $\mathcal{C}$ by a function $f$ that maps $S$ to $S^\bullet$, the state space of $\mathcal{C}^\bullet$, if, denoting $s^\bullet_- = f(s_-)$ and $s^\bullet_+ = f(s_+)$, the following assertions are satisfied:

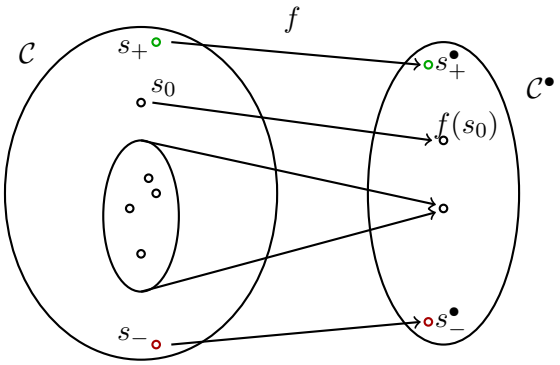



Figure 4.2: Definition of the reduced model

- $f^{-1}(s_-^\bullet) = \{s_-\}$
- Let $s^\bullet \in S^\bullet$, and denote by $\mu^\bullet(s^\bullet)$, the probability to reach $s_+^\bullet$ starting from $s^\bullet$, then for all $s \in S$,

$$\mu^\bullet(f(s)) = 0 \Rightarrow \mu(s) = 0,$$

.

The first assertion entails that only the state $s_-$ is mapped to $s_-^\bullet$. The second one entails that if a state is mapped to a state that cannot reach $s_+^\bullet$, then it also cannot reach $s_+$. Observe that $\mathcal{C}^\bullet$ is a BDTMC. These assumptions ensure that trajectories in the model can be mapped to trajectories in the reduced model as transitions between state are preserved. This is necessary for the existence of the objects built in the next section. The efficiency of the method relies on additional properties of this reduced model.

In the following, everything marked with a "$\bullet$" like $s_+^\bullet$ is related to the reduced model.

**Running example.** *Let us recall the example of the tandem queues. Figure 4.3 shows on the upper left part a SPN representing the tandem queues. The state of the system is represented by the pair $(n_1, n_2)$ of the number of tokens in each queue. The property of interest is that a state where the sum of the number of tokens in the two queues exceeds $N$ is reached before the initial state where the system is empty. This property will later be referred as* "Global overflow in tandem queues" *and can be expressed in LTL as* $\mathbf{X}\,(\, n_1 + n_2 > 0\, \mathbf{U}\, n_1 + n_2 \geq N\,)$.

*On the bottom left part of Figure 4.3, the DTMC corresponding to this model where all states where $n_1 + n_2 \geq N$ have been lumped together into an absorbing state $s_+$ and the state where $n_1 = n_2 = 0$ have been made absorbing*

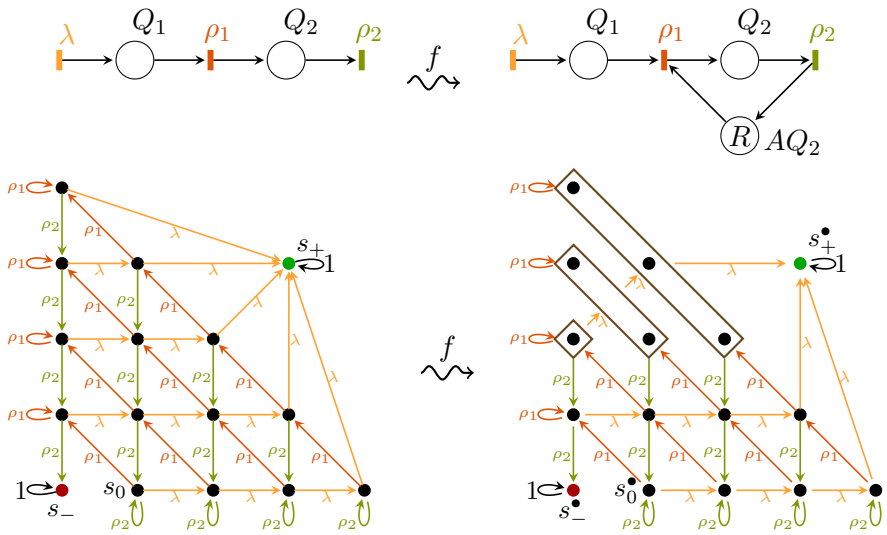



Figure 4.3: Reduction of the tandem queues as an SPN and as a DTMC.

*and rename $s_-$. This DTMC is a BDTMC because the only two BSCCs are singletons $\{s_-\}$ and $\{s_+\}$.*

*The reduced DTMC $\mathcal{C}^\bullet$ is obtained from the original model by applying the following function to the state space,*

$$f(n_1, n_2) = \begin{cases} (n_1, n_2) & \text{if } n_2 \leq R \\ (n_1 + n_2 - R, R) & \text{otherwise} \end{cases}$$

*This reduced model is shown on the bottom right of Figure 4.3. We see that the function $f$ lumps together all states where $n_1 + n_2$ is the same when $n_2 \geq R$. The intuition behind this reduction is to forbid clients in the first queue to enter the second one, as soon as there are $R$ clients in the second queue. This increases the probability of a global overflow. The corresponding SPN is shown on the upper right of Figure 4.3.*

Constraints can be added on the reduced model to bound the random variable $W_{s_0}$. In this case confidence intervals build from Chernoff-Hoeffding bounds can be computed. Moreover, if the distribution of $W_{s_0}$ is bounded and known up to some parameters, tighter confidence intervals can be constructed. In the next section, we define properties of reduced models that will be used to produce an important sampling yielding distribution of the likelihood which are bivaluated.

### 4.2.3 Guaranteed Variance Reduction

In Section 3.4, it was shown that an importance sampling should increase the probability of transitions toward the goal. This is reflected by the fact

that the function $f$ should ensure that $\mu(s_0) \leq \mu^\bullet(f(s_0))$ and more generally for an arbitrary state that $\mu(s) \leq \mu^\bullet(f(s))$. This can be used either in a weak sense, as in most of the states this condition should hold, or in a strong sense, as in all states this condition should hold.

At the same time, we will see in the next section, that when we use reduced models to build an importance sampling, the probability to reach the state $s_+$ from $s_0$ under the importance sampling is smaller than $\mu(s_0)/\mu^\bullet(f(s_0))$ thus $\mu^\bullet(f(s_0))$ should not be too big compared to $\mu(s_0)$. These two constraints highlight difficulties to produce suitable reduced models as the reduced model should increase the probability to reach the goal but not too much. We study here the strong case where:

$$\forall s,\ \mu(s) \leq \mu^\bullet(f(s)). \tag{4.1}$$

We want to substitute to this condition a sufficient structural one in order to check it by an analysis of the model and not to evaluate it on each state by numerical computation of $\mu$.

Let us look at a particular case. Suppose that the Markov chain $\mathcal{C}$ does not have any cycle such that it induces a topological order. The condition holds for the states $s_+$ and $s_-$. An induction over the states of $S$ starting from $s_+$ and $s_-$ following the reverse topological order may be done, the induction hypothesis is:

$$\mu(s) = \sum_{s' \in S} \mathbf{P}(s, s') \cdot \mu(s') \leq \sum_{s' \in S} \mathbf{P}(s, s') \cdot \mu^\bullet(f(s')).$$

In order to conclude that $\mu(s) \leq \mu^\bullet(f(s))$, the additional hypothesis

$$\forall s \in S,\ \sum_{s' \in S} \mathbf{P}(s, s') \cdot \mu^\bullet(f(s')) \leq \mu^\bullet(f(s))$$

is required

Using the additional hypothesis of this particular case, we define reduction with guaranteed variance in the general case as follows:

**Definition 23 (Reduction with Guaranteed Variance)**

Let $\mathcal{C}$ be a BDTMC and $\mathcal{C}^\bullet$ a reduction of $\mathcal{C}$ by $f$. The DTMC $\mathcal{C}^\bullet$ is a *reduction with guaranteed variance* by $f$ if, for all $s \in S$ such that $\mu^\bullet(f(s)) > 0$, it holds that:

$$\sum_{s' \in S} \mathbf{P}(s, s') \cdot \mu^\bullet(f(s')) \leq \mu^\bullet(f(s)). \tag{4.2}$$

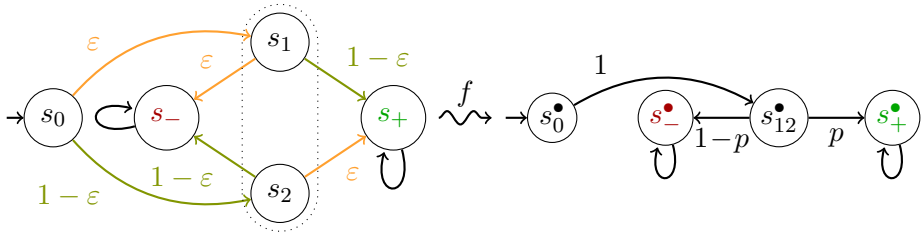



Figure 4.4: Example of reduced models

**Remark 6** *This condition may be difficult to ensure when building a reduced model. Figure 4.4 illustrates this difficulty. Suppose that $\varepsilon$ is a tiny probability and that in order to build the reduced model we decide to lump together states $s_1$ and $s_2$ in a new state $s_{12}^\bullet$. We need to choose the probability $p$ to go from $s_{12}^\bullet$ to $s_+^\bullet$. One can check that $\mu(s_0) = 2\varepsilon(1-\varepsilon)$ and $\mu^\bullet(s_{12}) = p$. The definition of reduced models only implies that $p > 0$. The principle that the reduced model should increase the probability to reach $s_+$ implies that $p > 2\varepsilon(1-\varepsilon)$. Equation (4.2) requires additional constraints on $p$. If we use this equation on state $s_1$, we have that*

$$\boldsymbol{P}(s_1, s_+) \cdot \mu^\bullet(f(s_+)) + \boldsymbol{P}(s_1, s_-) \cdot \mu^\bullet(f(s_-)) = 1 - \varepsilon \leq p,$$

*thus $\mu^\bullet(f(s_0)) \geq 1 - \varepsilon$. If $p = 1 - \varepsilon$, the probability for the simulation to reach $s_+$ under the importance sampling induced by the reduced model is $2\varepsilon$. For a small value of $\varepsilon$, say $10^{-9}$, this probability is still a rare event.*

*This example illustrates that a reduced model should not lump together states with very different probabilities to reach the state $s_+$. It is straightforward in this example that states $s_1$ and $s_2$ have very different behaviors but in more realistic settings, we have to merge a large (or even infinite) number of states together, and some of them may have different behaviors.*

**Running example.** *In the tandem queues example, the reduced model is a reduction with guaranteed variance. As the proof requires additional tools, we only provide some intuition here. Aggregated states in the reduced model have the same behaviors when there are less than $R$ clients in the second queue ($n_2 \leq R$). In states where $n_2 \geq R$, the transition $\rho_1$ is replaced by a self-loop which prevents the system from emptying too fast. This difference between the two models slows down clients in the reduced model and it increases their number. Thus $\mu(s) \leq \mu^\bullet(f(s))$ in all states.*

### 4.2.4 Structural Guarantee

Unfortunately, Equation (4.2) is difficult to check. The numerical verification that this equation holds is a two-step procedure. The first step is to compute

$\mu^\bullet$ for every state of $\mathcal{C}^\bullet$. This step is required later for computing the importance sampling and is not problematic if the size of $\mathcal{C}^\bullet$ is sufficiently small which is already supposed. The second step is to check that for any state in $\mathcal{C}$, the equation holds. This supposes to enumerate all states of $\mathcal{C}$. As the size of $\mathcal{C}$ can be huge, this is in general unfeasible. Moreover, most models have scaling parameters (like $N$ or $R$ for the tandem example) and we want to prove that the property holds for any value of these parameters.

For these reasons, we define a stronger structural requirement for the reduction which will ensure that Equation (4.2) is satisfied by construction. This is expressed in the next proposition using enriched BDTMC instead of BDTMC for the sake of clarity.

**Proposition 1**

Let $\mathcal{C}$ be an enriched BDTMC, $\mathcal{C}^\bullet$ be a reduction of $\mathcal{C}$ by $f$. Assume there exists a function $g : S \times E \to E^\bullet$ such that for all $s \in S$, $e \mapsto g(s,e)$ is an injection which maps each transition of the model to a transition in the reduced model, such that if:
$\forall s \in S,\ \forall e \in E$ :

1. $p(s,e) = p^\bullet\big(f(s), g(s,e)\big)$
2. $p(s,e) > 0 \ \Rightarrow\ \mu^\bullet\big(f(\delta(s,e))\big) \leq \mu^\bullet\big(\delta^\bullet(f(s), g(s,e))\big)$

then $\mathcal{C}^\bullet$ is a reduction of $\mathcal{C}$ by $f$ with guaranteed variance.

*Proof:*
For all $s \in S$, starting from the left hand side of Equation (4.2):

$$
\begin{aligned}
&\sum_{s' \in S} \mathbf{P}(s,s') \cdot \mu^\bullet(f(s')) \\
&\qquad\qquad \text{Using the definition of } \mathbf{P} \text{ in enriched DTMC} \\
&= \sum_{e \in E} p(s,e) \cdot \mu^\bullet(f(\delta(s,e))) \\
&\qquad\qquad \text{Using hypothesis 1} \\
&= \sum_{e \in E} p^\bullet(f(s), g(s,e)) \cdot \mu^\bullet(f(\delta(s,e))) \\
&\qquad\qquad \text{Using hypothesis 2} \\
&\leq \sum_{e \in E} p^\bullet(f(s), g(s,e)) \cdot \mu^\bullet(\delta^\bullet(f(s), g(s,e))) \\
&\qquad\qquad \text{Using that } g \text{ is an injection} \\
&\leq \sum_{e^\bullet \in E^\bullet} p^\bullet(f(s), e^\bullet) \cdot \mu^\bullet(\delta^\bullet(f(s), e^\bullet))) \\
&\qquad\qquad \text{Using the definition of } \mu^\bullet \\
&= \mu^\bullet(f(s))
\end{aligned}
$$

□

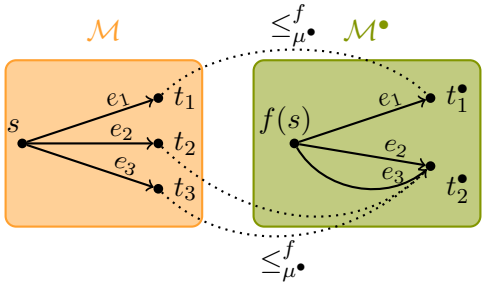
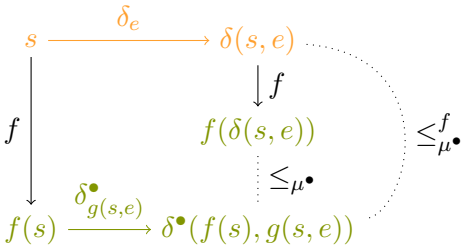



Figure 4.5: Structural conditions for guaranteed variance.

The hypotheses of Proposition 1 are easier to check than Equation 4.2.4 because they only rely on the structure of the model and the computation of $\mu^\bullet$. Condition 1 is completely structural. Condition 2 is more difficult, as shown in Figure 4.5. Actually for each state $s \in S$, each outgoing transition is mapped to a transition starting from $f(s)$ in the reduced model (in the figure, the mapping is done by the identity). The target states are in relation by the relation $\leq^f_{\mu^\bullet}$ define by $s \leq^f_{\mu^\bullet} t$ if $\mu^\bullet(f(s)) \leq \mu^\bullet(t)$. This condition can also be seen as follows:

$\forall s \in S, \forall e \in E :$




where $\delta_e$ stands for the function $s \mapsto \delta(s,e)$ and $s^\bullet \leq_{\mu^\bullet} t^\bullet$ if $\mu^\bullet(s^\bullet) \leq \mu^\bullet(t^\bullet)$.

For any state $s$, if $s$ is mapped into $\mathcal{C}^\bullet$ by $f$ and the transition labeled by $g(s,e)$ is taken (bottom left path of the diagram), it is more probable to reach $s^\bullet_+$ from the obtained state $\delta^\bullet\big(f(s), g(s,e)\big)$ than from the state $f\big(\delta(s,e)\big)$ which would be reached if the event $e$ is taken first, and its result is mapped to $\mathcal{C}^\bullet$ by $f$ (upper right path of the diagram). Thus the reduced model is going faster to $s^\bullet_+$ than the initial one is going to $s_+$.

Most of the time, Condition 2 is proven using a coupling (defined in Section 2.2.2). We therefore restate Proposition 1 in this context:

**Corollary 2**

Let $\mathcal{C}$ be an enriched BDTMC, $\mathcal{C}^\bullet$ be a reduction of $\mathcal{C}$ by $f$. Assume there exists a function $g : S \times E \to E^\bullet$ such that $\forall s \in S,\ e \mapsto g(s,e)$ is an injection which maps each transition of the model to a transition in the reduced model. $\leq_C$ is a relation over the states of $\mathcal{C}^\bullet$ such that if:
$\forall s \in S,\ \forall e \in E,$

1. $p(s,e) = p^\bullet(f(s), g(s,e))$;
2. $p(s,e) > 0 \;\Rightarrow\; f(\delta(s,e)) \leq_C \delta^\bullet(f(s), g(s,e))$;
3. $\leq_C$ is a coupling relation with target state $s_+$.

Then $\mathcal{C}^\bullet$ is a reduction of $\mathcal{C}$ by $f$ with guaranteed variance.

**Running example.** *For the tandem example, the function $g$ is chosen to be equal to the identity. Thus Condition 1 of Proposition 2 holds. For Condition 2, the coupling $S^\otimes = \{((n_1,n_2),(n_1',n_2')) \mid n_1+n_2 \leq n_1'+n_2' \wedge n_1 \leq n_1'\}$ defined at page 29 is used. For any state $(n_1,n_2) \in C^\bullet$ such that $n_2 < R$, the function $f$ behaves like the identity and $\delta_e$ and $\delta_e^\bullet$ are identical. Thus the diagram commutes and the property holds.*

*We now focus on the a state $s = (n_1,n_2)$ such that $n_2 \geq R$:*

- *For event $\lambda$: $\delta_\lambda$ and $\delta_\lambda^\bullet$ behave the same, in the sense that they both increase the first component $n_1$ by one, thus $f(\delta(s,\lambda)) = \delta^\bullet(f(s),\lambda)$ and the property holds.*
- *For event $\rho_1$: $\delta_{\rho_1}$ and $\delta_{\rho_1}^\bullet$ perform self-loops and thus the property holds.*
- *For event $\rho_2$: The case where $n_2 = R$ is similar to the previous cases. In the case where $n_2 > R$, the behaviors of $\delta_{\rho_2}$ and $\delta_{\rho_2}^\bullet$ are different:*
  - ▷ $f(\delta_{\rho_2}(n_1,n_2)) = (n_1+n_2-R-1, R)$
  - ▷ $\delta_{\rho_2}^\bullet(f(n_1,n_2)) = (n_1+n_2-R, R-1)$.

  *As $\big((n_1+n_2-R-1,R),(n_1+n_2-R,R-1)\big) \in S^\otimes$ the property holds.*

*Using Proposition 1, the reduction proposed for the tandem is thus a reduction with guaranteed variance reduction.*

We now weaken Condition 1 in order to enlarge the field of application of the method.

**Proposition 3**

Let $\mathcal{C}$ be an enriched BDTMC, $\mathcal{C}^\bullet$ be a reduction of $\mathcal{C}$ by $f$. Assume there exists a function $g : S \times E \to E^\bullet$ such that $\forall s,\ e \mapsto g(s,e)$ is an injection which maps each transition of the model to a transition in the reduced model, such that:

1. $\forall s \in S,\ \forall e \in E,$

$$p^\bullet\big(f(s), g(s,e)\big) > p(s,e) \Rightarrow \mu^\bullet\big(f(s)\big) \leq \mu^\bullet\Big(\delta^\bullet\big(f(s), g(s,e)\big)\Big) \quad \text{(1.a)}$$

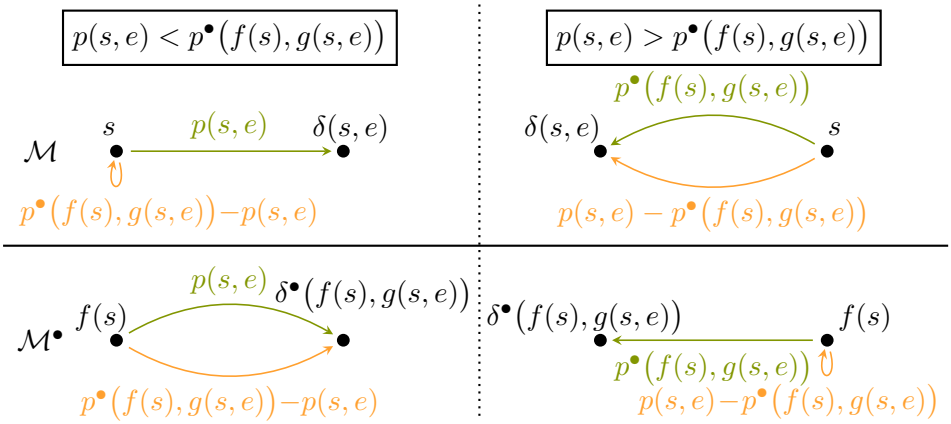
$$p^\bullet\big(f(s), g(s,e)\big) < p(s,e) \Rightarrow \mu^\bullet\big(f(s)\big) \geq \mu^\bullet\Big(\delta^\bullet\big(f(s), g(s,e)\big)\Big) \quad \text{(1.b)}$$

2. $\forall s \in S,\ \forall e \in E$ s.t. $p(s,e) > 0,$
$\mu^\bullet\Big(f\big(\delta(s,e)\big)\Big) \leq \mu^\bullet\Big(\delta^\bullet\big(f(s), g(s,e)\big)\Big)$

then $\mathcal{C}^\bullet$ is a reduction of $\mathcal{C}$ by $f$ with guaranteed variance.

*Proof:*

Let us give the idea underlying this proof: when a transition has different probabilities in the reduced and the initial models, it can be split into two parts by introducing loops. The first part of the transition is the same in both models and corresponds to the smallest probability. The second part is the probability difference between the original and reduced models. However, in the model with the smallest original transition probability, it is introduced as a self-loop, while in the other one, its target state is left unchanged.




Formally, we define a new enriched DTMC $\mathcal{D} = (S, E \uplus \tilde{E}, \tilde{\delta}, \tilde{p})$ where $\tilde{E}$ is a copy of $E$. Let $h(s) = \sum_{e \in E} \max\Big(p(s,e), p^\bullet\big(f(s), g(s,e)\big)\Big)$. For all $s \in S$:

$$\begin{aligned}
&\forall e \in E, \quad \tilde{\delta}(s,e) = \delta(s,e) \text{ and } \tilde{p}(s,e) = \tfrac{1}{h(s)} \min\Big(p(s,e), p^\bullet\big(f(s), g(s,e)\big)\Big),\\
&\forall e \in \tilde{E}, \quad \text{if } p^\bullet\big(f(s), g(f(s),e)\big) > p(s,e) \text{ then } \tilde{\delta}(s,e) = s,\\
&\qquad\qquad \text{if } p^\bullet\big(f(s), g(f(s),e)\big) \leq p(s,e) \text{ then } \tilde{\delta}(s,e) = \delta(s,e),\\
&\qquad\qquad \tilde{p}(s,e) = \tfrac{1}{h(s)} \left|p(s,e) - p^\bullet\big(f(s), g(s,e)\big)\right|.
\end{aligned}$$

The chain $\mathcal{D}$ is indeed a DTMC. Similarly, we define $\mathcal{D}^\bullet = (S^\bullet, E \uplus \tilde{E}, \tilde{\delta^\bullet}, \tilde{p^\bullet})$

with for all $s \in S$:

$$\begin{aligned}
&\forall e \in E,\ \tilde{\delta^\bullet}\big(f(s), g(s,e)\big) = \delta^\bullet\big(f(s), g(s,e)\big) \\
&\qquad \text{and } \tilde{p^\bullet}\big(f(s), g(s,e)\big) = \tfrac{1}{h(s)} \min\Big(p(s,e), p^\bullet\big(f(s), g(s,e)\big)\Big), \\
&\forall e \in \tilde{E},\ \text{if } p^\bullet\big(f(s), g(f(s),e)\big) > p(s,e) \text{ then } \tilde{\delta^\bullet}\big(f(s), g(s,e)\big) = \delta^\bullet\big(f(s), g(s,e)\big), \\
&\qquad \text{if } p^\bullet\big(f(s), g(f(s),e)\big) \leq p(s,e) \text{ then } \tilde{\delta^\bullet}\big(f(s), g(s,e)\big) = f(s), \\
&\qquad \tilde{p^\bullet}\big(f(s), g(s,e)\big) = \tfrac{1}{h(s)} \left| p(s,e) - p^\bullet\big(f(s), g(s,e)\big) \right|.
\end{aligned}$$

The above figure illustrates how transitions are split to create transitions with equal probabilities. An edge from left to right (respectively right to left) sends a state on another state with higher(respectively lower) probability to reach $s_+$.

As $\mu$ (respectively $\mu^\bullet$) expresses unconstrained time reachability probabilities, its value is identical on $\mathcal{C}$ and $\mathcal{D}$ (respectively on $\mathcal{C}^\bullet$ and $\mathcal{D}^\bullet$). The two models differ only by loops and transitions that can be split or not.

We can now prove that $\mathcal{D}^\bullet$ is a reduction with guaranteed variance of $\mathcal{D}$ by checking that it fulfills the hypotheses of Proposition 3. By construction the first condition holds. For all $s \in S$ and $e \in E \uplus \tilde{E}$ there are two cases:

1. If $e \in E$ as $\tilde{\delta}(s,e) = \delta(s,e)$, the condition holds by hypothesis.
2. If $e \in \tilde{E}$, then there are three cases:
   (a) $\tilde{p}(s,e) = 0$ : the condition trivially holds.
   (b) $\tilde{p}(s,e) = \frac{1}{h(s)}\big(p^\bullet(f(s), g(s,e)) - p(s,e)\big)$ : this case is depicted on the left part of the figure, and we have $\tilde{\delta}(s,e) = s$ and $\tilde{\delta^\bullet}(f(s), g(s,e)) = \delta^\bullet(f(s), g(s,e))$. Using hypothesis (1.a), we have:

$$\begin{aligned}
\mu^\bullet(f(\tilde{\delta}(s,e))) &= \mu^\bullet(f(s)) \\
&\leq \mu^\bullet(\delta^\bullet(f(s), g(s,e))) \\
&= \mu^\bullet(\tilde{\delta^\bullet}(f(s), g(s,e))).
\end{aligned}$$

   (c) $\tilde{p}(s,e) = \frac{1}{h(s)}\big(p(s,e) - p^\bullet(f(s), g(s,e))\big)$ : this case is depicted on the right part of the figure, and we have $\tilde{\delta}(s,e) = \delta(s,e)$ and $\tilde{\delta^\bullet}(f(s), g(s,e)) = f(s)$. Using hypothesis (1.b) and (2), we have:

$$\begin{aligned}
\mu^\bullet(\tilde{\delta^\bullet}(f(s), g(s,e))) &= \mu^\bullet(f(s)) \\
&\geq \mu^\bullet(\delta^\bullet(f(s), g(s,e))) \\
&\geq \mu^\bullet(f(\delta(s,e))) \\
&= \mu^\bullet(f(\tilde{\delta}(s,e))).
\end{aligned}$$

□

**Remark 7** *In this proposition, the requirement that transitions in relation by g should have equal probabilities in the two models is replaced by hypotheses (1.a) and (1.b): their probability can be greater in the reduced model if the transition leads to a state with higher probability to reach the state $s_+$ and can be smaller if the transition leads to a state with lower probability to reach the state $s_+$. These hypotheses can be checked with the coupling relation required for hypothesis (2).*

## 4.3 Reachability Analysis for Markov Chain

In this section, an importance sampling for unbounded reachability problems using reduced models is defined. Given a stochastic model $\mathcal{M}$ taking as semantics a DTMC or a CTMC which can be infinite, we are interested in computing the probability to reach a state $s_+$ before reaching the state $s_-$, supposing that these two states are reached with probability one, so that the problem input is a BDTMC.

### 4.3.1 General Importance Sampling

In the following, efficient important samplings based on reduced models (Definition 22) are constructed. Due to the fact that $\mu \neq \mu^\bullet \circ f$, for $s \in S$ such that $\mu^\bullet(f(s)) > 0$ the sum $h(s) = \sum_{s' \in S} \frac{\mu^\bullet(f(s'))}{\mu^\bullet(f(s))} \mathbf{P}(s, s')$ of the outgoing probabilities is not necessarily equal to 1. When $h(s) > 1$, the only reasonable choice for the construction of the transition probability matrix $\mathbf{P}'$ is to build an intermediate transition matrix summing to $h(s)$ for state $s$ and then divide each probability by $h(s)$. When $h(s) < 1$, there are two possibilities. Either one normalizes by $h(s)$, or the remaining probability $1 - h(s)$ leads to the sink state $s_-$. These two importance samplings are described more precisely in the following definition.

**Definition 24 (General Importance Sampling with Normalization)**

Let $\mathcal{C}$ be a BDTMC and $\mathcal{C}^\bullet$ a reduction of $\mathcal{C}$ by $f$. For $s \in S$, let $h(s) = \sum_{s' \in S} \frac{\mu^\bullet(f(s'))}{\mu^\bullet(f(s))} \mathbf{P}(s, s')$. We define a transition probability matrix $\mathbf{P}' = (\mathbf{P}'(s, s'))_{(s,s') \in S \times S}$ by the following rules, for all $s' \in S$:

- if $\mu^\bullet(f(s)) > 0$ then $\mathbf{P}'(s, s') = \frac{\mu^\bullet(f(s'))}{h(s)\mu^\bullet(f(s))} \mathbf{P}(s, s')$
- if $\mu^\bullet(f(s)) = 0$ then $\mathbf{P}'(s, s') = \mathbf{P}(s, s')$.

**Remark 8** *This importance sampling makes the state $s_-$ unreachable: the probability to go to a state $s'$ is proportional to $\mu^\bullet(f(s'))$, the probability to reach a state where $\mu^\bullet(f(s')) = 0$ is also null. The definition of the reduced*

*model (Definition 22) implies that if* $\mu^\bullet(f(s')) = 0$ *then* $\mu(s') = 0$ *thus* $s_+$ *is no longer reachable from* $s'$*.*

**Remark 9** *The second case of this definition is required for the completeness of the definition but is irrelevant as soon as state* $s_+$ *is reachable from the initial state. States where* $\mu^\bullet(f(s)) = 0$ *are not reachable and cannot reach state the* $s_+$ *thus trajectories visiting such a state always end in* $s_-$*.*

**Definition 25 (General Importance Sampling with Sink State)**

Let $\mathcal{C}$ be a BDTMC and $\mathcal{C}^\bullet$ a reduction of $\mathcal{C}$ by $f$. For $s \in S$, let $h(s) = \sum_{s' \in S} \frac{\mu^\bullet(f(s'))}{\mu^\bullet(f(s))} \mathbf{P}(s, s')$. We define a transition probability matrix $\mathbf{P}' = (\mathbf{P}'(s, s'))_{(s,s') \in S \times S}$ by the following rules:

- if $\mu^\bullet(f(s)) > 0$ and $h(s) \leq 1$ then for all $s' \in S \setminus \{s_-\}$,

$$\mathbf{P}'(s, s') = \frac{\mu^\bullet(f(s'))}{\mu^\bullet(f(s))} \mathbf{P}(s, s') \text{ and } \mathbf{P}'(s, s_-) = 1 - h(s)$$

- if $\mu^\bullet(f(s)) > 0$ and $h(s) > 1$ then for all $s' \in S$,

$$\mathbf{P}'(s, s') = \frac{\mu^\bullet(f(s'))}{h(s)\mu^\bullet(f(s))} \mathbf{P}(s, s')$$

- if $\mu^\bullet(f(s)) = 0$ then for all $s' \in S$, $\mathbf{P}'(s, s') = \mathbf{P}(s, s')$

**Remark 10** *Again the last case is irrelevant and only required for completeness.*

By construction, in the two definitions $\mathbf{P}'$ is a transition probability matrix. It complies to Conditions (3.2). Thus $\mathbf{P}'$ defines an importance sampling.

We are now in position to describe the whole method for a model $\mathcal{M}$ and a reachability goal $s_+$. This is a three-step procedure:

1. Specify a model $\mathcal{M}^\bullet$ with associated DTMC $\mathcal{C}^\bullet$, and a function $f$ such that $\mathcal{C}^\bullet$ is a reduction of $\mathcal{C}$ by $f$.

2. Compute function $\mu^\bullet$ with a numerical model checker applied on the DTMC $\mathcal{C}^\bullet$.

3. Estimate $\mu(s_0)$ with a statistical model checker applied on $\mathcal{C}$ using one of the importance samplings of Definition 24 and 25.

Steps 2 and 3 are automatic and can be performed by tools. Step 1 requires to manually design the reduced model. Figure 4.1 shows a schematic

description of this method. The choice between the two importance sampling is discussed in Chapter 6.7.

With the importance sampling using normalization (Definition 24), the random variable $W_{s_0}$ (Definition 20) can take in general any positive real value.

In the next proposition we study the set of values taken by the distribution of the random variable $W_{s_0}$ induced by the importance sampling with sink state (Definition 25).

**Proposition 4**

Let $\mathcal{C}$ be a BDTMC and $\mathcal{C}^\bullet$ its reduction. The importance sampling of Definition 25 has the following property: for all $s$ such that $\mu(s) > 0$, $W_s$ is a random variable which takes its values in $\{0\} \cup [\mu^\bullet(f(s)), \infty[$.

*Proof:*
Let $\sigma = (s = u_0, u_1, \ldots, u_l = s_+)$ be a trajectory starting in $s$ ending in $s_+$. As the trajectory avoids $s_-$, its value is

$$\frac{\mu^\bullet\big(f(u_0)\big)\max(h(u_0),1)}{\mu^\bullet\big(f(u_1)\big)} \cdots \frac{\mu^\bullet\big(f(u_{l-1})\big)\max(h(u_{l-1}),1)}{\mu^\bullet\big(f(u_l)\big)}$$

$$= \mu^\bullet\big(f(s)\big)\left(\prod_{\substack{0\le i<l,\\ h(u_i)>1}} h(u_i)\right) \geq \mu^\bullet\big(f(s)\big)$$

□

**Corollary 5**

Let $\mathcal{C}$ be a BDTMC and $\mathcal{C}^\bullet$ its reduction.

$$\mathbb{P}(W_{s_0} > 0) \leq \frac{\mu(s_0)}{\mu^\bullet(f(s_0))}$$

*Proof:*
The probability $\mathbb{P}(W_s > 0)$ is the probability for a trajectory to ends up in state $s_+$ starting from $s_0$, under this importance sampling. The expected value of $W_{s_0}$ can be written: $\mathbb{E}(W_{s_0}) = \mathbb{P}(W_{s_0} > 0) \cdot \mathbb{E}(W_{s_0}|W_{s_0} > 0)$. As $\mathbb{E}(W_{s_0}) = \mathbb{E}(V_{s_0}) = \mu(s_0)$ and $\mathbb{E}(W_{s_0}|W_{s_0} > 0) \geq \mu^\bullet(f(s_0))$.

□

Even if this proposition gives some clue on the values taken by $W_{s_0}$, the shape of its distribution is unknown and its variance may be infinite. Therefore, the shape of this distribution will be experimentally studied in Chapter 6.7.

### 4.3.2 Importance Sampling with Guaranteed Variance

In Section 4.3.1, a general importance sampling was defined. However, in this case, the shape of the distribution of the random variable $W_{s_0}$ was unknown and its variance possibly infinite.

In this section, the reduced models with guaranteed variance defined in Section 4.2 are used to construct important sampling for which the shape of the distribution has bounded support and the reduction of the variance of $W_{s_0}$ compare to $V_0$ is guaranteed.

Actually, in the general case, the normalization with $h(s) > 1$ increases the likelihood and may induce a large variance of $W_{s_0}$. For a reduction model with guaranteed variance, for any $s \in S,\ h(s) \leq 1$, therefore the definition of an important sampling with normalization remains unchanged (cf Definition 24) and Definition 25 of importance sampling with sink state can be rewritten as:

**Definition 26 (Importance Sampling with Guaranteed Variance)**

Let $\mathcal{C}$ be a BDTMC and $\mathcal{C}^\bullet$ be a reduction of $\mathcal{C}$ by $f$ with guaranteed variance. Let $\mathbf{P}'$ be a transition matrix on $S$ defined by:
Let $s$ be a state of $S$,

- if $\mu^\bullet\big(f(s)\big) = 0$ then for all $s' \in S$, $\mathbf{P}'(s, s') = \mathbf{P}(s, s')$
- if $\mu^\bullet\big(f(s)\big) > 0$ then for all $s' \in S \setminus \{s_-\}$,

$$\mathbf{P}'(s, s') = \frac{\mu^\bullet\big(f(s')\big)}{\mu^\bullet\big(f(s)\big)}\mathbf{P}(s, s') \quad \text{and} \quad \mathbf{P}'(s, s_-) = 1 - \sum_{s' \in S} \mathbf{P}'(s, s').$$

The reduction of the variance for such importance samplings is proven in the following proposition using the conditions defined earlier for reduced models with guaranteed variance, first for the normalization case, and then for the sink state case.

**Proposition 6**

Let $\mathcal{C}$ be a BDTMC and $\mathcal{C}^\bullet$ be a reduction with guaranteed variance by $f$. The importance sampling based on the matrix $\mathbf{P}'$ of Definition 24 has the following properties:

- For all $s \in S$ such that $\mu(s) > 0$,
  $W_s$ is a random variable which has value in $[0, \mu^\bullet\big(f(s)\big)]$.
- $\mu(s) \leq \mu^\bullet\big(f(s)\big)$ and $\mathbb{V}(W_s) \leq \mu(s)\mu^\bullet\big(f(s)\big) - \mu^2(s)$.

- A confidence interval for this importance sampling can be computed using Chernoff-Hoeffding bounds.

*Proof:*
Let $\sigma = (s = u_0, \dots, u_l = s_+)$ be a trajectory starting in $s$ ending in $s_+$.
Its value is:

$$\frac{\mu^\bullet\big(f(u_0)\big)h(u_0)}{\mu^\bullet\big(f(u_1)\big)} \cdots \frac{\mu^\bullet\big(f(u_{l-1})\big)h(u_{l-1})}{\mu^\bullet\big(f(u_l)\big)} \leq \frac{\mu^\bullet\big(f(u_0)\big)}{\mu^\bullet\big(f(u_l)\big)} = \mu^\bullet\big(f(s)\big)$$

We know that $\mathbb{E}(W_s) = \mu(s)$, then $\mathbb{P}\Big(W_s = \mu^\bullet\big(f(s)\big)\Big) = \frac{\mu(s)}{\mu^\bullet\big(f(s)\big)}$.
This implies that $\mu(s) \leq \mu^\bullet\big(f(s)\big)$ and $\mathbb{V}(W_s) = \mu(s)\mu^\bullet\big(f(s)\big) - \mu^2(s)$.
The variance is finite and as $W_{s_0}$ takes bounded values, Chernoff-Hoeffding bounds apply and we can compute confidence interval (see 2.3.4).

□

Similarly for the importance sampling with sink state the following proposition holds.

**Proposition 7**

Let $\mathcal{C}$ be a BDTMC and $\mathcal{C}^\bullet$ be a reduction with guaranteed variance by $f$. The importance sampling based on the matrix $\mathbf{P}'$ of Definition 26 has the following properties:

- For all $s$ such that $\mu(s) > 0$,
  $W_s$ is a random variable which has value in $\{0, \mu^\bullet\big(f(s)\big)\}$.
- $\mu(s) \leq \mu^\bullet\big(f(s)\big)$ and $\mathbb{V}(W_s) = \mu(s)\mu^\bullet\big(f(s)\big) - \mu^2(s)$.
- A confidence interval for this importance sampling can be computed using classical method on Bernouilli distribution.

*Proof:*
Let $\sigma = (s = u_0, \dots, u_l = s_+)$ be a trajectory starting in $s$ ending in $s_+$.
As the trajectory avoids $s_-$, its value is:

$$\frac{\mu^\bullet\big(f(u_0)\big)}{\mu^\bullet\big(f(u_1)\big)} \cdots \frac{\mu^\bullet\big(f(u_{l-1})\big)}{\mu^\bullet\big(f(u_l)\big)} = \frac{\mu^\bullet\big(f(u_0)\big)}{\mu^\bullet\big(f(u_l)\big)} = \mu^\bullet\big(f(s)\big)$$

We know that $\mathbb{E}(W_s) = \mu(s)$, then $\mathbb{P}\Big(W_s = \mu^\bullet\big(f(s)\big)\Big) = \frac{\mu(s)}{\mu^\bullet\big(f(s)\big)}$.
This implies that $\mu(s) \leq \mu^\bullet\big(f(s)\big)$ and $\mathbb{V}(W_s) = \mu(s)\mu^\bullet\big(f(s)\big) - \mu^2(s)$.
The variance is finite and as $W_{s_0}$ is a Bernouilli law, it takes only two values and it is possible to compute a confidence interval (see 2.3.5).

□

This proposition shows that Inequation (4.2) is a sufficient condition for obtaining an importance sampling with guaranteed variance reduction.

**Remark 11** *Since $\mu(s_0) \ll 1$, $\mathbb{V}(V_{s_0}) \approx \mu(s_0)$. If $\mu(s_0) \ll \mu^\bullet(f(s_0))$, we obtain $\mathbb{V}(W_{s_0}) \approx \mu(s_0)\mu^\bullet(f(s_0))$, so the variance is reduced by a factor $\mu^\bullet(f(s_0))$. In the case where $\mu(s_0)$ and $\mu^\bullet(f(s_0))$ have same order of magnitude, the reduction of variance is even bigger.*

**Running example.** *We have already seen that the tandem example is a reduction with guaranteed variance. The importance sampling produced by the reduced model behaves as follows: for all states $s = (n_1, n_2)$ such that $n_2 < R$ we can compute*

$$\begin{aligned}
&\textstyle\sum_{s' \in S} \frac{\mu^\bullet(f(s'))}{\mu^\bullet(f(s))} \boldsymbol{P}(s, s') \\
&= \textstyle\sum_{e \in E} \frac{\mu^\bullet(f(\delta(s,e)))}{\mu^\bullet(f(s))} p(s, e) \\
&\quad \textit{Since } n_2 \leq R,\ f(\delta(s, e)) = \delta^\bullet(f(s), e)) \\
&= \frac{\sum_{e \in E} \mu^\bullet(\delta^\bullet(f(s),e)) p^\bullet(f(s),e)}{\mu^\bullet(f(s))} \\
&= 1
\end{aligned}$$

*This shows that when the reduced model behaves like the original one (i.e. $f(\delta(s, e)) = \delta^\bullet(f(s), e))$) the probability to take the transition to $s_-$ is equal to 0.*

We are now in position to describe the whole method for guaranteed variance importance sampling for a model $\mathcal{M}$ and an unbounded reachability problem. It is a four-step procedure:

1. Specify a model $\mathcal{M}^\bullet$ with associated DTMC $\mathcal{C}^\bullet$ and the functions $f$ and $g$.
2. Prove using a coupling on $\mathcal{M}^\bullet$ that Proposition 3 holds.
3. Compute function $\mu^\bullet$ with a numerical model checker applied on $\mathcal{C}^\bullet$.
4. Estimate $\mu(s_0)$ with a statistical model checker applied on $\mathcal{C}$ using importance samplings of Definition 24 or Definition 26.

As in the general case, the last two steps are done by tools. The first step is the specification of $\mathcal{M}^\bullet$ which requires to manually study $\mathcal{M}$. The second step is in also done by hand (see the proofs of correctness in the examples).

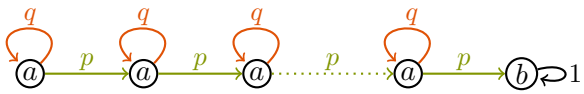



Figure 4.6: Rarity triggered by time bound

## 4.4 Time-Bounded Reachability for DTMC

### 4.4.1 Challenge

In this section the previous method is generalized, to the setting of bounded reachability probabilities in discrete time Markov chains.

The main motivation for developing a dedicated method to analyze bounded reachability probabilities is that an event can be frequent or even recurrent in a system, even if the time of its first occurrence is very large. Figure 4.6 shows an example of a DTMC where the probability to reach a state labeled by $b$ without time constraint is 1. However, if the transition probability $p$ is sufficiently small and if a time bound is set for the maximal number of steps, this problem might become a rare event one and an importance-sampling method is required. The methods presented in Section 4.3 where designed to estimate time-unbounded probabilities, even if the simulator is adapted to reject trajectories exceeding the time bound, the method is not efficient. In fact, in the reduced models built by these methods, the probability to reach the goal state should be greater than in the initial one. However, in this case, if the time bound is not taken into account this probability is already equal to 1 for any state which makes the importance sampling built by the method useless.

In the unbounded setting, the computation of a reachability property probability over a continuous time Markov chain is done by looking at the embedded DTMC. The formalism is the same for the two settings. In the time-bounded reachability setting, a distinction has to be made between the discrete and continuous cases: the two formalisms are different and will be handled separately. This section only focuses on DTMC.

### 4.4.2 Adapting Reachability Analysis

Given a model $\mathcal{M}$ taking as semantics a DTMC, $\mathcal{C}$ we are interested in computing the probability to reach a state $s_+$ in $u$ steps starting from the initial state $s_0$.

In order to adapt the method from the time-unbounded reachability case, the cross product between the Markov chain $\mathcal{C}$ and a counter is build. The counter counts the remaining number of transitions in the Markov chain before reaching the horizon. Trajectories that do not reach the state $s_+$ after $u$ steps end up in state $s_-$.

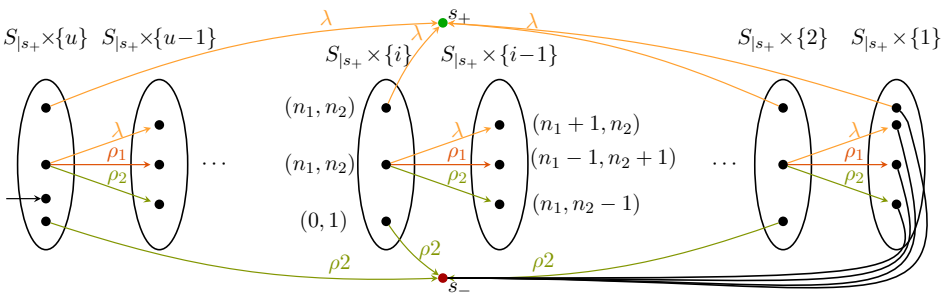



Figure 4.7: "Unfolding $\mathcal{C}_u$" for the example of the tandem queues, here $S_{|s_+}$ stands for $S\backslash\{s_+\}$

**Definition 27 (Time bounded Reachability Problem for DTMC)**

Given a DTMC $\mathcal{C}$, a reachability goal $s_+$ and a time bound $u$, we define the Markov chain $\mathcal{C}_u = \big(S_u, (s_0, u), \mathbf{P}_u\big)$ by:

- $S_u = \big(S\backslash\{s_+\} \times [1, u]\big) \cup \{s_-, s_+\}$
- $\mathbf{P}_u(s_-, s_-) = \mathbf{P}_u(s_+, s_+) = 1$ ($s_-, s_+$ are *absorbing* states)
- $\forall s, s' \in S\backslash\{s_+\},\ \forall 1 < v \leq u$
$$\begin{aligned} \mathbf{P}_u\big((s, v), (s', v-1)\big) &= \mathbf{P}(s, s') \\ \mathbf{P}_u\big((s, v), s_+\big) &= \mathbf{P}(s, s_+) \end{aligned}$$
- $\forall s \in S\backslash\{s_+\},\quad \mathbf{P}_u((s, 1), s_+) = \mathbf{P}(s, s_+)$
$\mathbf{P}_u((s, 1), s_-) = 1 - \mathbf{P}_u((s, 1), s_+)$
- Other transition probabilities are null.

**Remark 12** *Note that by construction $\mathcal{C}_u$ is a BDTMC: observe that the probability to reach $s_+$ or $s_-$ from any state is equal to 1. Moreover by construction, $\mu(s, v) = \mu_v(s)$ where $\mu(s, v)$ is the probability to reach $s_+$ in $\mathcal{C}_u$ and $\mu_v(s)$ is the probability to reach $s_+$ in $\mathcal{C}$ in $v$ steps (see Definition 9).*

**Running example.** *Figure 4.4.2 describes the Markov chain $\mathcal{C}_u$ associated with the example. The state space of $\mathcal{C}$ is replicated $u$ times. States where $n_1 + n_2 \geq N$ (respectively $n_1 + n_2 = 0$) are merged across all the copies into $s_+$ (respectively $s_-$). The initial state is $((1, 0), u)$. On the last copy, only transition $\lambda$ in the states where $n_1 + n_2 = N - 1$ lead to $s_+$, all others lead to state $s_-$.*

From a theoretical point of view, the importance sampling method developed in the previous section can now be used, on a Markov chain $\mathcal{C}_u$, but

as the size of $\mathcal{C}_u$ is $u$ times bigger than the one of $\mathcal{C}$, the relation between $\mathcal{C}$ and $\mathcal{C}_u$ must be used as much as possible, for the sake of scalability.

As the design of a reduced model relies on the dynamics of the system, we will define it on $\mathcal{C}$. We define a reduced model $\mathcal{C}^\bullet$ of $\mathcal{C}$ by a reduction function $f$. We then define as reduced model for $\mathcal{C}_u$, the BDTMC $\mathcal{C}_u^\bullet$ by applying Definition 27 to $\mathcal{C}^\bullet$. For all $0 \leq v \leq u$, the function $f_u$ is an extension of $f$ defined by $f_u(s,v) = (f(s), v)$. Moreover Remark 12 applies to distributions $(\mu_v^\bullet)_{v=0}^u$ where $\forall s^\bullet \in S^\bullet,\ \mu_v^\bullet(s^\bullet) = \mu^\bullet(s^\bullet, v)$, with $\mu^\bullet(s^\bullet, v)$ the probability to reach $s_+^\bullet$ in $\mathcal{C}_u^\bullet$ and $\mu_v^\bullet(s^\bullet)$ is the probability of reaching $s_+^\bullet$ in $\mathcal{C}^\bullet$ in $v$ steps.

In this context Definition 25 can be reformulated as:

**Definition 28 (Time Bounded Importance Sampling for DTMC)**

Let $\mathcal{C}$ be a DTMC, $\mathcal{C}^\bullet$ be a reduction of $\mathcal{C}$ by $f$ and $u$ a positive integer. Then $\mathbf{P}'_u$ is the transition matrix on $S_u$ the state space of $C_u$. Let $s$ be a state of $S\backslash\{s_+\}$ and $0 < v \leq u$,
let $h_v(s) = \sum_{s' \in S} \frac{\mu_{v-1}^\bullet(f(s'))}{\mu_v^\bullet(f(s))} \mathbf{P}(s,s')$. $\mathbf{P}'_u$ is defined by:

1. if $\mu_v^\bullet(f(s)) = 0$ then for all $s' \in S_u,\ \mathbf{P}'_u((s,v),s') = \mathbf{P}(s,s')$
2. if $\mu_v^\bullet(f(s)) > 0$ and $h_v(s) \leq 1$ then
   - If $v > 1$ then $\forall s' \in S\backslash\{s_+\}$,
     $\mathbf{P}'_u((s,v),(s',v-1)) = \frac{\mu_{v-1}^\bullet(f(s'))}{\mu_v^\bullet(f(s))} \mathbf{P}(s,s')$
   - $\mathbf{P}'_u((s,v),s_+) = \frac{\mathbf{P}(s,s_+)}{\mu_v^\bullet(f(s))}$
   - $\mathbf{P}'_u((s,v),s_-) = 1 - h_v(s)$.
3. if $\mu_v^\bullet(f(s)) > 0$ and $h_v(s) > 1$ then
   - If $v > 1$ then $\forall s' \in S\backslash\{s_+\}$,
     $\mathbf{P}'_u((s,v),(s',v-1)) = \frac{\mu_{v-1}^\bullet(f(s'))}{h_v(s)\mu_v^\bullet(f(s))} \mathbf{P}(s,s')$
   - $\mathbf{P}'_u((s,v),s_+) = \frac{\mathbf{P}(s,s_+)}{h_v(s)\mu_v^\bullet(f(s))}$
   - $\mathbf{P}'_u((s,v),s_-) = 0$.

**Remark 13** *This importance sampling uses $s_-$ as a sink state. The importance sampling with normalization can be defined similarly to Definition 24*

**Remark 14** *Similarly to the unbounded reachability case, suppose $h_v(s) \leq 1$ holds for all $s \in S$ and $v \in \{1,\ldots,u\}$, which means that Case 3 of Definition 28 never occurs, $\mathcal{C}_u^\bullet$ is a reduction with guaranteed variance of $\mathcal{C}_u$ by $f$ and an importance sampling with guaranteed variance is obtained.*

As the reduction of $\mathcal{C}_u$ to $\mathcal{C}_u^\bullet$ is obtained as an unfolding of a reduction of $\mathcal{C}$ to $\mathcal{C}^\bullet$, the guaranteed variance reduction property (Definition 23) can be rewritten for time-bounded properties as

**Definition 29 (Guaranteed Variance for Bounded Property)**

Let $\mathcal{C}$ be a DTMC and $\mathcal{C}^\bullet$ a reduction of $\mathcal{C}$ by $f$. $\mathcal{C}^\bullet$ is a *bounded reduction with guaranteed variance* if for all $v > 1$, for all $s \in S$ such that $\mu_v^\bullet(f(s)) > 0$ we have :

$$\sum_{s' \in S} \mathbf{P}(s, s') \cdot \mu_{v-1}^\bullet(f(s')) \leq \mu_v^\bullet(f(s)) \tag{4.4}$$

Similarly to the time-unbounded case (Proposition 1) structural requirements can be define to guarantee the reduction of variance. The first, more restrictice, case can be rewritten as:

**Proposition 8**

Let $\mathcal{C}$ be an enriched DTMC, $\mathcal{C}^\bullet$ be a reduction of $\mathcal{C}$ by $f$. Let $u$ be a positive integer. Assume there exists a function $g : S \times E \to E^\bullet$ such that $\forall s \in S,\ e \mapsto g(s, e)$ is an injection which maps each transition of the model to a transition in the reduced model, such that:
$\forall 1 < v \leq u,\ \forall s \in S,\ \forall e \in E$ :

1. $p(s, e) = p^\bullet(f(s), g(s, e))$
2. $p(s, e) > 0 \ \Rightarrow\ \mu_v^\bullet(f(\delta(s, e))) \leq \mu_v^\bullet(\delta^\bullet(f(s), g(s, e)))$

then $\mathcal{C}_u^\bullet$ is a reduction of $\mathcal{C}_u$ by $f$ with guaranteed variance.

The second, more general, case (Proposition 3) cannot be adapted to time-bounded setting. In fact, the proof requires to add self-loops to the Markov chain which modify its transient behaviors.

However, when a coupling on the states of $\mathcal{M}^\bullet$ is available, another proof of the guaranteed variance reduction can be established. This proof is based on the same hypotheses as the variance reduction in the unbounded case.

**Corollary 9**

Let $\mathcal{C}$ be an enriched DTMC, $\mathcal{C}^\bullet$ be a reduction of $\mathcal{C}$ by $f$. Let $u$ be an integer. Assume there exists a function $g : S \times E \to E^\bullet$ such that $\forall s \in S,\ e \mapsto g(s, e)$ is an injection which maps each transition of the model to a transition in the reduced model. $\leq_C$ is a relation over the states of $\mathcal{C}^\bullet$ such that:
$\forall s \in S,\ \forall e \in E$ :

1. $p(s, e) = p^\bullet(f(s), g(s, e))$
2. $p(s, e) > 0 \ \Rightarrow\ f(\delta(s, e)) \leq_C \delta^\bullet(f(s), g(s, e))$

3. $\leq_C$ is a coupling relation with target state $s_+$,

then $\mathcal{C}_u^\bullet$ is a reduction of $\mathcal{C}_u$ by $f$ with guaranteed variance.

*Proof:*
Using that $\leq_C$ is a coupling relation with target state $s_+$ implies that $\forall v > 1, \forall (s, s') \in S \times S,\ s \leq_C s' \Rightarrow \mu_v(s) \leq \mu_v(s')$, which is the hypothesis of Proposition 8.

□

**Remark 15** *This corollary has exactly the same hypotheses as Corollary 2. In this setting, we only need to build one reduced model with guaranteed reduction for both the time-unbounded and time-bounded reachability settings.*

### 4.4.3 Algorithmic Considerations

As $\mathcal{C}_u^\bullet$ is bigger than $\mathcal{C}^\bullet$, computing $\mu^\bullet$ over $\mathcal{C}_u^\bullet$ may be too expensive. More precisely, let $n$ be the size of $\mathcal{C}^\bullet$. The size of $\mathcal{C}_u^\bullet$ is $un$. Using Gaussian elimination to compute $\mu^\bullet$ requires $\Theta\big((un)^3\big)$ operations. Using iterative or more efficient matrix inversion methods leads to algorithms taking more than $O\big((un)^2\big)$ operations. Instead we compute $\mu_u^\bullet$ over $\mathcal{C}^\bullet$, which is the probability to reach $s_+^\bullet$ in at most $u$ steps. We use Equation 8 which requires $u$ matrix vector multiplications. The computation only requires $\Theta\big(un^2\big)$ operations. If the transition probability matrix of $\mathcal{C}^\bullet$ is sparse (which is usually the case) we can decrease furthermore this complexity. We denote by $d$ the maximum out-degree of the states of $\mathcal{C}^\bullet$. A simulation takes at most $u$ steps going through states $(s_u, u), \ldots, (s_1, 1), s_\pm$ where $s_u = s_0$ and $s_\pm \in \{s_+, s_-\}$. In state $(s_v, v)$, the computation of the distribution $P'_u((s_v, v), -)$ (cf. Definition 26) requires the computation of the values of $\mu_v^\bullet(f(s))$ and $\mu_{v-1}^\bullet(f(s'))$, for each possible target state $s'$ from $s_v$, where there are at most $d$ target states. Therefor, the vectors $\{\mu_v^\bullet\}_{0<v\leq u}$ may be computed iteratively one from the other with complexity $\Theta(ndu)$. Let us remark that in typical modelings, $d$ is very small compared to $n$. For example $d = 3$ in the running example. In most of the examples shown in this thesis, in which the state space size depend on a parameter, the number $d$ is independent of the parameter.

More precisely, we derive from $\mathbf{P}^\bullet$ the transition probability matrix of $\mathcal{C}$, the matrix $\mathbf{P}_0^\bullet$, a square (sub stochastic) matrix, indexed by $S\backslash\{s_-\}$ and defined by $\forall s, s' \in S\backslash\{s_-\}$:

- $\mathbf{P}_0^\bullet(s, s') = \mathbf{P}^\bullet(s, s')$,
- $\mathbf{P}_0^\bullet(s_+, s_+) = 1, \mathbf{P}_0^\bullet(s_+, s') = 0$.

Then $\mu_v^\bullet = \mathbf{P}_0^\bullet \cdot \mu_{v-1}^\bullet$ and $\mu_0^\bullet$ is null except $\mu_0^\bullet(s_+) = 1$. For large values of $u$, the space complexity to store the vectors $\mu_v^\bullet$ becomes intractable and the

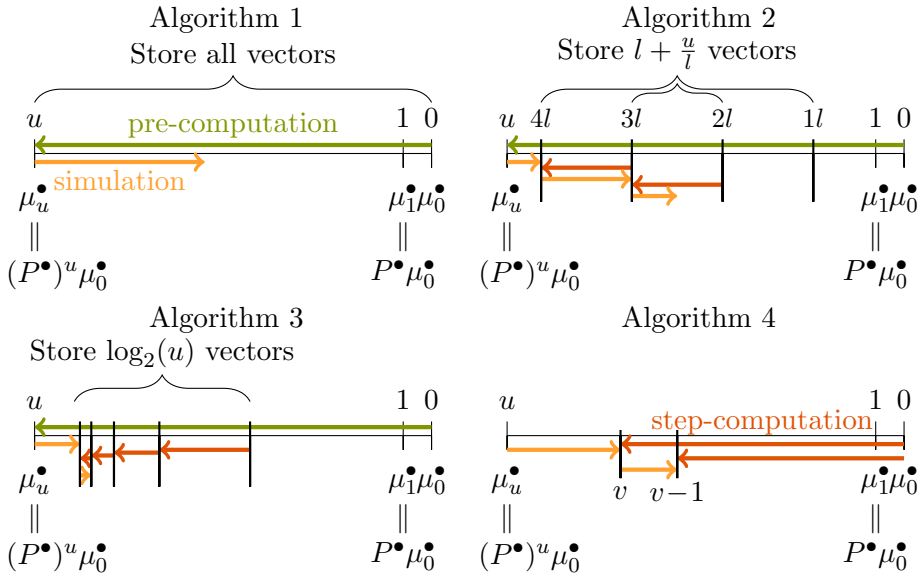



Figure 4.8: Different strategies for storing vectors

challenge is to obtain a suitable space-time trade-off. This problem of space complexity is due to the fact that the computation of vectors $\mu_v^\bullet$ is performed iteratively starting from the end ($v = 0$) whereas the simulation starts from the initial state ($v = u$) and decreases $v$ at each step.

We propose four methods to implement this space-time trade-off. The methods consist of a precomputation stage and a simulation stage. Their difference lies in the information stored during the first stage and the additional numerical computations during the second stage. In the precomputation, the three first methods compute iteratively the $u$ vectors $\mu_v^\bullet = (P_0^\bullet)^v(\mu_0^\bullet)$ for $v$ from 1 to $u$. The fourth one does not have a precomputation stage. An overview of the four algorithms is shown in Figure 4.8. The respective time and space complexities of the four algorithms are shown in Table 4.1.

### First Algorithm

The first algorithm correspond to the naive implementation. It consists in storing all the vectors $(\mu_v^\bullet)_{v=1}^u$ during the precomputation stage and then proceeding to the simulation without any additional numerical computations as shown in the top left corner of Figure 4.8. The precomputation stage is described in Algorithm 1 where list $L$ is the only memory requirement, and takes $\Theta(nu)$ memory space. The precomputation requires $u$ matrix vector multiplications taking $\Theta(ndu)$ elementary operations.

**Algorithm 1:**

`Precomputation`$(u, \mu_0^\bullet, P_0^\bullet)$
**Result**: $L$

`/* List` $L$ `fulfills` $L(i) = \mu_i^\bullet$ `*/`

1 $L(1) \leftarrow \mu_0^\bullet$
2 **for** $i = 2$ **to** $u$ **do**
3 $\quad L(i) \leftarrow P_0^\bullet L(i-1)$

### Second Algorithm

The second algorithm is described in the top right corner of Figure 4.8. Let $l(< u)$ be an integer. In the precomputation stage, it only stores the $\lfloor \frac{u}{l} \rfloor + 1$ vectors $\mu_v^\bullet$ with $v$ multiple of $l$ in list $L$ and $\mu^\bullet_{l\lfloor \frac{u}{l} \rfloor + 1}, \ldots, \mu_u^\bullet$ in list $K$ (see the precomputation stage of Algorithm 2).

During the simulation stage, in a state $(s, v)$, with $v = ml$, the vector $\mu_{v-1}^\bullet$ is present neither in $L$ nor in $K$. Therefore the algorithm uses vector $\mu^\bullet_{l(m-1)}$ stored in $L$ to compute iteratively all vectors $\mu^\bullet_{l(m-1)+i} = P^{\bullet i}(\mu^\bullet_{l(m-1)})$ for $i$ from 1 to $l-1$ and stores them in $K$ (see the step computation stage of Algorithm 2). Then it proceeds to $l$ consecutive steps of simulation without additional computations. The integer $l$ is chosen close to $\sqrt{u}$ in order to minimize the space complexity of the algorithm.

### Third Algorithm

The third algorithm is shown in the bottom left corner of Figure 4.8. Let $l = \lfloor \log_2(u) \rfloor + 1$. In the precomputation stage, it only stores $l + 1$ vectors in $L$. More precisely,

- initially using the binary decomposition of $u$ ($u = \sum_{i=0}^{l} a_{u,i} 2^i$), the list $L$ of $l + 1$ vectors consists of $\mathbf{w}_{i,v} = \mu^\bullet_{\sum_{j=i}^{l} a_{v,j} 2^j}$, for all $1 \leq i \leq l + 1$ (see the precomputation step of algorithm 3).

- During the simulation stage in a state $(s, v)$, with the binary decomposition of $v$ ($v = \sum_{i=0}^{l} a_{v,i} 2^i$), the list $L$ consists of $\mathbf{w}_{i,v} = \mu^\bullet_{\sum_{j=i}^{k} a_{v,j} 2^j}$, for all $1 \leq i \leq l + 1$. Observe that the first vector $\mathbf{w}_{1,v}$ is equal to $\mu_v^\bullet$. We obtain $\mu_{v-1}^\bullet$ by updating $L$ according to $v - 1$.

Let us describe the updating of the list performed by the stepcomputation of Algorithm 3. Let $i_0$ be the smallest index such that $a_{v,i_0} = 1$. Then, for $i > i_0$, $a_{v-1,i} = a_{v,i}$ and $a_{v-1,i_0} = 0$, and for $i < i_0$, $a_{v-1,i} = 1$. The new list $L$ is then obtained as follows.

- For $i > i_0$ $\mathbf{w}_{i,v-1} = \mathbf{w}_{i,v}$, $\mathbf{w}_{i_0,v-1} = \mathbf{w}_{i_0-1,v}$.

**Algorithm 2:**

`Precomputation`$(u, \mu_0^\bullet, P_0^\bullet)$
**Data**: $\mathbf{w}, l$
**Result**: $L, K$

`/* List` $L$ `fulfills` $L(i) = \mu_{i \cdot l}^\bullet$ `*/`

$l \leftarrow \lfloor \sqrt{u} \rfloor$; $\mathbf{w} \leftarrow \mu_0^\bullet$
**for** $i$ **from** *1* **to** $\lfloor \frac{u}{l} \rfloor l$ **do**
  $\mathbf{w} \leftarrow P_0^\bullet \mathbf{w}$
  **if** $i \mod l = 0$ **then**
    $L(\lfloor \frac{i}{l} \rfloor) \leftarrow \mathbf{w}$

`/* List` $K$ `contains` $\mu_{\lfloor \frac{u}{l} \rfloor l+1}^\bullet, \ldots, \mu_u^\bullet$ `*/`

**for** $i$ **from** $\lfloor \frac{u}{l} \rfloor l + 1$ **to** $u$ **do**
  $\mathbf{w} \leftarrow P_0^\bullet \mathbf{w}$
  $K(i \mod l) \leftarrow \mathbf{w}$

`Stepcomputation`$(v, l, P_0^\bullet, K, L)$
**Result**: $K$

`/* List` $K$ `contains` $\mu_{\lfloor \frac{v}{l} \rfloor l+1}^\bullet, \ldots, \mu_{\lceil \frac{v}{l} \rceil l}^\bullet$ `*/`

**if** $v \mod l = 0$ **then**
  $\mathbf{w} \leftarrow L(\lfloor \frac{v}{l} \rfloor - 1)$
  **for** $i$ **from** $(\lfloor \frac{v}{l} \rfloor - 1)l + 1$ **to** $v - 1$ **do**
    $\mathbf{w} \leftarrow P_0^\bullet \mathbf{w}$
    $K(i \mod l) \leftarrow \mathbf{w}$

- For $i_0 < i$, the vectors $\mathbf{w}_{i,v-1}$ are stored along $2^{i_0-1}-1$ iterated matrix-vector products starting from vector $\mathbf{w}_{i_0,v-1}$:

$$\mathbf{w}(j, v-1) = P_0^{\bullet 2^j}\mathbf{w}(j+1, v-1)$$

.

The computation associated with $v$ requires $1+2+\cdots+2^{i_0-1}$, i.e. $\Theta(nd2^{i_0})$ matrix-vector products. Observing that the bit $i$ is reset at most $u2^{-i}$ times, the complexity of the whole computation is

$$\sum_{i=1}^{k} 2^{k-i}\Theta(nd2^i) = \Theta(ndu\log(u)).$$

**Algorithm 3:**

`Precomputation`$(u, \mu_0^\bullet, P_0^\bullet)$
**Data**: $\mathbf{w}, l, i_0$
**Result**: L

`/* L fulfills` $L(i) = \mu^\bullet_{\sum_{j=i}^{l} a_{u,j}2^j}$ `*/`

```
1  l ← ⌊log2(u)⌋ + 1;  w ← μ0•;  L(l + 1) ← w
2  for i from l downto 0 do
3      if a_{u,i} = 1 then
4          for j from 1 to 2^i do
5              w ← P0• w
6      L(i) ← w
```

`Stepcomputation`$(v, P_0^\bullet, L)$

`/* L is updated accordingly to` $v-1$ `*/`

```
8  i0 ← min(i | a_{v,i} = 1);  w ← L(i0 + 1);  L(i0) ← w
9  for i from i0 − 1 downto 0 do
10     for j = 1 to 2^i do
11         w ← P0• w
12     L(i) ← w
```

### Fourth Algorithm

The last algorithm is also a naive approach. It does not store any vector and performs the whole computation at each step as shown in the bottom right corner of Figure 4.8. This algorithm is not effective as the time complexity is extremely high. Otherwise stated, it provides a lower bound on the memory requirement i.e. $2n$, the space needed for a matrix-vector multiplication

without taking into account the space required to store the matrix. The space memory of $2n$ is also the space needed by a numerical model checker computing $\mu_u(s_0)$.

**Algorithm 4:**

`Stepcomputation`$(v, \mu_0^\bullet, P_0^\bullet)$
**Result**: **w**, **w'**
`/* Vector w equals to` $\mu_v^\bullet$ `*/`
$\mathbf{w} \leftarrow \mu_0^\bullet$
**for** $i = 1$ **to** $u$ **do**
  $\mathbf{w'} \leftarrow P_0^\bullet \mathbf{w}$
  $\mathbf{w} \leftarrow \mathbf{w}$

The four algorithms are numbered according to their decreasing space complexity. The corresponding space-time trade-off is summarized in Table 4.1, where the space requirement only takes into accounts the manipulated vectors and is expressed as a number of reals. Other memory requirements are negligible as $n$ and $u$ grow.

**More on simulation.**

Simulating sequentially trajectories is really inefficient since the additional computations during the simulation stage are repeated during every simulation. Thus, for Algorithm 2, 3 and 4 we proceed with a bunch of trajectories simultaneously simulated step by step. This requires additional memory as each simulation needs space for the state of the system and the event queue. The impact of the batch size on the simulation time and space requirement is investigate in more detail in Chapter 6.7.

To summarize the overall method for evaluating time-bounded reachability in the discrete setting: considering a discrete model $\mathcal{M}$, having as semantics

Table 4.1: Compared complexities of the four algorithms

| Complexity | Algo. 1 | Algo. 2 | Algo. 3 | Algo. 4 |
|---|---|---|---|---|
| Space | $nu$ | $2n\sqrt{u}$ | $n \log u$ | $2n$ |
| Time for the precomputation | $\Theta(ndu)$ | $\Theta(ndu)$ | $\Theta(ndu)$ | 0 |
| Additional time for the simulation | 0 | $\Theta(ndu)$ | $\Theta(ndu \log(u))$ | $\Theta(ndu^2)$ |

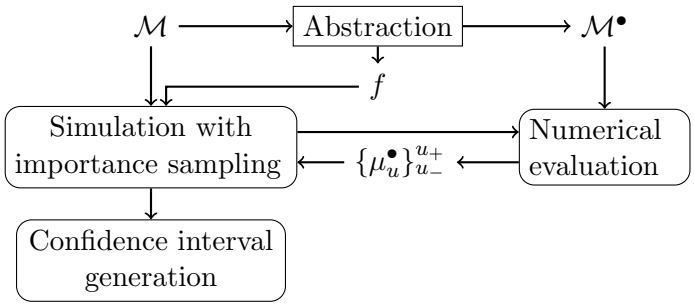



Figure 4.9: Schematic summary of the method for DTMC

a DTMC $\mathcal{C}$, the probability to reach a target state $s_+$ in $u$ steps can be estimated as follows:

1. Exhibit a suitable reduced model $\mathcal{M}^\bullet$ having as semantics a DTMC $\mathcal{C}^\bullet$, such that $\mathcal{C}^\bullet$ is a reduction of $\mathcal{C}$ by $f$.
2. (Optional) Prove that this reduction guarantees the variance using a coupling and Proposition 9.
3. Choose one of the algorithms implementing the time-memory trade-off. Alternatively, start with Algorithm 1 and if it fails due to memory requirement, use a less memory consuming algorithm.
4. Perform the pre-computation of the chosen algorithm.
5. Use the distributions $\mu_v^\bullet$ to perform importance sampling on a batch of simulations of model $\mathcal{M}$. At each step for each trajectory, take a transition, then update the vector $\mu_v^\bullet$ according to the chosen algorithm.

Steps 4 and 5 are automatic. Step 3 could also be automatized by trying each algorithm starting from the first one. As for the unbounded case, the first and second steps require human intervention. Figure 4.9 shows a scheme of this method.

## 4.5 Time-Bounded Reachability for CTMC

In this section, our method is extended to compute time-bounded reachability probabilities in CTMC. This is done by reducing the problem on the CTMC to a time-bounded reachability problem over its embedded DTMC (c.f. Definition 13).

The problem studied in this section takes as input a CTMC $\mathcal{C}$ and a positive real $\tau$ to compute the probability to reach a state $s_+$ in $\tau$ time units starting from state $s_0$.

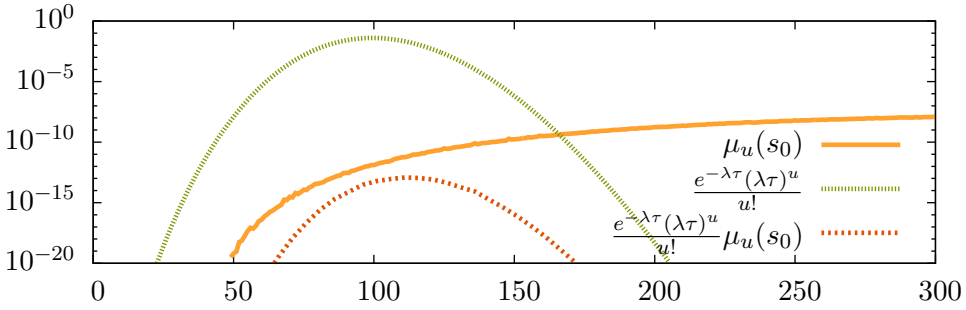



Figure 4.10: Repartition of Poisson and $\mu_u(s_0)$ Probabilities as a function of discrete time

### 4.5.1 Transient Analysis

Recall that for any CTMC with a bound on the outgoing rates of all states, a uniform CTMC (Definition page 33) can be built such that bounded reachability probabilities are equals in the two chains.

In a uniform CTMC with outgoing rate $\lambda$, the probability to reach state $s_+$ in $\tau$ time units starting from state $s$ is given by $\mu_\tau(s)$ (see Definition 14) which can be expressed as a weighted sum of the probabilities to reach state $s^+$ starting from $s$ in the embedded DTMC with the following formula:

$$\mu_\tau(s) = \sum_{u\in\mathbb{N}} \frac{e^{-\lambda\tau}(\lambda\tau)^u}{u!}\mu_u(s).$$

This sum is weighted by Poisson probabilities $\frac{e^{-\lambda\tau}(\lambda\tau)^u}{u!}$, which are plotted on Figure 4.10 along with the probabilities $\mu_u(s_0)$ for the tandem example. One can see that $\frac{e^{-\lambda\tau}(\lambda\tau)^u}{u!}$ tends quickly to 0 when $u$ tends to 0 or to the infinity.

As this sum is infinite, bound are necessary, in [35], given some rate $\lambda\tau$ and accuracy parameters $\alpha, \beta$, a numerical method computes indexes $u^-$ and $u^+$ and coefficients $c_u$ such that for $u^- \leq u \leq u^+$:

$$c_u(1-\alpha-\beta) \leq \frac{e^{-\lambda\tau}(\lambda\tau)^u}{u!} \leq c_u,$$

$$\sum_{u<u^-} \frac{e^{-\lambda\tau}(\lambda\tau)^u}{u!} \leq \alpha,$$

$$\sum_{u>u^+} \frac{e^{-\lambda\tau}(\lambda\tau)^u}{u!} \leq \beta.$$

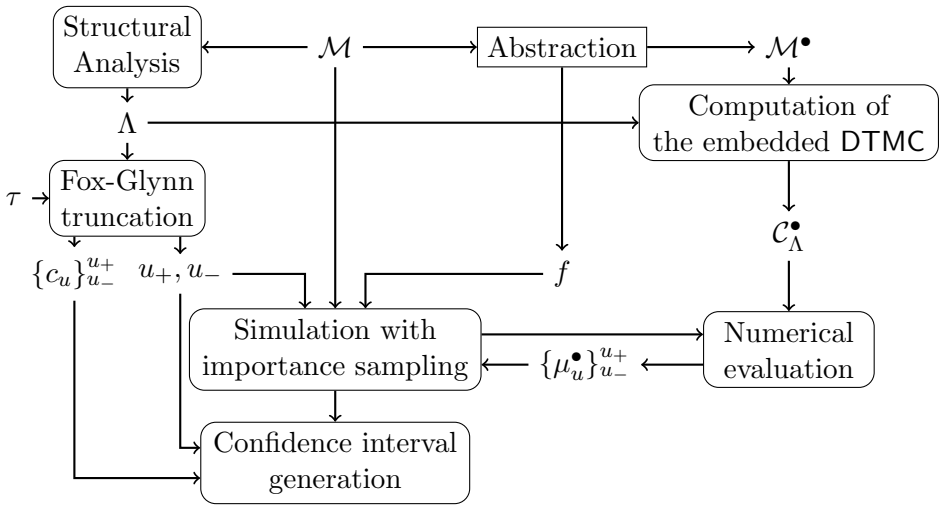



Figure 4.11: Scheme summarizing the method for bounded CTMC

We are now interested in estimating:

$$\sum_{u=u^-}^{u^+} \frac{e^{-\lambda\tau}(\lambda\tau)^u}{u!}\mu_u(s)$$

The method described in Section4.4 allows to compute $\mu_u(s)$ in a DTMC, for all $u$ between $u^-$ and $u^+$. Figure 4.11 summarizes how the different algorithms fit together to build the importance sampling.

Additional analysis is required to bound the errors made by the different parts of the computation. Each estimation of $\mu_u(s)$ is subject to a statistical error controlled by a confidence interval $I_u$ and a confidence level $\varepsilon_u$ such that $\mathbb{P}\big(\mu_u(s) \notin I_u\big) \leq \varepsilon_u$ holds. The error coming from the truncation of the sum is controlled, for the Poisson probabilities term, by $\alpha$ and $\beta$. It remains to bound $\mu_u(s)$ on $\mathbb{N}\backslash\{u^-, \ldots, u^+\}$. As $\mu_u(s)$ is increasing with respect to $u$, a simple bound is $\mu_{u^-}(s)$ for the lower part and $\mu_\infty(s) = \mu(s)$ for the upper part. These two probabilities can be estimated using previous methods which return confidence intervals whose upper bound are $\mathrm{up}_{u^-}$ and $\mathrm{up}_\infty$ with confidence level $\varepsilon'_{u^-}$ and $\varepsilon'_\infty$ such that $\mathbb{P}\Big(\sup_{k<u}\big(\mu_k(s)\big) > \mathrm{up}_u\Big) \leq \varepsilon'_u$. Finally the sum between 0 and $u^-$ is bounded by $\alpha\mathrm{up}_{u^-}$ and the sum after $u^+$ is bounded by $\alpha\mathrm{up}_\infty$

Figure 4.12 is a diagram summarizing the different bounds. The dotted lines are the boundings of $\mu_u(s)$ on $\mathbb{N}\backslash\{u^-, \ldots, u^+\}$ and the area of the filled part is bounded by $\alpha$ and $\beta$.

By combining all the parts together we obtain a statistical evaluation of

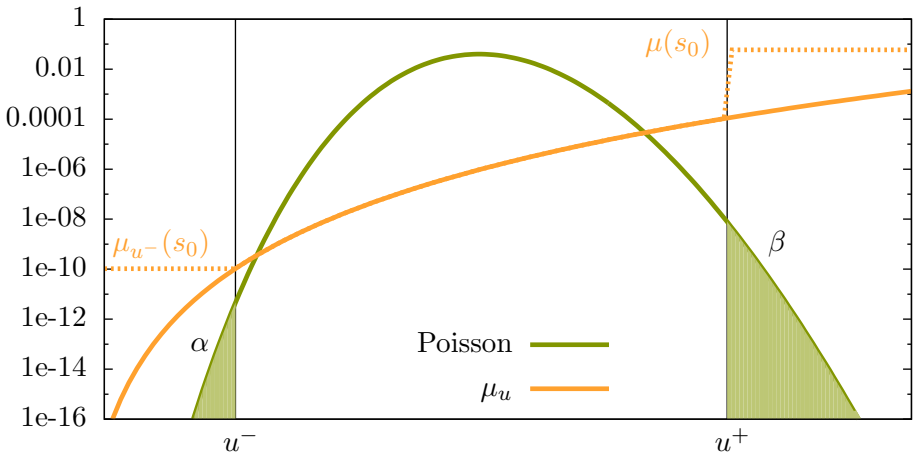



Figure 4.12: Bounding Poisson and $\mu_u(s_0)$ Probabilities

$\mu_\tau(s)$, as follows. The threshold probability is defined by:

$$\sum_{u=u^-}^{u^+} \varepsilon_u + \varepsilon'_{u^-} + \varepsilon'_\infty, \tag{4.5}$$

and the confidence interval $I$ is defined by:

$$I = \sum_{u=u^-}^{u^+} [c_u(1-\alpha-\beta), c_u] \cdot I_u + [0, \alpha \mathrm{up}_{u^-}] + [0, \beta \mathrm{up}_\infty] \tag{4.6}$$

When $\mu(s_0)$ is close to 1, the upper bound is not accurate enough. A workaround is to cut this sum into two parts:

$$\sum_{u>u^+} \frac{e^{-\lambda\tau}(\lambda\tau)^u}{u!}\mu_u(s_0) \leq \sum_{u>u^+}^{u^{++}} \frac{e^{-\lambda\tau}(\lambda\tau)^u}{u!}\mu_{u^{++}}(s_0) + \sum_{u>u^{++}} \frac{e^{-\lambda\tau}(\lambda\tau)^u}{u!}\mu(s_0)$$

Let $\gamma$ be a bound on the sum of Poisson probabilities from $u^{++}$ to infinity.

$$\sum_{u>u^+} \frac{e^{-\lambda\tau}(\lambda\tau)^u}{u!}\mu_u(s_0) \leq \beta\mu_{u^{++}}(s_0) + \gamma\mu(s_0)$$

Figure 4.13 summarizes the bounding of the upper part of the sum.

### 4.5.2 Guaranteed Variance for CTMC

To obtain guaranteed variance reduction in the setting of bounded reachability problem for a CTMC, we could require that the embedded DTMC of the

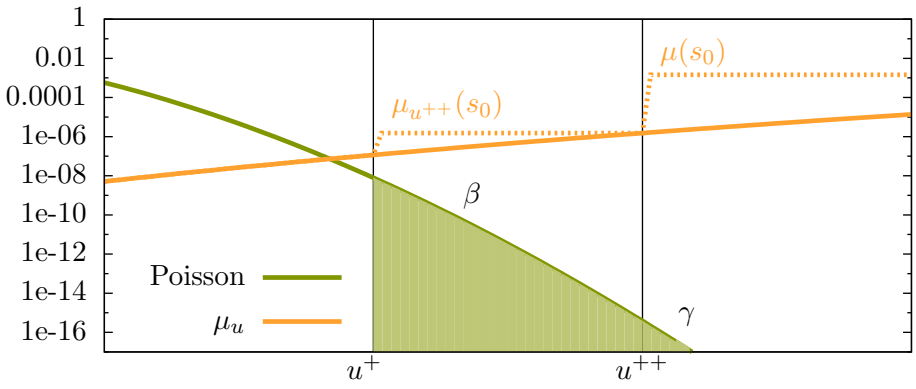



Figure 4.13: Bounding upper part of Poisson and $\mu_u(s_0)$ Probabilities

Markov chain satisfies Inequality (4.4), but it would be more convenient to work directly on the CTMC. The next proposition is a version of proposition 3 in the CTMC setting. Note that the hypotheses of this proposition are on the CTMC only although the conclusion is on the embedded DTMC.

**Proposition 10**

Let $\mathcal{C}$ be an enriched CTMC, $\mathcal{C}^\bullet$ be a reduction of $\mathcal{C}$ by $f$. Assume there exists a function $g : S \times E \to E^\bullet$ such that $\forall s \in S,\ e \mapsto g(s,e)$ is an injection which maps each transition of the model to a transition in the reduced model, such that:

1. $\forall s \in S,\ \forall e \in E,\ \forall u \in \mathbb{N}^*$

$$\lambda^\bullet\big(f(s), g(s,e)\big) > \lambda(s,e) \Rightarrow \mu_u^\bullet\big(f(s)\big) \leq \mu_u^\bullet\Big(\delta^\bullet\big(f(s), g(s,e)\big)\Big) \quad (1.a)$$

$$\lambda^\bullet\big(f(s), g(s,e)\big) < \lambda(s,e) \Rightarrow \mu_u^\bullet\big(f(s)\big) \geq \mu_u^\bullet\Big(\delta^\bullet\big(f(s), g(s,e)\big)\Big) \quad (1.b)$$

2. $\forall s \in S,\ \forall e \in E$ s.t.
$\lambda(s,e) > 0,\ \mu_u^\bullet\Big(f\big(\delta(s,e)\big)\Big) \leq \mu_u^\bullet\Big(\delta^\bullet\big(f(s), g(s,e)\big)\Big)$

then the embedded DTMC of $\mathcal{C}^\bullet$ is a bounded reduction of the embedded DTMC of $\mathcal{C}$ by $f$ with guaranteed variance for bounded property.

*Proof:*
We mimic the proof of Proposition 3. The same splitting of transitions can be performed to build two CTMC where all corresponding transitions have the same rate. As this transformation is done on a CTMC, the additional loops do not change the transient behaviors of the model. This was not the case for DTMC where the transient behaviors where affected by loops.

We define a new enriched CTMC $\mathcal{D} = (S, E \uplus \tilde{E}, \tilde{\delta}, \tilde{\lambda})$ where $\tilde{E}$ is a copy of $E$. For all $s \in S$:

$$\begin{aligned}
&\forall e \in E, \quad \tilde{\delta}(s,e) = \delta(s,e) \text{ and } \tilde{\lambda}(s,e) = \min(\lambda(s,e), \lambda^\bullet(f(s), g(s,e))) \\
&\forall e \in \tilde{E}, \quad \text{if } \lambda^\bullet(f(s), g(f(s),e)) > \lambda(s,e) \text{ then } \tilde{\delta}(s,e) = s \\
&\qquad\qquad \text{if } \lambda^\bullet(f(s), g(f(s),e)) \leq \lambda(s,e) \text{ then } \tilde{\delta}(s,e) = \delta(s,e) \\
&\qquad\qquad \tilde{\lambda}(s,e) = |\lambda(s,e) - \lambda^\bullet(f(s), g(s,e))|
\end{aligned}$$

Similarly, we define $\mathcal{D}^\bullet = (S^\bullet, E \uplus \tilde{E}, \tilde{\delta^\bullet}, \tilde{\lambda^\bullet})$ with for all $s \in S$:

$$\begin{aligned}
&\forall e \in E, \ \tilde{\delta^\bullet}(f(s), g(s,e)) = \delta^\bullet(f(s), g(s,e)) \\
&\qquad\qquad \text{and } \tilde{\lambda^\bullet}(f(s), g(s,e)) = \min(\lambda(s,e), \lambda^\bullet(f(s), g(s,e))) \\
&\forall e \in \tilde{E}, \ \text{if } \lambda^\bullet(f(s),g(f(s),e)) > \lambda(s,e) \text{ then } \tilde{\delta^\bullet}(f(s),g(s,e)) = \delta^\bullet(f(s),g(s,e)) \\
&\qquad\qquad \text{if } \lambda^\bullet(f(s), g(f(s),e)) \leq \lambda(s,e) \text{ then } \tilde{\delta^\bullet}(f(s), g(s,e)) = f(s) \\
&\qquad\qquad \tilde{\lambda^\bullet}(f(s), g(s,e)) = |\lambda(s,e) - \lambda^\bullet(f(s), g(s,e))|
\end{aligned}$$

We can now prove that the embedded DTMC of $\mathcal{D}^\bullet$ is a reduction with guaranteed variance of the embedded DTMC of $\mathcal{D}$ by checking that it fulfills hypotheses of Proposition 8. Let $\Lambda(s)$ be equal to $\sum_{e' \in E \uplus \tilde{E}} \tilde{\lambda}(s,e')$. By construction, the first condition of Proposition 8 holds. For all $u \in \mathbb{N}^*$, $s \in S$ and $e \in E \uplus \tilde{E}$ there is two cases:

1. If $e \in E$, as $\tilde{\delta}(s,e) = \delta(s,e)$, the condition holds by hypothesis.
2. If $e \in \tilde{E}$, there are three cases:
   (a) $\tilde{\lambda}(s,e) = 0$ and the condition trivially holds.
   (b) $\tilde{\lambda}(s,e) = \big(\lambda^\bullet(f(s), g(s,e)) - \lambda(s,e)\big)$ then $\tilde{\delta}(s,e) = s$ , $\tilde{\delta^\bullet}(f(s), g(s,e)) = \delta^\bullet(f(s), g(s,e))$. Using Hypothesis (1.a) we have:
   $$\begin{aligned}
   \mu_u^\bullet(f(\tilde{\delta}(s,e))) &= \mu_u^\bullet(f(s)) \\
   &\leq \mu_u^\bullet(\delta^\bullet(f(s), g(s,e))) \\
   &= \mu_u^\bullet(\tilde{\delta^\bullet}(f(s), g(s,e)))
   \end{aligned}$$
   (c) $\tilde{\lambda}(s,e) = \big(\lambda(s,e) - \lambda^\bullet(f(s), g(s,e))\big)$ then $\tilde{\delta}(s,e) = \delta(s,e)$, $\tilde{\delta^\bullet}(f(s), g(s,e)) = f(s)$ . Using Hypotheses (1.b) and (2) we have:
   $$\begin{aligned}
   \mu_u^\bullet(\tilde{\delta^\bullet}(f(s), g(s,e)) &= \mu_u^\bullet(f(s)) \\
   &\geq \mu_u^\bullet(\delta^\bullet(f(s), g(s,e))) \\
   &\geq \mu_u^\bullet(f(\delta(s,e))) \\
   &= \mu_u^\bullet(f(\tilde{\delta}(s,e)))
   \end{aligned}$$

□

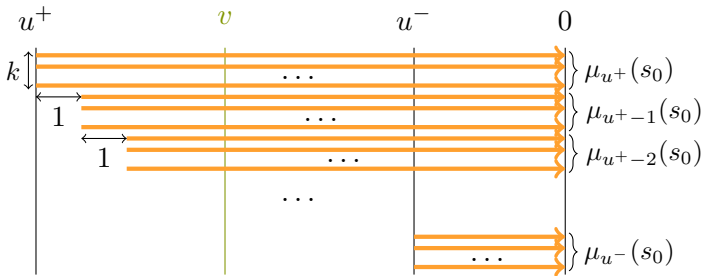



Figure 4.14: Parallel simulation of trajectories estimating $(\mu_u(s_0))_{u=u^-}^{u^+}$

### 4.5.3 More on Simulation.

In the previous section, we describe a theoretical method to produce a confidence interval framing of $\mu_\tau(s_0)$. Even if our system is a CTMC, the distribution required for the importance sampling method comes from the embedded DTMC analysis.

Estimating $\mu_\tau(s_0)$ requires to apply $(u^+ - u^-)$ times the algorithm for a bounded DTMC. When $\tau\lambda$ is big this becomes intractable.

#### Simultaneous Simulation

Trajectories for all $u$ between $u^-$ and $u^+$ are simultaneously simulated step by step in order to use the same vector of probability $\mu_v^\bullet$ at each step. This allows $\mu_v^\bullet$ to be computed only once and used for all trajectories devoted to the computation of $\mu_u(s_0)$ where $u \geq v$. The simulation starts with the generation of $k$ trajectories devoted to the computation of $\mu_{u_+}(s_0)$. After one step of simulation of these $k$ trajectories, we generate $k$ new trajectories devoted to the computation of $\mu_{u^+-1}(s_0)$. We resume the simulation with these $2k$ trajectories and so on until the generation of the trajectories devoted to the computation of $\mu_{u^-}(s_0)$. Figure 4.14 illustrates this point.

#### Reuse of Simulation

Further optimization improvement is possible in order to reduce the width of confidence interval of each $\mu_u(s_0)$.

One can use the same simulation to estimate several $\mu_u(s_0)$. Given a trajectory that reaches state $s_+$ or $s_-$ in $v$ steps, all estimators of $\mu_{v'}(s_0)$ with $0 < v' \leq u$ can take into account this trajectory. More precisely each estimator with $v' < v$ adds a realization with likelihood equal to 0 and each estimator with $v' \geq v$ adds a realization with likelihood equal to $W_{s_0}$.

In an ideal setting, one only have to perform the simulation for estimating $\mu_{u^+}(f(s_0))$ to estimate all $(\mu_u(f(s_0))_{u^-}^{u^+}$. Experimentally this idealistic case can happen (see Section 7 for an example), but in general there is a difficulty in the fact that values of $\mu_v(s_0)$ for small $v$ compared to $u_+$ can correspond

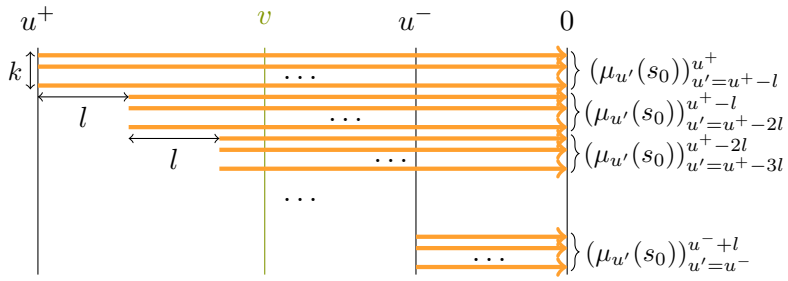



Figure 4.15: Parallel simulation of trajectories estimating $(\mu_u(s_0))_{u=u^-}^{u^+}$ with reuse of trajectories

to probabilities of rare events and in that case are not accurately estimated with importance sampling probability distribution computed for $\mu_{u^+}$.

Recall that $\mu_u(s_0)$ is an increasing function of $u$. Thus if $v$ is a lot smaller than $u$ then $\mu_v(s_0)$ can be a lot smaller than $\mu_u(s_0)$. The likelihood of a trajectory simulated under an importance sampling built for estimating $\mu_{u^+}(s_0)$ without normalization takes value in $0 \cup [\mu^\bullet_{u^+}(f(s_0)); \infty[$. Thus the probability for a trajectory to reach $s^+$ knowing it reaches $s^+$ or $s^-$ in $v$ steps is smaller than $\mu_v(s_0)/\mu^\bullet_{u^+}(f(s_0))$ which can be too small. This mean that almost all trajectories will not reach $s_+$ in $v$ steps and thus the estimation of $\mu_v(s_0)$ is still subject to the rare event problem.

A more elaborate approach is necessary to prevent this issue. The set of integers $\{u^-, \dots, u^+\}$ can be split into several parts. Let $l$ be an integer smaller than $u^+$. Instead of simulating $k$ trajectories for each $u \in \{u^-, u^- + 1, \dots, u^+\}$, we can perform $k \cdot l$ simulations for each $u \in \{u^-, u^- + l \dots, u^+\}$. The total number of simulations is unchanged but each $\mu_u(s_0)$ receive contribution from at most $k \cdot l$ trajectories.

### Guaranteed Variance Reduction while Reusing Simulation

In the guaranteed reduction setting, some adaptations are required to maintain guaranteed variance reduction while reusing simulation. Consider an importance sampling with sink state build for estimating $\mu_u(s_0)$. When one of its trajectory reaches the state $s_+$ in $v$ steps with $v \leq u$, the likelihood $W_{s_0}$ of this trajectory is equal to $\mu^\bullet_u\big(f(s_0)\big)$ instead of $\mu^\bullet_v\big(f(s_0)\big)$. The distribution of $W_{s_0}$ is still bivaluated when simulations are reused and takes value in $\{0, \mu^\bullet_u\big(f(s_0)\big)\}$. Therefore, confidence intervals can still be build.

Similarly, when an importance sampling with normalization build for estimating $\mu_u(s_0)$ is used, the random variable $W_{s_0}$ of a trajectory reaching $s_+$ in $v$ takes value in $[0; \mu^\bullet_u\big(f(s_0)\big)]$. Chernoff-Hoeffding bounds still apply and confidence intervals can be computed.

## 4.6 From Model Checking to Reachability

Previous sections focus on time-bounded and time-unbounded reachability problems. In this section, we show how more general model checking problems can be transformed into such reachability problems. To achieve this goal we extend the definition of a Markov chain with a set of atomic propositions $AP$ and a labeling function $L : S \to 2^{AP}$.

### 4.6.1 From Until Formula to Reachability

Consider a model checking problem on model $\mathcal{M}$ given as a DTMC or a CTMC against a formula $\phi = a \,\mathbf{U}^{u}\, b$. Where the constant $u$ can either denote $\infty$ or an integer if $\mathcal{M}$ is a DTMC, or a real if $\mathcal{M}$ is a CTMC. Here we are interested in transforming this model checking problem to a reachability problem.

The Markov chain $\mathcal{M}$ is transformed by lumping together all states where $b$ holds ($S_b$) in a new state that we called $s_+$. All states where $\neg a \wedge \neg b$ holds ($S_{\overline{a b}}$) are lumped together in state $s_-$. States $s_+$ and $s_-$ are made absorbing. All remaining states ($S_{a\bar{b}}$) are left unchanged. If a state is no longer reachable from the initial state due to this transformation, it is removed from the model. In the case $u = \infty$, the formula $\phi$ can be rewritten as $\phi = a \,\mathbf{U}\, b$. The corresponding DTMC $\mathcal{C}_\phi$ is then given by:

**Definition 30 (From Until Formula to Reachability)**

Given a DTMC or a CTMC $\mathcal{C}$ and a formula $\phi = a \,\mathbf{U}\, b$, the Markov chain $\mathcal{C}_\phi$ is defined by:

- $S_\phi = S_{a\bar{b}} \cup \{s_-, s_+\}$
- $s_-, s_+$ are *absorbing* states:
  $\mathbf{P}_\phi(s_-, s_-) = \mathbf{P}_\phi(s_+, s_+) = 1$
- $\forall s \in S_{a\bar{b}},\ s' \in S_\phi,\ \mathbf{P}_\phi(s, s') = \begin{cases} \mathbf{P}(s, s') & \text{if } s' \in S_{a\bar{b}} \\ \sum_{s'' \in S_{\overline{a b}}} \mathbf{P}(s, s'') & \text{if } s' = s_- \\ \sum_{s'' \in S_b} \mathbf{P}(s, s'') & \text{if } s' = s_+ \end{cases}$

This definition applies both to DTMC and CTMC: as states lumped together are absorbing ones, their exit rate is irrelevant.

When $u = \infty$, in order to use the method described in section 4.3 an additional hypothesis is required: $\mathcal{M} \models \mathbb{P}_{=1}[Fb]$ then $\mathcal{C}_\phi$ is a BDTMC and the method applies.

When $u$ is an integer or a real $(\mathcal{C}_\phi, u)$ is an instance of time bounded reachability problems that can be solved using methods of Sections 4.4 and 4.5.

### 4.6.2 From Model Checking Against Finite State Automaton to Reachability

We adapt our method to model checking problems expressed by an automaton. Let $\mathcal{M}$ be a model taking as semantics a discrete or continuous time Markov chain. Let $AP$ be the set of atomic propositions labeling the states of $\mathcal{M}$ and $E$ the set of events labeling the transitions of $\mathcal{M}$. Let $\mathcal{A}$ be a finite-state automaton over the alphabet $AP \times E$. Let $u$ be a time bound taking values in $\mathbb{N} \cup \{\infty\}$, if $\mathcal{M}$ has discrete time semantics, and values in $\mathbb{R}^+ \cup \{\infty\}$ otherwise.

The following definition is a classical definition of automaton.

**Definition 31 (Finite State Automaton)**

A finite state automaton $\mathcal{A}$ is a tuple $(Q, Q_0, \Sigma, T, F)$, where

- $Q$ is a finite set of locations,
- $Q_0 \subset Q$ is the non empty set of initial locations,
- $\Sigma$ is a finite alphabet,
- $T \subset Q \times \Sigma \times Q$ is a set of transitions,
- $F \subset Q$ is the set of final locations,

and $\forall q \in F,\ a \in \Sigma,\ \exists q' \in F\ s.t.\ (q, a, q') \in T$, i.e., the set of final locations is absorbing and complete.

We define the synchronized product of the Markov chain underlying model $\mathcal{M}$ and the automaton $\mathcal{A}$ as follows: as the automaton is not deterministic this synchronization build the powerset construction of the automaton,

**Definition 32 (Synchronized Product with an Automaton)**

Given an enriched DTMC or a CTMC $\mathcal{C} = (S, s_0, E, \delta, p)$, a labeling function $L : S \to 2^{AP}$ and a finite state automaton $\mathcal{A} = (Q, Q_0, E \times 2^{AP}, T, F)$, the enriched synchronized Markov chain $\mathcal{C}_{\mathcal{A}} = (S^{\otimes}, s_0^{\otimes}, E^{\otimes}, \delta^{\otimes}, p^{\otimes})$ is defined by:

- $S^{\otimes} = (S \times 2^Q) \cup \{s_+, s_-\}$
- $s_0^{\otimes} = (s_0, Q_0)$
- $E^{\otimes} = E$
- $\delta^{\otimes}(s_-, -) = s_-,\ \delta^{\otimes}(s_+, -) = s_+$

$$\delta^{\otimes}((s, Q_1), e) = \begin{cases} s_+ \text{ if } (Q_1 \times \{(e, L(s))\} \times Q_f) \cap T \neq \emptyset \\ s_- \text{ if } (Q_1 \times \{(e, L(s))\} \times Q) \cap T = \emptyset \\ (\delta(s, e), (Q_1 \times \{(e, L(s))\} \times Q)) \text{ otherwise} \end{cases}$$

- $p^{\otimes}((s, Q_1), e) = p(s, e)$

The probability to reach $s_+$ in the synchronized product $\mathcal{C}_{\mathcal{A}}$ is equal to the probability that a trajectory of $\mathcal{C}$ is accepted by the automaton.

When $u = \infty$, if $\mathcal{C}_{\mathcal{A}}$ is BDTMC, then the method described in Section 4.3 applies. When $u$ is finite, the corresponding time bonded method applies as well.

The model checking of Markov chain against such an automaton can be related to the model checking of asCSL as defined in [6]. In asCSL, one computes the probability of a path to be accepted by a finite automaton given as a regular expression, in a given time interval. Compared to asCSL, the logic we present (1) restricts the interval of the path operator to intervals containing 0 and (2) disallows the nesting of probabilistic operator. These restrictions are due to the difficulty to deal with steady state probabilities, nested operators and infinite paths in statistical model checking.

## 4.7 Conclusion

In this chapter, we present a theoretical framework allowing to estimate probabilities of rare events using importance samplings. Using numerical computations on a reduced model, important samplings are produced which, with additional hypotheses, ensure the reduction of the variance. This method applies to discrete and continuous models in time-bounded and time-unbounded settings and allows to compute reachability properties or more general probabilistic model checking problems. It is the first method to ensure the reduction of variance and producing confidence intervals which do not relies on asymptotic hypotheses.

To this point, the reduced model is build manually which can be tedious. In the following chapter, an automated method is proposed for some classes of systems. Moreover, experiments are conducted in Chapter 6.7 to assess the efficiency of these importance samplings. Finally, in Chapter 7, a case study is performed on a biological system using this method to estimate the avancement of a reaction.

# Chapter 5

# Patterns for Stochastic Bounds

## 5.1 Introduction

In Chapter 4 we described a method for building an importance sampling that guarantees a variance reduction and produces a confidence interval. The main drawback of this method is that it requires a human intervention to design an abstraction of the studied model. In this chapter, we present a class of systems for which a stochastically bounding model with smaller state space can be automatically built and can be used as a reduced model.

Stochastic bound relies on an ordering of the state space of the system to be studied. In case of a DTMC or a CTMC we assume that the state space $S$ is equal to $\{1, \ldots, n\}$ where 1 and $n$ are the two absorbing states. We are interested in computing the probability to reach 1 or $n$ from a non absorbing state in time bounded or time unbounded setting. Bounding models are Markov chains where the probability to reach $n$ (respectively 1) is greater (respectively smaller) than in the initial model from a non absorbing state. Moreover, these properties are ensured without computing the transient or steady state distributions. Bounding models satisfy two conditions, stochastic ordering and stochastic monotony. Stochastic ordering is a property between the original model and the bounding one, whereas stochastic monotony is an intrinsic property of the bounding model. Both of them can be checked by studying the transition probability matrix or the infinitesimal generator of the Markov chain.

Bounding models are used to analyze systems with large state spaces when the properties of interest can not be computed either numerically due the size of the system or statistically due to the rare event problem or difficulties to estimate steady state probabilities. Bounding models are built with additional constraints that make the bounding model lumpable, yielding a smaller state space. Numerical methods are applied on the bounding model

to compute an upper or a lower bound of the value of interest.

Bounding models are classically used to analyze a Markov chain with a large state space. In [2, 33, 41] algorithms computing bounding models from the transition probability matrix of a Markov chain are presented. These algorithms take also as input an equivalence relation over the state space, states in this relation are aggregated in the bounding model. In [73] a stochastic system is defined as a tensor product of several Markov chains yielding a compact representation for a large Markov chain. These representations are used (for example in [34]) to build bounding models by analyzing each component instead of the whole system.

The systems considered in this chapter are instantiations of a same pattern. All instances share a common structure but can have different stochastic behaviors. States of the pattern are totally ordered and this order induces a partial order on states of the global system. A pattern that stochastically bounds all the instantiations is built yielding a bounding model for the whole system. The interaction between the instances are constrained to ensure the monotonicity of the bounding model. Furthermore this bounding model is a reduced model that can be used to build importance sampling with guaranteed variance reduction.

In Section 5.2, the class of systems that we consider is defined. In Section 5.4, a coupling on the state space of systems is defined. In Section 5.5, this coupling is used to build reduced models with guaranteed variance.

## 5.2 Framework

Our model is a parallel composition with partial synchronization of a finite number of copies of one automaton that we call a *pattern*. This pattern is divided in a finite numbers of zones. Several processes move each on its own pattern.

### 5.2.1 Syntax

We first define the class of automata we shall use for that purpose.

**Definition 33 (Pattern)**

A pattern $\mathcal{A} = \langle Tp, \Sigma, L, Z, V, \delta, \{\omega_k\}_{k \in Tp} \rangle$ is defined by:

- $Tp = \{1, \ldots, K\}$ is a finite set of process types of cardinal $K$.
- $\Sigma = \Sigma_f \uplus \Sigma_b$ is the set of labels partitioned into two subsets.
- $L = \{1, \ldots, n\}$ is a finite set of local states of cardinal $n$.

- $Z = (t_i)_{i=1}^{\zeta+1}$ with $\zeta$ positive integer, $(t_i)_{i=1}^{\zeta+1}$ an increasing sequence of states such that $t_1 = 1$ and $t_{\zeta+1} = n$. This sequence divides the set of local states in subsets of contiguous states called zones : $\mathtt{z}_{2i-1} = \{t_i\}$ and $\mathtt{z}_{2i} = \{t_i, \ldots, t_{i+1}\}$. We define the boundary of a zone $\mathtt{z}$, denoted by $\partial\mathtt{z}$, as $\partial\mathtt{z} = \mathtt{z} = \{t_i\}$, if $\mathtt{z} = \mathtt{z}_{2i-1}$ and $\partial\mathtt{z} = \{t_i, t_{i+1}\}$ if $\mathtt{z} = \mathtt{z}_{2i}$. Thus $\zeta$ is the number of even zones.

- $\delta : L \times \Sigma \to L$ is the local transition function. It satisfies : $\forall l \in L, \forall e \in \Sigma$

  1. $e \in \Sigma_f \Rightarrow \delta(l, e) \geq l$ and we say that $e$ labels a forward moves.
     $e \in \Sigma_b \Rightarrow \delta(l, e) \leq l$ and we say that $e$ labels a backward moves.
  2. $\forall 1 \leq i \leq \zeta$, $l \in \mathtt{z}_{2i} \setminus \mathtt{z}_{2i+1} \wedge e \in \Sigma_f \Rightarrow \delta(l, e) \in \mathtt{z}_{2i}$.
     $l \in \mathtt{z}_{2i} \setminus \mathtt{z}_{2i-1} \wedge e \in \Sigma_b \Rightarrow \delta(l, e) \in \mathtt{z}_{2i}$.

- $V = (v_l^k)_{l \in L, k \in Tp}$ is a finite set of integer variables.

- $\forall k \in Tp$ $\omega_k : L \times \Sigma \to (\mathbb{N}^V \to \mathbb{R}_{\geq 0})$ is a "rate parameter" function. It associates to a transition a function from variables to non-negative real numbers.

The pattern defines a dynamic system where several processes are moving, each on its own automaton. This automaton is depicted as a line whose states are indexed from the leftmost 1 to the rightmost $n$ and whose transition function is $\delta$. Transitions with same label go in the same direction (condition 1); if it goes to the right, we call it a forward move and if it goes to the left, we call it a backward move. This line is divided in $\zeta$ contiguous zones (the even ones). Condition 2 ensures that a process in a even zone cannot leave it without passing by the boundary of this zone, which corresponds to a zone with odd index.

Condition 2 ensures that a transition keeps a process in its current zone or moves it to an adjacent one.

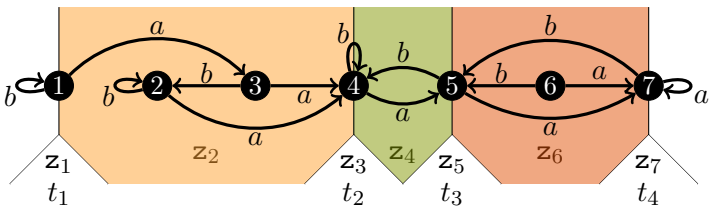



Figure 5.1: A pattern

The stochastic behavior of the whole system is a continuous time Markov chain where each process is in concurrence with the others. The stochastic behavior on each process is defined by the rate function $\omega_k$ which returns the rate of a transition. This rate depends on several parameters:

1. the type of the process which is local and constant for a process,
2. the location of the pattern and the label of the transition which are local to the process,
3. The valuation of the set of variables $V$ which is global to the whole system.

To build the stochastic semantics, the value of a variable $v_l^k$ in the set $V$ is set to the number of processes of type $k$ which current location is greater than $l$ in the whole system. These will formally be defined in Section 5.2.2.

A process interacts with other processes in two ways: First through synchronization at zone boundaries, second by the means of the set of variables $V$. After each transition in the whole system, the value of a variable $v_l^k \in V$ is set to the number of processes of type $k$ which current location is greater than $l$. These two interactions will formally be defined in Section 5.2.2.

**Remark 16** *For this definition the pattern must be a complete automaton, i.e. for all $l \in L$ and all $e \in \Sigma$, $\delta(l, e)$ is defined. A model can be completed by adding self-loops on states where transitions labeled by e are not defined. The rate of such a self-loop can be arbitrary, for instance the minimal rate of transitions of same label for the same process ($\omega(l, e) = min_{l' \in L}(\omega(l, e))$). This transformation does not affect the stochastic behavior of the system.*

**Example 7** *An example of pattern is depicted on Figure 5.1. Alphabets for this pattern are: $\Sigma_f = \{a\}$ and $\Sigma_b = \{b\}$, thus all forward jumps are labeled by a and all backward jumps are labeled by b. The set of states is $\{1, 2, \ldots, 7\}$. Zones are delimited by the sequence $Z = \{1, 4, 5, 7\}$ and zones are: $\mathtt{z}_1 = \{1\}$, $\mathtt{z}_2 = \{1, 2, 3, 4\}$ , $\mathtt{z}_3 = \{4\}$, $\mathtt{z}_5 = \{4, 5\}$ , $\mathtt{z}_6 = \{5, 6, 7\}$ and $\mathtt{z}_7 = \{7\}$. It is easy to check that the second condition on transitions holds: for every transition there is an even zone that contains its source and destination states.*

### 5.2.2 Operational Semantic

In order to build a fully stochastic model from the pattern defined in the previous section one needs to add additional information to the model. The number of processes and their type must be specified. In each even zone processes are *joined* together in *block*, the number of processes per block and the total number of blocksin each region is specified. On boundaries

of zone, processes *join* to form a block when they enter a zone and they *disjoint* when they exit a zone. In an odd zone the size of blocks are always of 1. Formally this is defined as follows:

**Definition 34 (instance)**

An instance is a pattern specified with the following list of parameters:

- $I = \{1, \ldots, m\}$ denotes the set of processes in the system.
- $\kappa : I \to Tp$ denotes the mapping from the set of processes to the set of types of processes. We denote by $\{m_i\}_{1 \leq i \leq K}$ the number of processes with type $i$, with $\sum_{1 \leq i \leq K} m_i = m$ the total number of processes.
- $\{a_{2i}\}_{1 \leq i \leq \zeta}$ where $a_{2i}$ is the cardinality of each processes per block in zone $\mathtt{z}_{2i}$. By convention, for all $0 \leq i \leq \zeta$, $a_{2i+1} = 1$. For any zone indexed by $i$ we say that its *multiplicity* is $a_i$.
- $\{b_{2i}\}_{1 \leq i \leq \zeta}$ where $b_{2i}$ is the maximal number of allowed blocks in zone $\mathtt{z}_{2i}$. By convention, for all $0 \leq i \leq \zeta$, $b_{2i+1} = m$.
- $\mathfrak{f} : \mathcal{P}(I) \times I \times \{1, \ldots, \zeta\} \to \big(\mathcal{P}(I) \to [0, 1]\big)$ is the probabilistic *block assigning function*. It ensure that if $f = \mathfrak{f}(E, i, z)$ , $i \in E$, $z$ is an even number of zone and $|E| \geq a_z$ then the sum of all $f(F)$ over $F$ such that $i \in F$ and $|F| = a_z$ is equal to 1. For all $F$ such that $i \notin F$ or $|F| \neq a_z$ then $f(F) = 0$. That is $f$ is a discrete probabilistic distribution over the possible blocks.

An instance yields an enriched CTMC. The CTMC of an instance is built as a transition system based on the product of copies of the pattern $\mathcal{A}$, one for each process in $I$ with additional constraints due to synchronization between processes. Inside an even zone (let say $\mathtt{z}_{2i}$), a process is associated with $a_{2i} - 1$ other processes, all in $\mathtt{z}_{2i}$, in some way that we shall explain later. We call the set of these associated processes a block. Processes in a block inside zone $\mathtt{z}_{2i}$ cannot enter or leave the zone without being synchronised; this synchronization happens when all processes in the block are located on the same boundary of the zone. There cannot be more than $b_{2i}$ blocks in zone $\mathtt{z}_{2i}$, therefore when the zone is full, processes waiting to form a new block and enter the zone must wait on the boundary.

**Definition 35 (Uniform block assigning function)**

A block assigning function $\mathfrak{f}$ is called uniform if for all $0 < g \leq \zeta, E \in \mathcal{P}(I)$ such that $|E| \geq a_{2g}, i \in E, F \subset E$ such that $i \in F$ and $|F| = a_{2g}$ the following holds:

$$\mathfrak{f}(E, i, 2g)(F) = \frac{1}{\binom{|E|-1}{a_{2g}-1}}$$

That is the probability distribution $\mathfrak{f}(E, i, 2g)$ is uniform over all valid blocks.

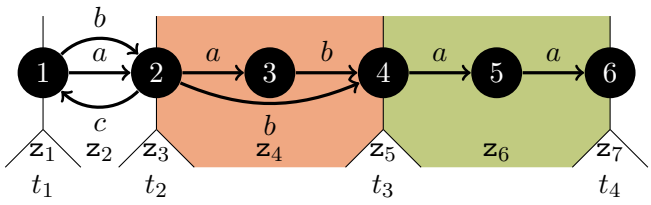



Figure 5.2: pattern for a database example

**Example 8** *Figure 5.2 is an example of pattern for a distributed database example with synchronization. The system contains $N$ databases. When a database receives an update it must synchronize it to all the others. Initially each database is in an idle state that is location 1, then one database receives a modification request with label $a$ and moves to location 2. This request may be denied with transition $c$. Otherwise the database enters the critical zone $\mathtt{z}_4$ and commits the change in two steps. Finally the database synchronizes with the others by entering zone $\mathtt{z}_6$. Transitions labeled by $b$ in location 1 and 2 are used by a database which did not receive a modification request to reach the location 4 where they can synchronized together. At the end of the synchronization databases reach state 6.*

*There are several database types depending on the frequency at which they receive modification requests. To simplify we suppose that type 1 database receive a high amount of modification request while type 2 receive a low amount of modification. We also suppose that half of the database are of type 1 and half of type 2. The rate parameter function is as follows:*

$$\begin{array}{ll} \omega_1(1,a) = 1 & \omega_2(1,a) = 0.5 \\ \omega_{1/2}(_,a) = 1 & \omega_{1/2}(1,b) = 10\min\left(1, v_3^1 + v_3^2\right) \\ \omega_{1/2}(2,c) = 0.5 - 0.5\min\left(1, v_3^1 + v_3^2\right) & \omega_{1/2}(3,b) = 10 \\ \omega_{1/2}(2,b) = 10\min\left(1, v_3^1 + v_3^2\right) & \end{array}$$

*The multiplicity of all zones except $\mathtt{z}_4$ is 1. The multiplicity of $\mathtt{z}_4$ is $a_4 = N$. The maximal number of allowed blocks is $N$ in each zone except $\mathtt{z}_6$ where $b_6 = 1$.*

*The block assigning function is the uniform one.*

From an instance $\left(\mathcal{A}, I, \kappa, \{a_{2i}\}_{1\le i\le\zeta}, \{b_{2i}\}_{1\le i\le\zeta}\right)$, the annotated CTMC $\mathcal{M} = (S, \Sigma, \Delta, \Omega, \mathfrak{f})$ is defined as follows:

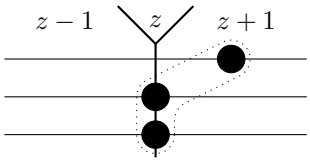



Figure 5.3: Example of synchronization. Each horizontal line represents the state space of a pattern. Bullets represent actual states of each pattern. The dotted line represents the joint block.

**The state space $S$.**

Let $\mathsf{Part}(I)$ be the set of partitions of $I$. The set of global states $S$ is the subset of $(L \times \{1, \ldots, 2\zeta+1\})^I \times \mathsf{Part}(I)$ of tuples $s = ((l_i, z_i)_{i \in I}, \{I_1, \ldots, I_u\})$ such that:

- $\forall i \in I,\ l_i \in \mathtt{z}_{z_i}$ that is $z_i$ is the index of a zone containing local state $l_i$.
- $\forall 1 \leq j \leq u,\ \forall i, i' \in I_j,\ z_i = z_{i'}$. The subset $I_j$ fulfills $|I_j| = a_{z_i}$. Given $i \in I$ there is a unique $j$ such that $i \in I_j$. The $i^{th}$ process is synchronized with $a_{z_i} - 1$ other processes which are all in the same zone, these processes being those of $I_j \backslash \{i\}$. We denote by $p_i$ the *block* containing $i$: we have $p_i = I_j$.

We also use the redundant notation for state $s$, $(l_i, z_i, p_i)_{i \in I}$ where $p_i$ is the block containing $i$. Depending of the context one notation can be more convenient than the other.

For further use, we introduce two special states.

- $s_0 = ((1,1)_{i \in I}, \{\{1\}, \{2\}, \ldots, \{m\}\})$ so is the initial global state.
- $s_f = ((n, 2\zeta+1)_{i \in I}, \{\{1\}, \{2\}, \ldots, \{m\}\})$ so is the final global state.

The partition $\{I_1, \ldots, I_t\}$ is the set of equivalence classes of the equivalence relation defined by synchronizations.

**The set of events $E$.**

Events in our transition system are pairs in $\Sigma \times I$ of an event of the pattern automaton and the index of the process performing the event.

**The successor function $\Delta$.**

A transition in such a system proceeds most of the time in an asynchronous way by applying a local transition of the pattern on a single process. This is true as long as the moving process remains inside a zone. But boundaries of zones, which are zones of length zero, force synchronization between some

processes: A process cannot enter a zone $\mathtt{z}$ of multiplicity $\mu > 1$ without being associated with $\mu - 1$ other processes in a block, these processes being all in the same local state in $\partial\mathtt{z}$. Symmetrically, a process cannot leave this zone unless all processes in the corresponding block are ready to leave, which requires that they all are in the same local state in $\partial\mathtt{z}$.

Due to synchronization at the creation of blocks the successor function returns a set of states. When there is no creation of block the set is a singleton, otherwise the set contains a state for each choice of blocks.

The function $\mathfrak{f} : \mathcal{P}(I) \times I \times \{1, \ldots, \zeta\} \to \mathcal{P}(I) \to [0, 1]$ specifies how processes are chosen to build a block. This function takes as first argument a subset of $I$ corresponding to processes ready for synchronization, as second argument the index of the process initiating the transition and the index of the zone in which the new block is created. The function returns a discrete distribution probability over the set of subsets of the suitable cardinality.

From a global state $s = (l_j, z_j, p_j)_{j \in I}$ a global transition with label $e \in \Sigma$ and process $i \in I$ maps to a set of state $\Delta(s, (e, i))$ by applying the local transition $\delta$ of the pattern on state $l_i$ with label $e$ if possible and applying synchronization rules to update the block structure. These rules are precisely described in table 5.1 for processes lying in an even zone in this case $\Delta(s, (e, i))$ is a singleton, and in table 5.2 for processes lying in an odd zone, in this case $\Delta(s, (e, i))$ is set of state. If a transition is not possible due to synchronization, we say that the corresponding process is blocked and the transition is replaced by a self-loop. For the sake of conciseness we use the notation $\Delta(s, e, i)$ instead of $\Delta(s, (e, i))$. We denote $\mathtt{z}$ the zone $z_i$, containing $l$ and the global state $s^+ = \Delta(s, e, i) = (l_j^+, l_j^+, p_j^+)_{j \in I}$.

**The rate function $\Omega$.**

The rate function gives the stochastic behavior of the system by specifying rates for each transition of each process. As the model is a CTMC the waiting time in each global states follows an exponential distribution whose rate is equal to the sum of all rates of transitions of each process. The probability to take a transition is its rate divided by the sum of all rates of other transitions.

When there is a creation of block, several output states are possible whose probability are defined by $\mathfrak{f}$. The $\Omega$ function only specifies the rate for individual processes the final rate matrix is defined latter.

The global rate function is based on the local rate functions $\omega_k$ by evaluating a valuation for the variables of the pattern on a global state $s$. The variables of the pattern are evaluated as follows:

$$\forall l \in L,\ v_l^t((l_j, z_j, p_j)_{j \in I}) = \#\{j \in I \mid l_j \geq l \wedge \kappa(j) = t\}$$

This means that the variable $v_l^t$ takes for value the number of processes of type $t$ in a state bigger than $l$.

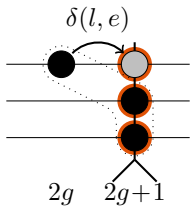
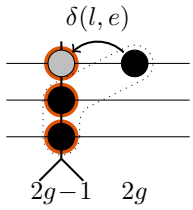
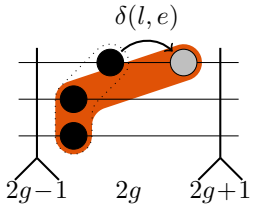
<table>
<tr><td colspan="3">Disjoining blocks with a forward move<br>$z_i$ is even: $z_i = 2g$ of multiplicity $a_{2g}$, $\delta(l,e) = t_{g+1}$</td></tr>
<tr><td>$\forall j \in p_i \setminus \{i\}, l_j = t_{g+1}$</td><td>$\forall j \in p_i, z_j^+ = 2g{+}1$<br>$p_j^+ = \{j\}$<br>$l_j^+ = t_{g+1}$</td><td>
</td></tr>
<tr><td colspan="3">Disjoining blocks with a backward move<br>$z_i$ is even: $z_i = 2g$ of multiplicity $a_{2g}$, $\delta(l,e) = t_g$</td></tr>
<tr><td>$\forall j \in p_i \setminus \{i\}, l_j = t_g$</td><td>$\forall j \in E, z_j^+ = 2g{-}1$<br>$p_j^+ = \{j\}$<br>$l_j^+ = t_g$</td><td>
</td></tr>
<tr><td colspan="3">No change of zone, $z_i$ is even: $z_i^+ = z_i = 2g$</td></tr>
<tr><td>$\delta(l,e) \notin \partial \mathbf{z}_{2g}$<br>or $\exists j \in p_i \,/\, l_j \neq \delta(l,e)$</td><td>$z_i^+ = 2g$<br>$l_i^+ = \delta(l,e)$<br>$p_i^+ = p_i$</td><td>
</td></tr>
<tr><td colspan="3">The move is a self-loop</td></tr>
<tr><td>None of the previous conditions are satisfied</td><td>$l_{i_j}^+ = l$</td><td></td></tr>
</table>

Table 5.1: Table of transitions for even zones

For a global state $s \in S$, an event $e \in \Sigma$ and a process $i \in I$,

$$\Omega(s,(e,i)) = w_{\kappa(i)}(l_i,e)(v_1^1(s), v_2^1(s), \ldots, v_n^1(s), v_1^2(s), \ldots, v_n^K(s)).$$

Like for the successor function we use also the notation $\Omega(s,e,i)$ as $\Omega(s,(e,i))$.

### The transition rate matrix

The transition rate matrix is obtained by combining the $\Omega$ function and the $\mathfrak{f}$ function. For all $s = (l_j, l_j, p_j)_{j\in I} \in S$, $e \in \Sigma$, $i \in I$ and $F \in \mathcal{P}(I)$ the $\Omega^*(s,(e,i,F))$ is defined as follows:

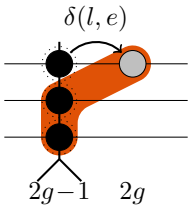
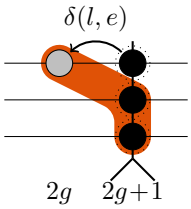
<table>
<tr><td colspan="3">Joining blocks with a forward move<br>$z_i$ is odd: $z_i = 2g-1$, $l_i = t_{g-1}$, $p_i = \{i\}$, $\delta(l,e) \in \mathtt{z}_{2g}$ of multiplicity $a_{2g}$</td></tr>
<tr><td>Let $E$ be the subset of indices $j$ /<br>$z_j = 2g-1$,<br>$l_j = t_g$<br>$p_j = \{j\}$<br>then $i \in E$ and $|E| \geqslant a_{2g}$<br>Let $F = \{j,\ z_j = 2g\}$<br>$|F| < b_{2g}a_{2g}$</td><td>Let $F$ such that $\mathfrak{f}(E,i,2g)(F) > 0$<br>$\forall j \in F,\ z_j^+ = 2g$<br>$l_i^+ = \delta(l,e)$ and<br>$l_j^+ = t_g$ if $j \neq i$<br>$p_j^+ = F$</td><td></td></tr>
<tr><td colspan="3">Joining blocks with a backward move<br>$z_i$ is odd: $z_i = 2g+1$, $l_i = t_g$, $p_i = \{i\}$, $\delta(l,e) \in \mathtt{z}_{2g}$ of multiplicity $a_{2g}$</td></tr>
<tr><td>Let $E$ be the subset of indices $j$ /<br>$z_j = 2g+1$,<br>$l_j = t_g$<br>$p_j = \{j\}$<br>then $i \in E$ and $|E| \geqslant a_{2g}$<br>Let $F = \{j,\ z_j = 2g\}$<br>$|F| < b_{2g}a_{2g}$</td><td>Let $F$ such that $\mathfrak{f}(E,i,2g)(F) > 0$<br>$\forall j \in F, z_j^+ = 2g$<br>$l_i^+ = \delta(l,e)$ and<br>$l_j^+ = t_g$ if $j \neq i$<br>$p_j^+ = F$</td><td></td></tr>
<tr><td colspan="3">The move is a self-loop</td></tr>
<tr><td>None of the previous conditions are satisfied</td><td>$l_{i_j}^+ = l$</td><td></td></tr>
</table>

Table 5.2: Table of transitions for odd zones

- If $|\Delta(s,(e,i))| = 0$ then $\Omega^*(s,(e,i,F)) = 0$
- If $|\Delta(s,(e,i))| = 1$ then $\Omega^*(s,(e,i,F)) = \Omega(s,(e,i))$.
- Otherwise $\Omega^*(s,(e,i,F)) = \Omega(s,(e,i))\mathfrak{f}(\{j,z_j = z_i\},i,z_i)(F)$

Similarly the $\Delta^*(s,(e,i,F))$ is defined as follows:

- If $|\Delta(s,(e,i))| = 0$ or $\mathfrak{f}(\{j,z_j = z_i\},i,z_i)(F) = 0$ then $\Delta^*(s,(e,i,F)) = s$
- If $|\Delta(s,(e,i))| = 1$ then $\Delta^*(s,(e,i,F)) = \Delta(s,(e,i))$
- Otherwise $\Delta^*(s,(e,i,F)) = \{(l'_j,l'_j,p'_j)_{j\in I}) \in \Delta(s,(e,i))$ s.t. $p'_i = F\}$

Then the rate transition matrix is defines by

$$R(s,s') = \sum_{e\in E, i\in I, F\in\mathcal{P}(I)} \mathbb{1}_{\{s'=\Delta^*(s,(e,i,F))\}}\Omega^*(s,(e,i,F))$$

The following lemma shows some arithmetic property of an instance.

**Lemma 1**

For a global state $s = (l_i, z_i, p_i)_{i\in I}$ reachable from the initial state and a zone indexed by $z$ of multiplicity $a_z$, the following holds:

$$\#\{i \mid z_i \geq z\} \text{ is a multiple of } a_z$$

*Proof:*
By induction on the length of a path starting from the initial state:
In the initial state $\forall z > 1, \#\{i, z_i \geq z\} = 0$
and for $z = 1, a_1 = 1$.
Let $s$ be a state satisfying the induction hypothesis. For any $z$, a transition from $s$ satisfies:

- A block is created in the zone of multiplicity $a_z$ by moving a process forward. In this case the cardinality of $\{i, z_i \geq z\}$ is increased by $a_z$.
- A blocks of multiplicity $a_z$ disappears by moving a process backward, then the cardinality of $\{i, z_i \geq z\}$ is decreased by $a_z$.
- Otherwise the cardinality of $\{i, z_i \geq z\}$ is unchanged.

□

**Corollary 11**

If there exists an execution from the initial state to the final state then $m$ is a multiple of multiplicity $a_i$ for any $i \in I$.

## 5.3 Symmetries

**Definition 36 (Symmetric instance )**

An instance is *symmetric* if $|\kappa(I)| = 1$

In this case we omit the type index for the rate functions and variables.

This definition makes sense because if there is only one type of processes the rate function $\Omega(s, e, i)$ is equal to $\omega(l_i, e)(v_1, \ldots, v_n)$ and all processes

in a same state behave the same way. This will be formalized below by specifying an equivalence relation over states where two global states are equivalent if they are identical up to the processes numbering, that is each local state gets the same number of processes and if there is a one to one correspondence between the blocks. Then we will show that a symmetric instance is strongly lumpable on this equivalence relation [59]. Afterward one builds the quotient of the state space $S$ by the equivalence relation and thus obtains a smaller system with the same stochastic behavior.

**Definition 37 (Equivalence relation)**

Let $s = ((l_i, z_i, p_i)_{i\in I})$ and $s' = ((l'_i, z'_i, p'_i)_{i\in I})$ be two global states. Let $\sigma$ be in $\mathfrak{S}_m$ We say that $s \sim^\sigma s'$ if:

$$\forall i \in I \ , \ l_i = l'_{\sigma(i)} \ , \ z_i = z'_{\sigma(i)} \ , \ \sigma(p_i) = p'_{\sigma(i)}$$

The equivalence relation $s \sim s'$ is defined by $\exists \sigma \in \mathfrak{S}_m, \ s \sim^\sigma s'$.

**Lemma 2**

In a symmetric instance instance $\mathcal{M} = (S, \Sigma, \Delta, \Omega, \mathfrak{f})$ the following holds: Let $s$ be a global state, $e \in E$ an event and $i \in I$ a process then

$$\forall s_1, s_2 \in \Delta(s, e, i), s_1 \sim s_2.$$

*Proof:*
For all $s, s' \in \Delta(s, e, i)$, let $F \ = \{j \in I, \ z_j = z_i\}$. Let $F_1, F_2 \subset F$ such that $s_1 = \Delta^*(s, e, i, F_1)$ and $s_2 = \Delta^*(s, e, i, F_2)$ let $\sigma \in \mathfrak{S}_m$ such that $\sigma(i) = i$, $\sigma(F_1) = F_2$ and $\sigma_{|I\setminus F}$ is the identity. Then $s_1 \sim^\sigma s_2$ holds.

□

**Lemma 3**

A symmetric instance $\mathcal{M} = (S, \Sigma, \Delta, \Omega, \mathfrak{f})$ satisfies the following: Let $\sigma \in \mathfrak{S}_m$ and $s = ((l_i, z_i)_{i\in I}, \{I_1, \ldots, I_u\})$, $s' = \big((l'_i, z'_i)_{i\in I}, \{I'_1, \ldots, I'_{u'}\}\big)$ be two global states such that $s \sim^\sigma s'$. Then:

- $(v_1(s), \ldots, v_n(s)) = (v_1(s'), \ldots, v_n(s'))$
- For all $i \in I$,
$$\Omega(s, e, i) = \Omega(s', e, \sigma(i)).$$

*Proof:*
The first claim is obtained from the definition of the variables. By hypothesis there is only one process type in $\mathcal{M}$ ; therefore $\Omega(s,e,i) = \omega(l_i,e)(v_1(s),\ldots,v_n(s))$ and $\Omega(s',e,\sigma(i))) = \omega_1(l'_{\sigma(i)},e)(v_1(s'),\ldots,v_n(s'))$. Thus as $l_i = l'_{\sigma(i)}$, $\Omega(s,e,i) = \Omega(s',e,\sigma(i))$.

□

**Proposition 12**

Let $\mathcal{M} = (S,\Sigma,\Delta,\Omega,\mathfrak{f})$ be a symmetric instance then it is strongly lumpable w.r.t. the relation $\sim$.

*Proof:*
Let $s = \big((l_j,z_j,p_j)_{j\in I}\big) = ((l_i,z_i)_{i\in I},\{I_1,\ldots,I_u\})$
and $s' = \big((l'_j,z'_j,p'_j)_{j\in I}\big) = \big((l'_i,z'_i)_{i\in I},\{I'_1,\ldots,I'_{u'}\}\big)$, such that $s \sim^\sigma s'$.
Let $i \in I$ be a process and $e \in \Sigma$ an event, $F \in \mathcal{P}(I)$ a subset of process and let $i'$ be equal to $\sigma(i)$, let $F' = \sigma(F)$.
Let $s^+ = \Delta^*(s,(e,i,F)) = \Big((l_j^+,z_j^+,p_j^+)_{j\in I}\Big)$
and $s'^+ = \Delta^*(s',(e,i',F')) = \Big((l_j'^+,z_j'^+,p_j'^+)_{j\in I}\Big)$ .
We have to prove that $\sum_{(e',i',F')|\Delta^*(s,(e',i',F'))\sim s^+} \Omega^*(s,(e',i',F')) = \sum_{(e',i',F')|\Delta^*(s',(e',i',F'))\sim s^+} \Omega^*(s',(e',i',F'))$. We first prove that $s^+ \sim s'^+$
For the sake of simplicity we will assume that $e$ labels a forward move. The proof is symmetric when $e$ labels a backward move.

- If the transition preserves the block structure of the global state $s$ then we have $s^+ \sim s'^+$ using the permutation $\sigma$.
- If the block containing $i$ is disjoint by the transition, $l_i^+ = t_{z_i+1}$ and $\forall g \in p_i\backslash\{i\}, l_g = t_{z_i+1}$. Block $p_i$ is disjoint to produce $a_{z_i}$ blocks of size 1, namely $\{g\}$, with $l_g^+ = t_{z_i+1}, \forall g \in p_i$.

  Same conditions are fulfilled by the state $s'$ Therefore $s^+ \sim^\sigma s'^+$.
- If the block containing $l_i$ is a singleton ($p_i = \{l_i\}$), there are at least $a_{z_i+1}$ other blocks of size one at the same position, and the number of blocks in the target zone $z_i^+$ is strictly smaller than $b_{z_i+1}$, then the transition joint $a_{z_i+1}$ blocks including $p_i$ in one block. Let $q$ be the union of these $a_{z_i+1}$ blocks of size 1. Partition of blocks in state $s^+$ is equal to: $\{p_j, p_j \not\subset q\} \cup \{p_{j_1} \cup \cdots \cup p_{j_{a_{z_i+1}}}\}$ where the new block takes position $t+1$. As $s \sim^\sigma s'$ there exists $a_{z_i+1}$ in $s'$ which are singletons in position $l_i$. The number of blocks in the target zone in $s'$ is equal to the one in $s$ thus the maximal number of allowed blocks in $s'$ is

not reached. The set $q' = p'_{\sigma(j_1)} \cup \cdots \cup p'_{\sigma(j_{a_{z_i+1}})}$ thus the partition of blocks in $s'^{+}$ is equal to $\{p'_j, p'_j \not\subset q'\} \cup \{p'_{\sigma(j_1)} \cup \cdots \cup p'_{\sigma(j_{a_{z_i+1}})}\}$

Finally $s^+ \sim^\sigma s'^{+}$.

We now compare the two sets $\{(e', i', F') | \Delta^*(s, (e', i', F')) \sim s^+\}$ and $\{(e', i', F') | \Delta^*(s', (e', i', F')) \sim s^+\}$.
Recalls that $s \sim^\sigma s'$, for all $(e', i', F') \in \Sigma, I, \mathcal{P}(I)$ such that $\Delta^*(s, (e', i', F')) \sim s^+$, it holds that $\Delta^*(s', (e', \sigma(i'), \sigma(F'))) \sim s^+$ using $s^+ \sim^\sigma s'^{+}$. Reciprocally for all $(e', i', F') \in \Sigma, I, \mathcal{P}(I)$ it holds that $\Delta^*(s', (e', i', F')) \sim s^+$ then $\Delta^*(s, (e', i', F')) \sim s^+$ and thus the two sets are equal. Using lemma 2 we have

$$\begin{aligned}
\sum_{\substack{(e',i',F')| \\ \Delta^*(s,(e',i',F'))\sim s^+}} \Omega^*(s, (e', i', F')) &= \sum_{\substack{(e',i'),\exists F| \\ \Delta^*(s,(e',i',F'))\sim s^+}} \Omega(s, (e', i')) \\
&= \sum_{\substack{(e',i'),\exists F| \\ \Delta^*(s,(e',i',F'))\sim s^+}} \Omega(s', (e', \sigma(i'))) \\
&= \sum_{\substack{(e',i',F')| \\ \Delta^*(s',(e',i',F'))\sim s^+}} \Omega^*(s', (e', i', F'))
\end{aligned}$$

□

To simplify the notation on symmetric instances, as $\sim$ is a lumping relation (Proposition 12) and as when a block is created all the choices of processes leads to equivalent state (Lemma 2), in the following, we use $\Delta$ in place of $\Delta^*$ and $\Omega$ in place of $\Omega^*$.

Every equivalence class of the relation $\sim$ contains states with the same stochastic behavior up to $\sim$. The following definition selects some representatives of the equivalence class which are easier to reason with.

Informally a state is *correctly ordered* if two conditions are fullfilled : processes in a same block are adjacently enumerated and the enumeration respects the partial ordering induced by the linear ordering of zones.

**Definition 38 (Correctly Ordered)**

We say that a state $s = (l_i, z_i, p_i)_{i \in I}$ is *correctly ordered* if

$$\forall i, j \in I,\ z_i > z_j \ \Rightarrow\ i < j$$

$$\forall i, j \in I,\ i \in p_j \ \Rightarrow\ \forall k \in \{i, \ldots, j\}, k \in p_j$$

**Lemma 4**

Let $s = (l_i, z_i, p_i)_{i \in I}$ be a global state, there exists a correctly ordered global state $\tilde{s} = (\tilde{l}_i, \tilde{z}_i, \tilde{p}_i)_{i \in I}$ such that $s \sim \tilde{s}$.

*Proof:*
We build a permutation iteratively. Let $I_\zeta = \{i, z_i = \zeta\}$. Let $c$ be the cardinality of $I_\zeta$. If $c \neq 0$, we build a permutation from $\{1, \dots c\}$ onto $I_\zeta$ such that $\forall i, j \in I_\zeta,\ i \in \tilde{p}_j \ \Rightarrow\ \forall k \in \{i, \dots, j\}, k \in \tilde{p}_j$.
Then we proceed the iteration from the $\zeta - 1$ zone.

□

## 5.4 Coupling

In this section we define a new relation $\leq_C$ on states of a symmetric instance. Informally, given two states, this relation defines whether one of them is "closer" to the final state $s_f$ than the other. We formalize this idea by showing that $\leq_C$ is a coupling relation and that $s_f$ is the maximal element for this relation. We use this relation only on symmetric instances, therefore for the sake of simplicity, we restrict the definition to this context in this whole section.

We need an additionnal constraint on the pattern to build the coupling relation:

**Definition 39 (Symmetric Monotonic Pattern)**

A pattern $\mathcal{A} = (Tp, \Sigma, L, Z, V, \delta, \{\omega\})$ is *symmetric monotonic* if $|Tp| = 1$ and for all $e \in \Sigma$, for all couple of states $l, l' \in L$ with $l \leq l'$, the following conditions are fulfilled:

(C1). $\delta(l, e) \leq \delta(l', e)$.

(C2). $\begin{cases} \delta(l, e) > l' & \Rightarrow \quad \omega(l, e) \leq \omega(l', e) \\ \delta(l', e) < l & \Rightarrow \quad \omega(l, e) \geq \omega(l', e) \end{cases}$

(C3). For all pairs of instantiation of variables $(v_i)_{i=1}^n, (v'_i)_{i=1}^n$ such that

$$\forall k \in \{1, \dots, n\},\ v_k \leq v'_k$$

the following condition holds:
$\begin{cases} e \in \Sigma_f \Rightarrow \omega(l, e)(v_1, \dots, v_n) \leq \omega(l, e)(v'_1, \dots, v'_n) \\ e \in \Sigma_b \Rightarrow \omega(l, e)(v_1, \dots, v_n) \geq \omega(l', e)(v'_1, \dots, v'_n) \end{cases}$

Condition (C1) on a monotonic pattern ensures that each transition with the same label, either forward or backward, preserves the order of states.

The pattern depicted in figure 5.1 satisfies this constraint. Condition (C2) ensures that if a process in state $l$ can overtake a process in state $l' \geq l$ by a forward transition $e$ then the rate of $e$ in $l$ is smaller than its rate in $l'$ (and similarly for a backward transition). Condition (C3) ensures that for two states containing a process in the same location, the rate of a forward transition is bigger for the state closer to the final state (and similarly for a backward transition).

Checking conditions C1,C2 and C3 can be done by enumeration. These conditions concern the pattern whose number of locations is small compared to the size of the state space of the instance. Each one of these conditions compares behaviors of two locations. As long as the rate expressions are simple enough, say piece-wise linear, all the checking can be done automatically.

**Remark 17** *An instance built from a symmetric monotonic pattern is indeed symmetric because all processes are of type* 1*.*

**Example 9** *In the database example, if all the databases are of type* 1*, the pattern is symmetric monotonic. Conditions C1, C2 and C3 hold.*

**Definition 40 (Coupling relation)**

Let $\mathcal{M} = (S, \Sigma, \Delta, \Omega, \mathfrak{f})$ be an instance, $\sigma$ be in $\mathfrak{S}_m$ and $s = (l_i, z_i, p_i), s' = (l'_i, z'_i, p'_i)$ be two states. The relation $s \leq_C^\sigma s'$ is defined by:

$$\forall i \in I, \quad \begin{array}{rl} & z_i < z'_{\sigma(i)} \\ \text{or} & z_i = z'_{\sigma(i)} \ \wedge \ \sigma(p_i) = p'_{\sigma(i)} \ \wedge \ l_i \leq l'_{\sigma(i)} \end{array}$$

$s \leq_C s'$ if there exists $\sigma \in \mathfrak{S}_m$ such that $s \leq_C^\sigma s'$.

For two states $s, s'$, $s \leq_C s'$ means that $s'$ is closer to the final state than $s$ in the sense that there is a one to one mapping which ensures that each process of $s$ is mapped to a process of $s'$ closer to $n$. It also requires that if a process in $s$ is mapped to a process in the same zone, its whole block is mapped to a block in $s'$.

The two next lemmas are technical properties between $\sim$ and $\leq_C$. They are required to prove that the relation we have built is indeed a coupling relation.

The next lemma shows that the relation $\sim$ and $\leq_C$ are "compatible". More formally, if $s \leq_C s'$ any state in the equivalence class of $s$ is in relation with any state of the equivalence class of $s'$.

**Lemma 5**

$$\forall s, s', \tilde{s}, \tilde{s}' \in S;\ (s \leq_C s') \wedge (s \sim \tilde{s}) \wedge (s' \sim \tilde{s}') \Rightarrow (\tilde{s} \leq_C \tilde{s}')$$

*Proof:*
Let $s = (l_i, z_i, p_i)_{i \in I}$ ; $s' = (l'_i, z'_i, p'_i)_{i \in I}$ ; $\tilde{s} = (\tilde{l}_i, \tilde{z}_i, \tilde{p}_i)_{i \in I}$ and $\tilde{s}' = (\tilde{l}'_i, \tilde{z}'_i, \tilde{p}'_i)_{i \in I}$.
Let $\sigma_1, \sigma_2$ and $\sigma_3$ be permutations such that $\tilde{s} \sim^{\sigma_1} s$, $s \leq_C^{\sigma_2} s'$ and $s' \sim^{\sigma_3} \tilde{s}'$.
Let $\sigma$ be equal to $\sigma_3 \circ \sigma_2 \circ \sigma_1$. Let us prove that $\tilde{s} \leq_C^{\sigma} \tilde{s}'$. For all $i \in I$, let $i_1 = \sigma_1(i)$; $i_2 = \sigma_2(i_1)$ and $i_3 = \sigma_3(i_2)$. If $z_{i_1} < z'_{i_2}$, as $\tilde{z}_i = z_{i_1}$ and $\tilde{z}'_{i_3} = z'_{i_2}$ we have $\tilde{z}_i < \tilde{z}'_{\sigma(i)}$. Otherwise:
$z_{i_1} = z'_{i_2}$ and due to the first equivalence we have: $\sigma_1(\tilde{p}_i) = p_{i_1}$ and $\tilde{l}_i = l_{i_1}$.
Due to the relation $\leq_C$ we have: $\sigma_2(p_{i_1}) = p'_{i_2}$ and $l_{i_1} \leq l'_{i_2}$.
Due to the second equivalence we have: $\sigma_3(p'_{i_2}) = \tilde{p}'_{i_3}$ and $l'_{i_2} = \tilde{l}'_{i_3}$.
By combining all these hypothesis we have:

$$\tilde{z}_i = z_{i_1} = z'_{i_2} = \tilde{z}'_{i_3} = \tilde{z}'_{\sigma(i)}$$

$$\sigma(\tilde{p}_i) = \sigma_3(\sigma_2(p_{i_1})) = \sigma_3(p'_{i_2}) = \tilde{p}'_{i_3} = \tilde{p}'_{\sigma(i)}$$

$$\tilde{l}_i = l_{i_1} \leq l'_{i_2} = \tilde{l}'_{i_3} = \tilde{l}'_{\sigma(i)}$$

□

In the following we pick for any pair of states $(s, s')$ such that $s \leq_C s'$ a fixed permutation $\sigma$ such that $s \leq_C^{\sigma} s'$. This ensures the unicity of the product CTMC defined below. Due to the previous lemma product CTMC built with different sets of permutations are equivalent by the relation $\sim$ and thus their respective lumpings coincide.

**Lemma 6**

$$\forall s, s' \in S;\ s \leq_C s' \Rightarrow \exists \tilde{s}, \tilde{s}' \in S \text{ correcly ordered s.t.}$$
$$s \sim \tilde{s} \text{ and } s' \sim \tilde{s}' \text{ and } \tilde{s} \leq_C^{Id} \tilde{s}'$$

*Proof:*
We assume that $s$ et $s'$ are correctly ordered. Let $s = (l_i, z_i, p_i)_{i \in I}$ and $s' = (l'_i, z'_i, p'_i)_{i \in I}$. Let $\sigma$ such that $s \leq_C^{\sigma} s'$.

Let $z$ be the rightest zone reached by $s$ and $L = \{i, l_i \in z\}$ ; we denote by $q$ its cardinal, which is a multiple of $a_z$.

Let $L'$ be the set of processes in $s'$ in zones on the right of $z$, that is $L' = \{i, z'_i \geq z\}$. Let us split $L'$ in $L'_> = \{i, z'_i > z\}$, $L'_= = \{i, z'_i = z\}$. Thanks to lemma 1, $|L'_>|$, $|L'_=|$ are multiples of $a_z$. As $s \leq^{\sigma}_C s'$, $L'$ contains $\sigma(L)$, thus we can also split $L'$ as the union of $\sigma(L)$ and $M = \{i, i \notin L; z'_{\sigma(i)} \geqslant z\}$. Remark that $|M|$, is also a multiple of $a_z$.

The permutation $\sigma$ induces a bijection from $M \cup L$ onto $L'$, and thus a bijection from $I \setminus (M \cup L) = \{i; z'_{\sigma(i)} < z\}$ onto $I \setminus L' = \{i; z'_i < z\}$.

We are going to construct three permutations $\mu, \mu'$ and $\tau$ such that : let $\tilde{s} = (\tilde{l}_i, \tilde{z}_i, \tilde{p}_i)_{i \in I}$ and $\tilde{s}' = (\tilde{l}'_i, \tilde{z}'_i, \tilde{p}'_i)_{i \in I}$ such that $\tilde{s} \sim^{\mu} s$, $\tilde{s}' \sim^{\mu'} s'$ $\tilde{s}$ and $\tilde{s}'$ are correctly ordered and $\tilde{s} \leq^{\tau}_C \tilde{s}'$, with $\forall i \in L, \tau(i) = i$. We then conclude recursively on the number of zones containing processes of $s$.

Let first suppose that $|L| \leq |L'_>|$. Then we define ; $\mu = \mu' = Id$, $\tau(i) = i$, for $i \in L \cup M$ (in that case, we do not change states $s$ and $s'$ but just re-affect the correspondance of processes between local states). The property $\tilde{l}_i < \tilde{l'}_i = l'_i$ results from $z_i = z < z'_i$, for $i \in L \cup M$.

Suppose now that $|L| > |L'_>|$ ; some of the blocks of $s$ in $L$ correspond via $\sigma$ to some blocks of $s'$ in $L$. Let $q$ be such that $qa_z = |L| - |L'_>|$. There are $q$ blocks $p_i$ in $L$ such that $p'_{\sigma(i)}$ is also in $L$, say $E$ the set of these blocks.

Remark that blocks of $s$ in $L$ are listed from 1 to $L$, block by block : $\{l_1, .., l_{a_z}\}$ that we may call the first block, $\{l_{a_z+1}, .., l_{2a_z}\}$ that we may call the second block, and so on until $\{l_{(r-1)a_z+1}, .., l_{ra_z}\}$ the last one, with $|L| = ra_z$. We replace $s$ by the equivalent state $\tilde{s}$ such that the $q$ blocks of $s$ in $E$ defined just above, are at the end of the list of indices in $L$. This is done using a permutation $\mu$ that fixes all indices in $I \setminus L$, thus $\tilde{s}$ is correctly ordered. We also change the numerotation of the processes of $s'$ which are in $z$ in the same way, just permuting the blocks, using the previous numerotation. This defines a permutation $\mu'$ that only moves the indices in $L$ such that $\sigma(i) \in L$ and replace $s'$ by the equivalent state $\tilde{s}' \sim^{\mu'} s'$. Precisely :

- for $i$ from 1 to $|L| - qa_z$, the property $\tilde{l}_i < \tilde{l'}_i = l'_i$ results from $z_i = z < z'_i$.

- for $i$ from $|L| - qa_z + 1$ to $|L|$, the property $\tilde{l}_i < \tilde{l'}_i$ results from the choice of the numerotation that ensures $\tilde{p}_i = \tilde{p}'_i$.

□

We are now in position to define the coupling Markov chain of a symmetric instance with itself:

**Definition 41 (Product CTMC)**

Let $\mathcal{M} = (S, \Sigma, \Delta, \Omega, \mathfrak{f})$ be a symmetric instance

We define the product CTMC of $\mathcal{M}$, as the chain $\mathcal{C}^{\otimes} = (\leq_C, \Sigma^{\otimes}, \Delta^{\otimes}, \Omega^{\otimes})$ where:

- $\Sigma^{\otimes} = \Sigma \uplus \tilde{\Sigma}$, where $\tilde{\Sigma}$ is a copy of $\Sigma$.
- $\forall (s, s') \in S^2$, $s \leq_C^{\sigma} s'$, $\forall i \in I$, $\forall e \in \Sigma$
  $\Delta^{\otimes}((s, s'), e, i) = (\Delta(s, e, i), \Delta(s', e, \sigma(i)))$
  and $\Delta^{\otimes}((s, s'), \tilde{e}, i) = \begin{cases} (\Delta(s, e, i), s') & \text{if } \Delta(s, e, i) \leq_C s' \\ (s, \Delta(s', e, \sigma(i)) & \text{if } s \leq_C \Delta(s', e, \sigma(i)) \\ (s, s') & \text{otherwise} \end{cases}$
- $\forall s \leq_C^{\sigma} s'$, $\Omega^{\otimes}((s, s'), e, i) = \min(\Omega(s, e, i), \Omega(s', e, \sigma(i)))$
  and $\Omega^{\otimes}((s, s'), \tilde{e}, i) = |\Omega(s, e, i) - \Omega(s', e, \sigma(i))|$

In order to make this definition consistent, we have to prove that the effect of a transition preserves the coupling relation. This is the purpose of the next lemma.

**Lemma 7**

Let $\mathcal{M} = (S, \Sigma, \Delta, \Omega, \mathfrak{f})$ be a symmetric instance. Then its product CTMC is preserved by transitions, that is:

$$\forall s \leq_C^{\sigma} s',\ \forall e \in \Sigma,\ \forall i \in I,\ \Delta(s, e, i) \leq_C \Delta(s', e, \sigma(i))$$

*Proof:*
Let $s = (l_i, z_i, p_i)_{i \in I}$ and $s' = (l'_i, z'_i, p'_i)_{i \in I}$ be two states in the relation $s \leq_C^{\sigma} s'$. By lemma 6 we may suppose that $s$ and $s'$ are correctly ordered and $\sigma = Id$. Let $e \in \Sigma$ and let $h$ be the process in $I$ which is moved by transition $e$. We denote by $s^+ = (l_j^+, z_j^+, p_j^+)_{j \in I}$ (respectively $s'^+ = (l_j'^+, z_j'^+, p_j'^+)_{j \in I}$) the successor of $s$ (respectively $s'$) by transition $(e, h)$: $s^+ = \Delta(s, e, h)$ (respectively $s'^+ = \Delta(s', e, h)$). There are several excluding cases:

- When $z_h^+ < z'^+_h$ the property trivially holds.
- When $z_h = z_h^+ = z'_h = z'^+_h$, that is, there is no change of zone neither for $s$ nor for $s'$ then condition C1 ensures that the property holds.
- When $z_h < z_h^+ = z'^+_h$, $z_h^+$ is odd and $e$ is a forward move, then $z_h^+ = z_h + 1$ and the block $p_h$ splits and reaches the boundaries of $z_h$. Then $\forall i \in p_h \backslash \{h\}, l_i = l_i^+ = t_{z_h/2+1} \leq l'^+_i$, using C1, $l_h^+ = t_{z_h/2+1} \leq l'^+_h$, thus the property holds.

- When $z'_h > z'^+_h = z^+_h$, $z'^+_h$ is odd and $e$ is a backward move, this case is symmetrical to the previous one by exchanging $s$ and $s'$.

- When $z_h < z^+_h = z'^+_h$, $z^+_h$ is even and $e$ is a forward move then $z^+_h = z_h + 1$, processes in $z_h$ joint together to build a new block $p^+_h$. There is at least $a_{z_h+1}$ processes in zone $z_h$ in $s$. We can order the processes in $s$ such that $p_h$ contains successive processes in $z_h$: Let $\sigma \in \mathfrak{S}_m$ such that $\tilde{s} \sim^\sigma s$ and $\forall i \in I,\ z_i \neq z_h \Rightarrow \sigma(i) = i$, and there exists $H = \{h_1, h_2, \ldots, h_{a_{z_h+1}}\}$, $\sigma(H) = p_h$, $\sigma(h) = h$ and $h_1$ is a multiple of $a_{z_h+1}$, this is possible because $s$ is correctly ordered. Let us consider the set of processes in $p^+_h$, there are two cases:

  ▷ If $z'_h = z_h + 1$ then $\forall i \in H, z'_i = z_h + 1$ due to the multiplicity of zones and because $s'$ is correctly ordered. Then by using C1, $l^+_h \leq l'^+_h$ and $\forall i \in H \backslash \{h\},\ l^+_i = t_{\frac{z_h+1}{2}} \leq l'^+_i$ and thus $s^+ \leq_C^{\sigma^{-1}} s'^+$.

  ▷ If $z'_h = z_h$ then $\forall i \in H, z'_i = z_h$ due to the multiplicity of zones and because $s'$ is correctly ordered. Then processes join to $l'_h$ to build a new block in $s'^+$. Let $\sigma' \in \mathfrak{S}_m$ such that $\sigma'(H) = p'^+_h$ then $s^+ \leq_C^{\sigma' \circ \sigma^{-1}} s'^+$.

- When $z'_h > z'^+_h = z^+_h$, $z'^+_h$ is even and $e$ is a backward move, this case is symmetrical to the previous one by exchanging $s$ and $s'$.

□

**Lemma 8**

Let $s$ be a global state and $e$ the label of a forward transition. For $i \in I$, we have $s \leq_C^\sigma \Delta(s, e, i)$ with $\sigma$ the identity.

*Proof:*
Let $s = ((l_j, z_j, p_j)_{j \in I})$
and let $s^+ = \Delta(s, e, i) = \Big((l^+_j, z^+_j, p^+_j)_{j \in I}\Big)$. Then $l^+_i = \Delta(l_i, e) \geq l_i$ and for all $j \neq i$, $l^+_j = l_j$ thus for all $j \in I$, $l'_j \geq l_j$ and $s \leq_C^{Id} s^+$.

□

**Lemma 9**

Let $s$ be a global state and $e$ the label of a backward transition. For $i \in I$, we have $\Delta(s, e, i) \leq_C^\sigma s$ with $\sigma$ the identity.

*Proof:*
Similar to the one of the previous lemma.

□

**Lemma 10**

Let $\mathcal{M} = (S, \Sigma, \Delta, \Omega, \mathfrak{f})$ be an instance built from a symmetric monotonic pattern. Then for all $e \in \Sigma$, $i \in I$ and $s \leq_C^\sigma s'$:

$$\begin{cases} \text{if } \Omega(s,e,i) \leq \Omega(s',e,\sigma(i)) & \text{then } s \leq_C \Delta(s',e,\sigma(i)) \\ \text{if } \Omega(s,e,i) \geq \Omega(s',e,\sigma(i)) & \text{then } \Delta(s,e,i) \leq_C s' \end{cases}$$

*Proof:*
First we assume that $\Omega(s,e,i) \leq \Omega(s',e,\sigma(i))$. We want to prove that $s \leq_C \Delta(s',e,\sigma(i))$. As the pattern is symmetric we have $\Omega(s,e,i) = \omega_1(l,e)$ and $\Omega(s',e,\sigma(i)) = \omega_1(l',e)$ thus $\omega_1(l,e) \leq \omega_1(l',e)$. As $s \leq_C^\sigma s'$ we also have $l \leq l'$.
Using the monotonicity of the pattern we know that $l \leq \delta(l',e)$.
There are now several cases depending of $e$'s transition type.

- $e$ labels a forward move then $s \leq_C s' \leq_C \Delta(s',e,\sigma(i))$ using lemma 9.
- $e$ labels a backward move which does not change the block structure, then as $l_i \leq \delta(l'_{\sigma(i)}, e)$ and the others processes do not move we have $s \leq_C^\sigma \Delta(s',e,\sigma(i))$.
- $e$ labels a backward move disjoining a block, then as $\forall j \neq i,\ l_j \leq l'_{\sigma(j)}$ and $l_i \leq \delta(l'_{\sigma(i)}, e)$, processes coupled with the block disjointed by the transition i.e. $\{j \in I | \sigma(j) \in p'_{\sigma(i)}\}$ are in a zone $z_i$ such that $z_i < z'_{\sigma(i)}$, thus $s \leq_C^\sigma \Delta(s',e,\sigma(i))$.
- $e$ labels a backward move joining a blocks. Let $s'^+ = (l'^+_j, z'^+_j, p'^+_j)_{j\in I}$ be equal to $\Delta(s',e,\sigma(i))$. Then we have: $\forall j \neq i,\ l_j \leq l'_{\sigma(j)}$ and $l_i \leq \delta(l'_{\sigma(i)}, e)$,then processes coupled with blocks joint by the transition i.e. $\{j \in I | \sigma(j) \in p'^+_{\sigma(i)}\}$ are in a zone $z_i$ such that $z_i < z'_{\sigma(i)}$, thus $s \leq_C^\sigma \Delta(s',e,\sigma(i))$.

Similarly if we assume $\Omega(s,e,i) \leq \Omega(s',e,\sigma(i))$ a symmetric proof allows to have $\Delta(s,e,i) \leq_C s'$ using lemma 8.

□

Using Proposition 4 the product CTMC of $\mathcal{M}$ is a coupling.

## 5.5 Application to Guaranteed Variance Reduction

This framework can be used to prove guaranteed variance reduction in two ways:

1. For a system which is not symmetric one can define its symmetrized counterpart and define the reduced model as the lumping of the symmetric model. In the following a method to compute the symmetrized version of a system is presented yielding a fully automatic method.

2. When the reduced model can be defined as a symmetric monotonic instance this framework ensures that $\leq_C$ is a coupling relation. Then using theorem 10 the guaranteed variance reduction can be proved. One still have to define a reduction function $f$ and the labels function $g$.

We now describe the symmetrization of an instance.

**Definition 42 (Symmetrization)**

Let $\mathcal{M} = (S, \Sigma, \Delta, \Omega, \mathfrak{f})$ be an instance. We define the symmetrization of $\mathcal{M}$ denoted by $Sym(\mathcal{M}) = (S, \Sigma, \Delta, Sym(\Omega), \mathfrak{f})$ by:
$\forall s \in S, e \in \Sigma, i \in I$,

$$Sym(\Omega)(s, e, i) = \begin{cases} \max_{s' \sim s} \Omega(s', e, i) & \text{if e is a forward move} \\ \min_{s' \sim s} \Omega(s', e, i) & \text{if e is a backward move} \end{cases}$$

The block assigning function is unchanged. By construction the resulting instance is symmetric.

**Example 10** *For the database example the symmetrization is obtaied by replacing databases of type* 2 *by databases of type* 1*. Table 5.3 reports the size of the CTMC obtained from the initial model with N databases and the size of the CTMC obtain from the symmetric model. A drastic reduction of the size of the CTMC is observed.*

| N | Initial model | Symmetric model |
|---|---|---|
| 20 | 8,821 | 690 |
| 40 | 107,141 | 2,580 |
| 60 | 492,961 | 5,670 |
| 80 | 1,484,281 | 9,960 |
| 100 | 3,519,101 | 15,450 |
| 120 | 7,155,421 | 22,140 |
| 140 | 13,071,241 | 30,030 |
| 160 | 22,064,561 | 39,120 |

Table 5.3: Number of states in the database example before and after symmetrization and lumping.

To build a reduced model from a pattern which is not monotonous we need to replace the pattern by a bounding pattern which is monotonous. This bounding pattern must fulfill some condition to ensure that we obtain a reduced model with guaranteed variance. We define an order relation over the set of patterns as follows:

**Definition 43 (Order on pattern)**

Let $\mathcal{A} = (Tp, \Sigma, L, Z, V, \delta, \{\omega\})$ and $\mathcal{A}' = (\{1\}, \Sigma, L, Z, V, \delta', \{\omega'\})$ two pattern with the same set of locations, events and zones.
$\mathcal{A}'$ is *bounding* $\mathcal{A}$ if $\mathcal{A}'$ is symmetric monotonic, and for all $k \in Tp$, $l_1, l_2 \in L$ the following holds:

$$\forall e \in \Sigma_f,\ \delta'(l_1, e) \leq \delta(l_1, e) \wedge \omega'_k(l_1, e) \leq \omega(l_1, e)$$

$$\forall e \in \Sigma_b,\ \delta'(l_1, e) \geq \delta(l_1, e) \wedge \omega'_k(l_1, e) \geq \omega(l_1, e)$$

Several algorithms computing bounding models for CTMC exist [2, 33, 41]. they can be applied to patterns to obtain a symmetric monotonic pattern bounding the initial pattern. The bounding pattern is suitable to build a reduced model by lumping its equivalent states..

Remark that the symmetrization of an instance $\mathcal{M}$ of a monotonous pattern is a bounding $\mathcal{M}$.

The following proposition proves that a bounding model could be used as a reduced model in the method described in Chapter 4.

**Proposition 13**

Let $\mathcal{M}$ and $\mathcal{M}'$ be two instances such that the pattern of $\mathcal{M}'$ is bounding the pattern of $\mathcal{M}$ and $\mathcal{M}'$ is symmetric monotonic then $\mathcal{M}'$ is a reduction with guaranteed variance of $\mathcal{M}$.

*Proof:*
We have to prove that the hypothesis of Proposition 10 holds. Let $\mathcal{M} = (S, \Sigma, \Delta, \Omega, \mathfrak{f})$ and $\mathcal{M}' = (S, \Sigma, \Delta', \Omega', \mathfrak{f})$.

Recall that the transition function and rate function that we use for the initial and reduced model must be $\Delta^*$ and $\Omega^*$ when applying the proposition.

The reduction function $f$ is defined as the identity. Recall that the set of event $E$ in the initial model is equal to $\Sigma \times I \times \mathcal{P}(I)$. The family of functions $g : S \times E \to E$ is defined as the identity on the labels: $g(s, e) = e$.

For all $s \in S$, $e \in Sigma$, $i \in I$ and $F \in \mathcal{P}(I)$, there are two cases to prove Condition 1:

- If $e \in \Sigma_f$,

$$\begin{aligned}
\Delta^{'*}(f(s), g(s,(e,i,F))) &= \Delta^{'*}(s,(e,i,F))) \\
&= \Delta^{'}(s,(e,i))\mathfrak{f}(s,F) \\
&= \omega'(l_i,e)\mathfrak{f}(s,F) \\
&\geq \omega_{\kappa(i)}(l_i,e)\mathfrak{f}(s,F) \\
&= \Delta(s,(e,i))\mathfrak{f}(s,F) \\
&= \Delta^{*}(s,(e,i,F))
\end{aligned}$$

  Thus Condition (1.b) holds. Due to Lemma 8, $s \leq_C \Delta(s,(e,i))$ and thus Condition (1.a) holds.

- If $e \in \Sigma_b$ the proof is symmetric.

Condition 2 holds due to:

$$\begin{aligned}
f(\Delta^*(s,(e,i,F))) &= \Delta^*(s,(e,i,F)) \\
&= \Delta^*(f(s),(e,i,F)) \\
&= \Delta^*(f(s),g(s,(e,i,F)))
\end{aligned}$$

□

**Example 11** *Proposition 13 applies to the database example. Thus, a reduced model with guaranteed variance can be built. Using 200 databases, 100 of each type, the size of the* CTMC *is* $53,075,700$*. The size of the reduced model however is* $60,900$ *which can be efficiently handled by numerical methods. We compute the probability that a whole transaction takes place in* 3 *time units, that is the state where all processes are in location* 7 *is reached within* 3 *time units. The result of the computation using the importance sampling methods described in Chapter 4 and the tool* Cosmos *described in Chapter 6 are that this probability is in the interval* $[3.09\ 10^{-38}, 1.17\ 10^{-37}]$ *with a confidence level of* $0.99$*. With* $300,000$ *trajectories the computation takes 2 hours and uses 4GB of memory.*

## 5.6 Conclusion

In this chapter we have developed a framework for which a bounding model can be built automatically. Moreover this bounding model is suitable to be used as a reduced model with guaranteed variance. The framework structurally sets for such a model the monotonicity of the bounding model. In a future work, we would like to model more complex systems. In particular,

in this framework constraints are given to control the interaction between several processes moving on a same pattern to ensure monotonicity, it would be interesting to weaken these constraints. For example it should be possible to let processes fork in a constant number of processes when entering a zone and join back into a single process when leaving the zone. An other future work is to extend this framework to open systems where processes can be created or destroyed.

The method described in this chapter could be automatized but to make it completely automatic, it remains to build a tool that implements all steps of this method. There is no theoretical difficulty to build such a tool but they could be technical ones as the condition on rate covers expression containing variables and not only reals.

# Part III

# Applications

# Chapter 6

# Cosmos

## 6.1 Introduction

In Chapter 4, a rare event acceleration method was proposed. We chose to implement it in the tool Cosmos which is a statistical model checker of stochastic Petri nets with general distributions against the hybrid automata stochastic logic (HASL) and has been initially developed by Paolo Ballarini and Hilal Djafri [9, 8, 27]. The implementation of rare event handling required to modify how probability distributions were sampled. In order to evaluate these modified probability distributions, facilities to compute numerically steady states and transient distributions were also implemented such that in total, the rare event handling required around 3600 new lines of C++ code. We also provided two main improvements for Cosmos:

- We extended the input formalism to Symmetric Stochastic Nets (SSN). This is a join work with Elvio Amparore, Marco Beccuti, Susanna Donatelli and Giuliana Franceschinis. This extension required heavy modifications of the simulator (around 2500 lines of code) to implement efficient simulation of SSN. We also extended the specification language of Cosmos to take advantage of the SSN formalism [I].
- We integrated Cosmos in the CosyVerif [II] platform (around 1800 lines of parser in Cosmos and around 1500 lines of Java and XML in CosyVerif). CosyVerif provides a uniform way to deal with models, as well as tools to edit and analyze them. Before this integration, CosyVerif dealt only with Petri nets, symmetric nets and hierarchical Petri nets. Thus support for stochastic models and automata had to be added. Contributions to the CosyVerif platform are join work with Étienne André, Clément Démoulins, Lom Messan Hillah, Francis Hulin-Hubard, Fabrice Kordon, Alban Linard and Laure Petrucci.

When one wants to analyze a particular model, one needs to choose the formalism, the method and the tool for the modeling. It is thus important

that the performances of the different methods and tools are benchmarked against one another in order to evaluate tool adequacy and efficiency. In this chapter, we perform different experiments:

- runtime comparisons between Cosmos and several statistical model checkers: Plasma [53], Prism [63], Uppaal [17], and Ymer [87],
- several experimentations and benchmarks of our importance sampling method presented in Chapter 4 against other importance sampling methods and numerical methods,
- runtime benchmarks for the new SSN simulator of Cosmos compared to its plain Petri net simulator.

Section 6.2 describes the general architecture of Cosmos. Section 6.3 describes the implementation of the extension of Cosmos which deals with importance sampling. Section 6.4 describes the contribution to the CosyVerif platform and the integration of Cosmos. Section 6.5 describes the extension of Cosmos to SSN as well as experiments using this extension. Section 6.6 compares Cosmos with some other statistical model checkers. Section 6.7 presents the experiments conducted on importance sampling. Section 6.8 contains our conclusions.

## 6.2 Description of the Tool

### 6.2.1 Architecture

Cosmos is a statistical model checker accepting as input several types of stochastic Petri nets with general distribution and a HASL formula. A HASL formula is described by a Linear Hybrid Automaton (LHA). The main algorithm of this tool randomly simulates the Petri net according to its stochastic semantics and synchronizes with the execution of the automaton. During the synchronization, it evaluates HASL expressions. A statistical procedure decides when to stop the simulation and produces a confidence interval for a HASL expression.

The tool Cosmos consists of about 17000 lines of C++ code and is freely available at [10] under the Gnu General Public License version 3 (GPLv3). The tool relies on code generation to perform efficient simulation. It is divided into three main parts:

1. The parsing and code generation part reads the input files and the command line in order to build data structures for the net and the automaton. Then C++ code simulating both the behaviors of the net and the automaton is generated. The resulting code is compiled by a C++ compiler and linked with the simulator library. The resulting binary is the complete simulator program.

2. The simulator part is a library implementing the algorithm synchronizing the net and the automaton. It also implements the stochastic generation of event using the pseudo random number generator provided by the BOOST library and handle these events in an event queue. Rare event algorithms are implemented in the simulator and modify the way events are produced.

3. The server part launches several copies of the simulator and aggregates their results. According to statistical parameters, a procedure decides whether enough trajectories have been simulated and stops all simulators when needed. Then, HASL expressions are evaluated and several output files are produced according to options. The computation of confidence intervals uses the BOOST library for the computation of quantiles of the normal distribution function and binomial distribution.

Figure 6.1 shows a detailed overview of COSMOS internal architecture. This schematics details all files with which COSMOS interacts. Input files can be in different formats. File formats `.gspn` and `.lha` are the legacy input languages for COSMOS described in the Chapter 6 of [27]. The file format `.grml` is used by the COSYVERIF platform. More details on this format are provided in Section 6.4. The files `spn.cpp`, `markingImpl.cpp` and `LHA.cpp` contain the generated code implementing the simulator. The `markingImpl.cpp` file is used to implement colored tokens when simulating SSN. The file `Result.res` contains the result of the computation (a copy of this file is also displayed on the command line). Several other files can be generated using some options. For the rare event part, files are generated to store the vector of probabilities $\mu^\bullet$ and the transition probability matrix $\mathbf{P}^\bullet$. The file `modelUnfold.grml` contains an unfolded version of a model when the input is a SSN. Data files `Rawdata.dat` and `Result.dat` store data produced by the simulator. Depending on several options, they may contain the result of the simulation over time, the probability density function or the cumulative density function of an expression, the step-by-step trace of simulations or a sampled trace of simulations. Their format allows them to be read by GNUPLOTto produce a graph. When using importance sampling for rare events, the reduction function is implemented in file `lumpingfun.cpp`.

Several tools interact with COSMOS either by calling it or by being called by it:

- A C++ compiler is required to build the simulator from the generated code. Until now, GCC and Clang have been used.

- When the Petri net is Markovian and the automaton has no clock, PRISM can be called by the simulator. In this case a state space generator is used and the CTMC of the product between the model and the automaton is given to PRISM. The vector of probabilities computed

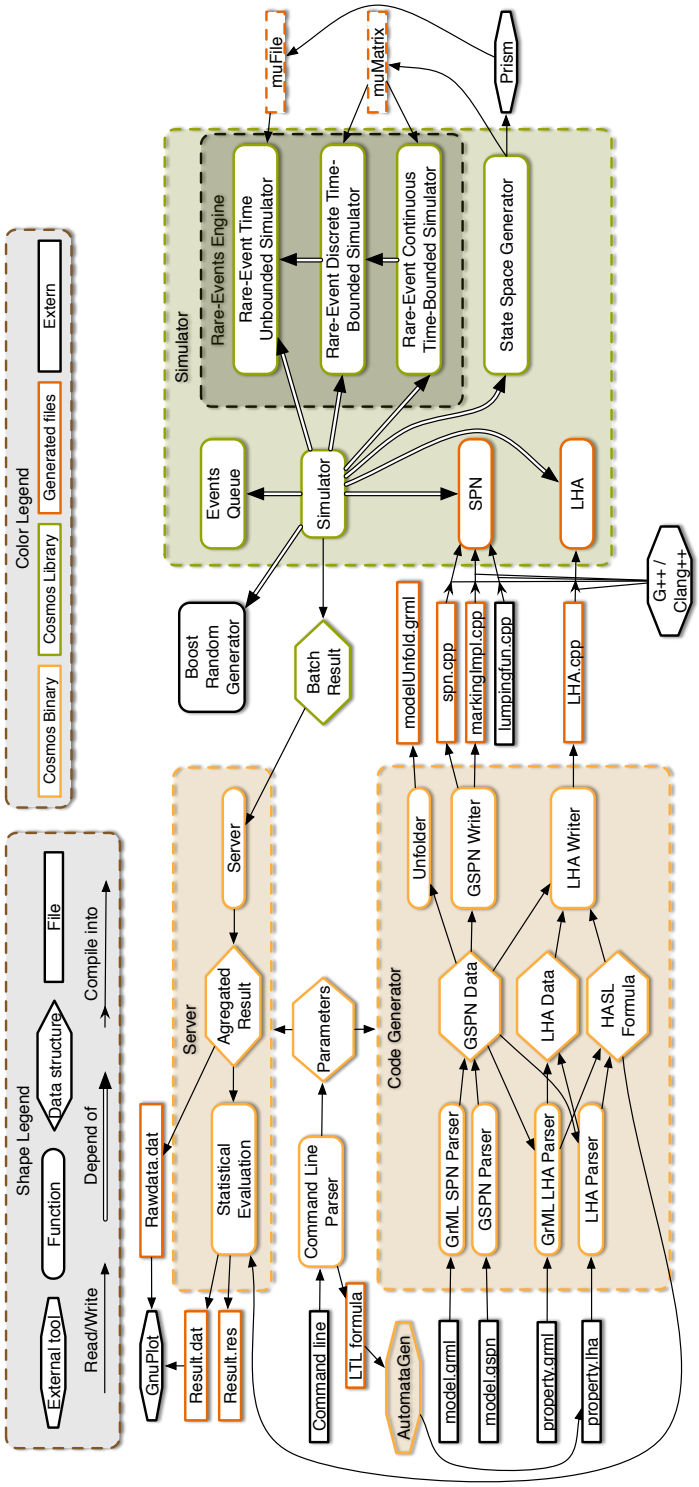



Figure 6.1: Overview of Cosmos architecture

by Prism on this model is parsed by Cosmos. This allows one to use Prism to compute transient and steady state probabilities.

- The plotter GnuPlot can be called by Cosmos to produce a graph from the various output files providing a better visualization of the results.
- A small OCaml tool called AutomataGen which is part of Cosmos builds a simple automaton from a LTL style formula. This provides an easier use of the tool for simple properties where the full expressive power of HASL is not required.
- Several OCaml scripts shipped with Cosmos perform benchmark, testing and plotting.

Moreover Cosmos has been integrated into the platform CosyVerif. Alligator, the server part of CosyVerif can call Cosmos and parse its result. Coloane, the client part of CosyVerif can call Cosmos through Alligator and displays the result. It can also produce `.grml` files for the Petri net or the automaton that can be used through a command line.

### 6.2.2 The HASL Logic

Cosmos uses the HASL logic as a specification language [9]. In this section, we describe this logic. A formula in this logic contains two parts, a Linear Hybrid Automaton (LHA) and a set of HASL expressions. The LHA is a deterministic automaton. It can be synchronized with a discrete event stochastic process (DESP), described as a Petri net, in order to select some of its trajectories. HASL expressions are formulas on the variables of the LHA, that are evaluated along each accepted trajectory of the system.

The synchronized product between the DESP and the automaton contains two kinds of transitions: either the transition is initiated by the DESP and the automaton performs a *synchronization* transition; or the transition is initiated by the automaton which performs an *autonomous* transition which does not affect the DESP. These two types of transitions are described in more detailed below. To ensure that the synchronization can efficiently be computed, some limitations on the synchronized product are required:

- The automaton is deterministic in the sense that for any trajectory of the DESP, there is at most one run of the automaton.
- For any finite trajectory of a DESP, the run of the automaton is also finite, that is, the automaton cannot take an infinity of autonomous transitions which are not synchronized with the DESP.
- In order to deal efficiently with the hybrid part of the automaton, variables evolve linearly with time in the implementation.

More formally, in order for an LHA to synchronize with a Petri net, several common objects are used in the definition of an LHA.
Given a SPN $\mathcal{M}$:

- The set of events $E$ is a subset of the set of transitions $T$ of $\mathcal{M}$.
- The set $\mathsf{Ind}$ is defined as the set of *indicator functions* which are real-valued functions whose values depend only on the marking of $\mathcal{M}$ in a given state.
- The set $\mathsf{Prop}$ is a set of boolean propositions where atoms are equalities or inequalities of indicator functions in $\mathsf{Ind}$.

The formal definition of a LHA is as follows:

**Definition 44 (Linear Hybrid Automaton (LHA))**

A Linear Hybrid Automaton is a tuple $\mathcal{A} = \big(E, L, \Lambda, L_0, L_f, X, \mathit{flow}, \rightarrow\big)$ where:

- $E$ is a finite alphabet of events;
- $L$ is a finite set of locations;
- $\Lambda : L \rightarrow \mathsf{Prop}$ is a location labeling function;
- $L_0$, the set of initial locations is a subset of $L$, ;
- $L_f$, the set of final locations is a subset of $L$;
- $X = (x_1, ...x_n)$ is a $n$-tuple of data variables;
- $\mathit{flow} : L \mapsto \mathsf{Ind}^n$ is a function which associates each location to one indicator per data variable representing the evolution rate of the variable in this location. $\mathit{flow}_i$ denotes the projection of $\mathit{flow}$ on its $i^{th}$ component.
- The transition relation $\rightarrow$ is a subset of $L \times \big((2^E \times \mathsf{Const}) \uplus (\{\sharp\} \times \mathsf{IConst})\big) \times \mathsf{Up} \times L$. This is a set of edges, where:
  - ▷ The letter $\sharp$ is a special event which labels autonomous transitions of the automaton.
  - ▷ $\mathsf{Const}$ denotes a set of *constraints* defined as boolean combinations of equalities or inequalities over $\mathsf{Ind}$ and $(x_i)_{i=1}^n$,
  - ▷ $\mathsf{IConst}$ denotes a set of *left-closed linear constraints* which are boolean combinations of expressions of the form

$$\sum_{i=1}^{n} \alpha_i x_i \bowtie c,$$

where $\alpha_i, c \in \mathsf{Ind}$ and $\bowtie \in \{\leq, \geq, =\}$.

▷ $\mathsf{Up} = \big(\mathsf{Up}_1, \ldots, \mathsf{Up}_n\big)$ denotes the sequence of updates. Each $\mathsf{Up}_i$ corresponds to the update of the variable $x_i$ and is an expression over $\mathsf{Ind}$ and $(x_i)_{i=1}^n$.

The notation $l \xrightarrow{E',\gamma,U} l'$ is defined as $(l, E', \gamma, U, l') \in \rightarrow$.

Furthermore the LHA $\mathcal{A}$ fulfills the following conditions.

- **Initial determinism:** $\forall l \neq l' \in \mathit{Init}$, $\Lambda(l) \wedge \Lambda(l') \Leftrightarrow \texttt{false}$. This must hold whatever the interpretation of the indicators occurring in $\Lambda(l)$ and $\Lambda(l')$ is.
- **Determinism on events:** $\forall E_1, E_2 \subseteq E \, s.t. \, E_1 \cap E_2 \neq \emptyset, \forall l, l', l'' \in L$, if $l'' \xrightarrow{E_1,\gamma,U} l$ and $l'' \xrightarrow{E_2,\gamma',U'} l'$ are two distinct transitions, then either $\Lambda(l) \wedge \Lambda(l') \Leftrightarrow \texttt{false}$ or $\gamma \wedge \gamma' \Leftrightarrow \texttt{false}$. Again this equivalence must hold whatever the interpretation of the indicators occurring in $\Lambda(l)$, $\Lambda(l')$, $\gamma$ and $\gamma'$ is.
- **Determinism on $\sharp$:** $\forall l, l', l'' \in L$, if $l'' \xrightarrow{\sharp,\gamma,U} l$ and $l'' \xrightarrow{\sharp,\gamma',U'} l'$ are two distinct transitions, then either $\Lambda(l) \wedge \Lambda(l') \Leftrightarrow \texttt{false}$ or $\gamma \wedge \gamma' \Leftrightarrow \texttt{false}$.
- **No $\sharp$-labeled loops:** For all sequences $l_0 \xrightarrow{E_0,\gamma_0,U_0} l_1 \xrightarrow{E_1,\gamma_1,U_1} \cdots \xrightarrow{E_{p-1},\gamma_{p-1},U_{p-1}} l_p$ such that $l_0 = l_p$, there exists $i \leq p$ such that $E_i \neq \sharp$.

Given a Petri net $\mathcal{M}$ and an LHA $\mathcal{A}$, a state of the synchronized product is a tuple $(m, l, X, t)$ where $m$ is a marking of $\mathcal{M}$, $l$ is a location in $L$ of the automaton $\mathcal{A}$, $X$ is the valuation of the data variable and $t$ is the current time. This synchronized product evolves with two different kinds of transitions. When a transition is fired in the Petri net, the automaton matches this transition with a transition of its own. The automaton performs autonomous transition as soon as some timed guards are satisfied. The Petri net has a fully-stochastic semantics and the automaton is deterministic yielding a fully-stochastic semantics for the synchronized product. More precisely the synchronized product evolves as follows:

- Due to its initial determinism, the LHA for the synchronized product has only one initial state: $(m_0, l_0, (0)_{i=1}^n, 0)$ where $m_0$ is the initial marking of the Petri net and $l_0 \in \mathit{Init}$ is the initial location of the automaton satisfying the initial determinism.
- The Petri net $\mathcal{M}$ fires a transition $e$ at time $t_1$ such that $m \xrightarrow{e,t_1} m'$. If there exists a synchronized transition in $\mathcal{A}$ such that $l \xrightarrow{e,\gamma,U} l'$ where $m', X + (t_1 - t) \cdot \mathit{flow}(l) \models \gamma$ and $m' \models \Lambda(l')$ and there is no valid $\sharp$

transition in the automaton that can occur before $e$, that is:

$$\forall t_2 \in [t, t_1],\ \forall l_2 \in L,\ \gamma \in \mathsf{IConst},\ U \in \mathsf{Up},$$
$$l \xrightarrow{\sharp,\gamma,U} l_2 \Rightarrow \big(m, X + (t_2 - t) \cdot \mathit{flow}(l) \not\models \gamma\big),$$

then a synchronized transition occurs leading to state $(m', l', \big(\mathsf{Up}_i[m', X + (t_1 - t) \cdot \mathit{flow}(l)]\big)_{i=1}^{n}, t_1)$ in the synchronized product.

- There is a valid autonomous transition labeled by $\sharp$ in the automaton, that is

$$\exists t' \geq t,\ l' \in L,\ \gamma \in \mathsf{IConst},\ U \in \mathsf{Up},$$
$$l \xrightarrow{\sharp,\gamma,U} l' \Rightarrow \big(m, X + (t' - t) \cdot \mathit{flow}(l) \models \gamma\big).$$

  If there is no scheduled transition in $\mathcal{M}$ before time $t'$ and there is no $\sharp$ transition which can occur before time $t'$, then a autonomous transition occurs leading to state $\big(m, l', \big(\mathsf{Up}_i[m', X + (t' - t) \cdot \mathit{flow}(l)]\big)_{i=1}^{n}, t_1\big)$.

- The synchronization fails when the automaton does not reach an accepting state and one of the following events occurs:
  - ▷ The system $\mathcal{M}$ is in deadlock and no $\sharp$ transition is available in the automaton.
  - ▷ The system takes a transition which cannot be matched with any transition of the automaton.

  In this case the trajectory is rejected.

- As soon as an accepting state of the automaton is reached, the trajectory is stopped and accepted.

**Definition 45 (HASL Expressions)**

HASL expressions are given by the term $Z$ in the following grammar:

$$\begin{aligned}
Z ::=&\ c \mid \mathrm{P} \mid \mathrm{E}[Y] \mid \mathrm{HYPOTHESIS}[c, c] \\
&\mid Z + Z \mid Z - Z \mid Z \times Z \mid Z/Z \\
Y ::=&\ c \mid \mathit{Last}(y) \mid \mathit{Min}(y) \mid \mathit{Max}(y) \mid \mathit{Int}(y) \mid \mathit{Mean}(y) \\
&\mid Y + Y \mid Y - Y \mid Y \times Y \mid Y/Y \\
y ::=&\ c \mid x \mid y + y \mid y \times y \mid y/y,
\end{aligned}$$

where $c$ denotes a real constant, $x$ denotes a variable of the automaton and the arithmetic operators have their usual meaning. Expressions derived from $y$ are computed at each simulation step of a trajectory. Expressions derived from $Y$ are computed at the end of an accepted trajectory, where:

- $Last(y)$ denotes the value of $y$ in the last state of the trajectory,
- $Min(y)$ and $Max(y)$ denote the minimal and maximal values taken by $y$ along a trajectory,
- $Int(y)$ is the integral over time of the value of $y$ along a trajectory,
- $Mean(y)$ is the average value taken by $y$ along a trajectory.

Expressions derived from $Z$ are HASL expressions and are evaluated on a set of trajectories where:

- P denotes the probability of acceptation of trajectories by the automaton.
- E$[Y]$ is the expected value of $Y$,
- HYPOTHESIS$[c_1, c_2]$ denotes a pair of hypotheses: whether the probability for a path to be accepted by the automaton is above or below $c_1$ with indifference zone of width $c_2$.

Expressions derived from $Y$ are efficiently computed using that the hybrid part of the automaton is piecewise linear. Expressions derived from $Z$ are evaluated as confidence intervals using statistical methods described in Section 6.2.3.

**Running example.** *Figure 6.2 depicts a HASL formula whose first expression $Expr_1$ characterizes the probability of global overflow in the tandem queues, that is there are more than $N$ clients in the two queues before reaching a state where the queues are empty. Furthermore $Expr_2$ estimates the mean time from the initial state to reach a state with more than $N$ clients without visiting a state with no client.*

*All the transitions of the automaton are labeled by $E, \top, \{\}$ meaning that they can synchronize with any transition of the Petri net, the guard is always true and there is no update. The determinism is ensured by the invariant of location.*

### 6.2.3 Statistical Procedures

We now detail the different statistical procedures proposed by COSMOS for evaluating HASL expression depending on several criteria. The statistical results on which they rely are recalled in Section 2.3.

- **Sequential hypothesis testing [85].** This procedure checks whether a probability is above a threshold. Parameters of this procedure are the

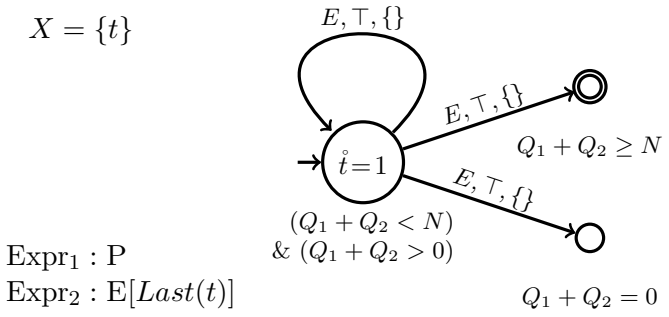



Figure 6.2: A HASL formula for the tandem global overflow property

probability of an error for a positive answer and a negative answer and the width of the indifference region. When the value of the probability is outside the indifference interval, the probability of an error is bounded by the parameter corresponding to the answer.

- **Chernoff-Hoeffding bounds [49].** This static method requires three related parameters, each of them can be determined by the two others. These parameters are the interval width, the confidence level and the number of samples. It outputs a confidence interval whose width satisfies the requirement and where the probabilistic guarantee is exact. It applies to estimate the expectation of a bounded random variable.

- **Chow-Robbins bounds [21].** This sequential method requires two parameters: the interval width and the confidence level. It outputs an interval whose width satisfies the requirement and where the probabilistic guarantee is asymptotic w.r.t. the width of the interval. It applies to estimate the expectation of a random variable, when no known bound is available.

- **Gaussian approximation.** This static method requires two parameters. The number of samples has to be given. The second parameter is either the confidence level or the interval width, one of these determining the other one. It ouputs an interval whose width satisfies the requirement and where the probabilistic guarantee is asymptotic w.r.t. the number of samples. It applies to estimate the expectation of a random variable. It is based on the central limit theorem.

- **Clopper-Pearson bound [23]** This static method computes confidence intervals for binomial distributions. It takes as input three parameters, the total number of samples, the confidence level and the number of successful samples and outputs a confidence interval for the probability of a sample to be successful.

## 6.3 Integration of Importance Sampling

Here the integration of algorithms presented in Chapter 4 is detailed.

### 6.3.1 Distribution Parameters

Cosmos is designed to simulate discrete event stochastic systems. To achieve this, the time of occurrence of each event is sampled according to its distribution. A queue of events stores the times at which each event will occur. The main loop of the simulation engine takes the first event in this queue, executes it and if needed adds or removes events from the queue. In this setting it is easy to simulate CTMCs. At each step of the CTMC simulation, a time is computed for each transition according to its rate and added to the queue.

Consequently Cosmos does not simulate directly a DTMC, but a CTMC whose rates are equal to the probability distribution of the DTMC. This approach has some impact on the implementation of the importance sampling algorithm.

In a general case we assume that vectors of probability $\mu^\bullet_u$, $u \in \mathbb{N} \cup \{\infty\}$ required to compute rate of the importance sampling are stored as an array. The reduction function $f$ is given by the user as a $C$ function. Algorithm 5 is used to compute each simulation step. Starting from a state $s \in S$ and a current likelihood $L \in \mathbb{R}^{+*}$, the algorithm computes the next step $t \in S$ and updates the likelihood. Function $\lambda$ is the rate function of the CTMC. Function $\lambda'$ is the rate function under the importance sampling, it is computed on-the-fly and is implemented as an array. The two float variables $Acc$ and $Acc'$ are used to store the sum of the rates. The data structure $EQ$ is the event queue, it is implemented as a heap to ensure that an insertion can be done in logarithmic time. The event $\perp$ is a special event representing all the events leading to the state $s_-$.

Algorithm 6 is the plain algorithm used in Cosmos when importance sampling is not required. In this algorithm the computation of the set $\{s' \in E \mid \lambda(t, e') \neq \lambda(s, e')\}$ is done efficiently using the structure of the model. As Cosmos takes as input a stochastic Petri net, the structure of the net is analyzed to precompute this set. This set is in general much smaller than the set $E$. The event queue must be initialized at the start of the trajectory.

When importance sampling is required, Cosmos uses Algorithm 5 which is more costly. In this algorithm, the event queue is emptied and filled again at each step of the simulation. In this case the function $\mu^\bullet$ is not local while without importance sampling the firing of a transition of the Petri net is local and only affects places to which it is connected.

Experiments simulating a fixed number of trajectories show that the ratio of speed between Algorithm 6 and Algorithm 5 under an importance sampling where $\mu^\bullet = 1$ is around 5.5 slower on the example of tandem queues.

**Algorithm 5:** Simulation Step Under Importance Sampling with Sink State

`SimulationStep`$(s, \mu_u^\bullet, \mu_{u+1}^\bullet, \lambda, L)$
**Data**: $\lambda', Acc, Acc', EQ$
**Result**: $t, L$

$EQ \leftarrow \emptyset$; $Acc \leftarrow 0$; $Acc' \leftarrow 0$
**for** $e \in \{s' \in E \mid \delta(s,e) \neq s_- \wedge \lambda(s,e) > 0\}$ **do**
    `// e loops over firable transitions not leading to` $s_-$`.`
    $\lambda'(s,e) \leftarrow \frac{\mu_{u+1}^\bullet(f(\delta(s,e)))}{\mu_u^\bullet(f(s))}$
    $Acc \mathrel{+}= \lambda(s,e)$
    $Acc' \mathrel{+}= \lambda'(s,e)$
    $EQ[e] \leftarrow SampleExponential(\lambda'(s,e))$
**if** $Acc' \leq Acc$ **then**
    `// Add a special transition with the remaining probabilities.`
    $\lambda'(s,\perp) \leftarrow Acc - Acc'$
    $Acc' \leftarrow Acc$
    $EQ[\perp] \leftarrow SampleExponential(\lambda'(s,\perp))$
$e \leftarrow argmin_{e \in E}(EQ[e])$
**if** $e = \perp$ **then**
    `// If` $e = \perp$ `the system is moved to state` $s_-$`.`
    `// The likelihood is set to 0.`
    $t \leftarrow s_-; L \leftarrow 0$
**else**
    $t \leftarrow \delta(s,e)$
    $L \mathrel{*}= \frac{\lambda(s,e)}{Acc} \frac{Acc'}{\lambda'(s,e)}$

**Algorithm 6:** Simulation Step

```
SimulationStep(s, EQ, λ)
Result: t, EQ
1  e ← argmin_{e∈E}(EQ[e])
2  EQ[e] ← ∞
3  t ← δ(s, e)
4  for e' ∈ {s' ∈ E | λ(t, e') ≠ λ(s, e')} ∪ {e} do
       // e' loops over transitions altered by the last
       transition.
5      if δ(t, e') > 0 then
6          EQ[e'] ← SampleExponential(λ(s, e'))
7      else
8          EQ[e'] ← ∞
           // If e' is no longer firable its firing time is set
           to ∞
```

This is acceptable as in rare event setting Algorithm 6 does not work at all.

### 6.3.2 State-Space Generation and Numerical Computation

In order to apply the methods described in Chapter 4, the tool has to generate the state space and the transition probability matrix of the reduced model and to perform numerical computation on it. As this part of the tool is only supposed to work with small reduced models, it is not as much optimized as the simulation part.

The workflow for using the importance sampling of Cosmos is in two parts:

1. Cosmos is run on the reduced model and if required with a "reduced" property. The tool generates the state space of the reduced model and performs the required numerical computation for the time-unbounded case. Results of this computation and the probability transition matrix are exported into a file.

2. Cosmos is run on the initial model. It compiles the reduction function with the model and imports the file containing the results of numerical analysis or the transition probability matrix to compute the importance sampling.

The state space generator is built on top of the simulator. From a state of the system, all possible transitions are fired and the state obtained after each firing is copied in a stack of states to be explored. Tools dedicated to state-space generation usually rely on methods based on decision diagrams and are much more efficient.

Transition probability matrices are stored as sparse matrices. Steady state analysis is done by exporting the transition probability matrix to Prism and retrieving the result. Prism state space generator is not used because Cosmos needs to identify states of the reduced model to apply the reduction function. Moreover it would require the reduced model to be given in the Prism language whereas the initial model is a Petri net. Transient analysis is performed by Cosmos using sparse matrix vector multiplication. This transient analysis is performed using algorithms described in Chapter 4.

### 6.3.3 Fox-Glynn Algorithm and Uniformization

The computation of Poisson probabilities is performed using the freely available implementation of Fox-Glynn algorithm described in [51]. The uniformization of the original Petri net model is done by adding a special transition which implements loops on the underlying CTMC. The loop rate is computed after each firing of a transition.

## 6.4 CosyVerif

### 6.4.1 Description

CosyVerif [II] is a software environment for the verification of systems. The goal is to provide an environment allowing to handle and modify models given in various formalisms and to interface many verification tools. This allows a user to build and modify models in an easy-to-use graphical interface and to then use various tools in the same interface. One may also edit a model on a personal computer while running verification tools on more powerful machines using a client-server architecture.

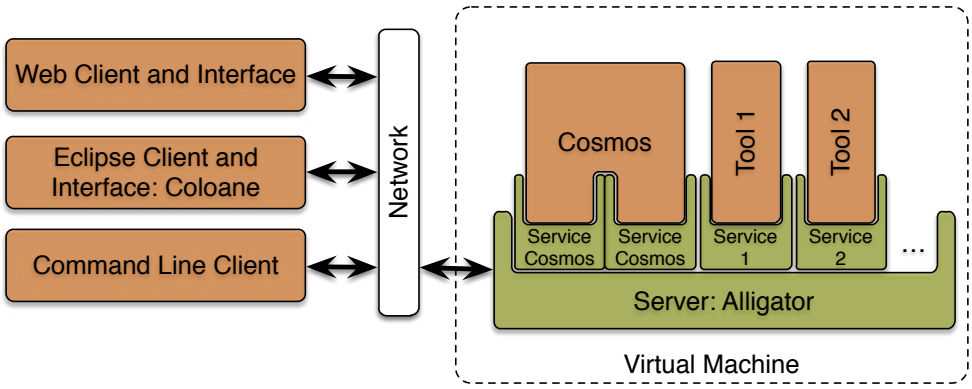



Figure 6.3: Scheme of CosyVerif architecture

To achieve these goals, CosyVerif relies on a common file format to

exchange any kind of graph-based model between its tools. This file format is structured using the notion of *formalism*. A *formalism* is a meta-model for a formal notation. Tool developers must then describe formalisms accepted by their tools. When a model is created, one chooses the formalism to which this model belongs. The platform CosyVerif matches then formalisms of the models and tools. It provides users with a list of available tools for a given model. To achieve this, a language for formalism has been developed and is called Formalism Markup Language (FML). It describes how a model is structured. All models described by a FML are graph based and are encoded in the Graph Markup Language (GrML). Both FML and GrML rely on XML as file format. Figure 6.4 illustrates the relation between FML, GrML formalisms and models.

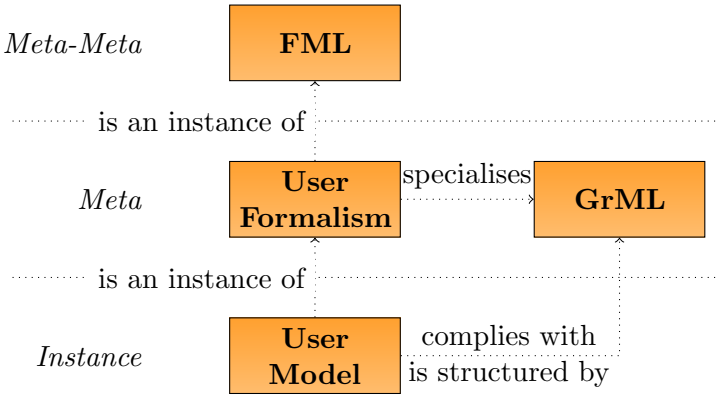



Figure 6.4: User formalism and model complying with FML and GrML

To handle various kinds of formalisms, CosyVerif uses a hierarchy of formalisms sharing common concepts. One part of this hierarchy concerns automata, the other one Petri nets. In each of these parts, variants of formalisms are defined incrementally, favoring the reuse of portions of formalisms. Figure 6.5 shows the hierarchy as it is implemented in CosyVerif.

### 6.4.2 Personal Contributions

CosyVerif is the successful result of a collaboration of experts of different formalisms. It consists of around 20K lines of code for the server and 60K lines of code for the client. It is mainly written in Java and XML. In this section, I describe my personal contributions to this platform.

- CosyVerif has been initially built for handling Petri nets and symmetric nets. In order to integrate Cosmos in the platform, several new formalisms had to be built in and lead to the current hierarchy of formalisms of CosyVerif 6.5:

- ▷ A formalism for handling general arithmetic and boolean expressions used in all formalisms of the platform. It also handles parameters in expressions in a uniform way.
- ▷ Stochastic extensions of Petri nets defining stochastic nets with general distributions from Petri nets and symmetric stochastic nets from symmetric nets.
- ▷ Several variants of automata, and in particular linear hybrid automata which are used by Cosmos.

Abstract formalisms are used in this hierarchy to denote partial implementation of formalisms. For example, `abstractPN-Core` defines a notion of Petri nets as a bi-parted graph without marking or valuation. This formalism is extended later in `P/T Net` when markings and valuations are integers and in `Symmetric-Net` when markings and valuations consist of colored tokens.

- The graphical editor of CosyVerif allows one to design models of each formalism of the hierarchy. This required to implement parser handling of all expressions appearing in the various formalisms and producing an AST which is encoded in GrML.

  Figure 6.6 shows a screen-shot of the graphical editor Coloane displaying a trace of a Cosmos simulation through the CosyVerif platform.

- All tools integrated to the CosyVerif platform are launched through small programs called *services* which specify the list of parameters required by a tool and call the tool with those parameters. Tools allowing different kinds of computation which require different inputs are integrated with several services, which is the case of Cosmos.

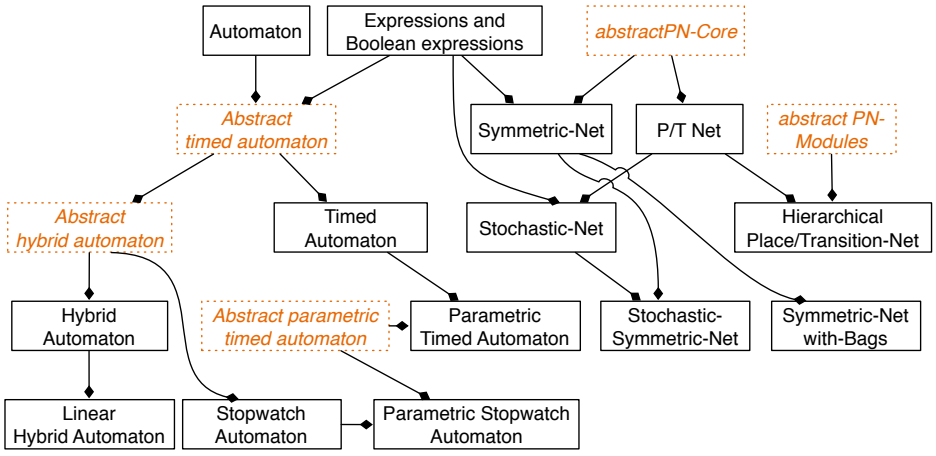



Figure 6.5: Hierarchy of formalism in CosyVerif

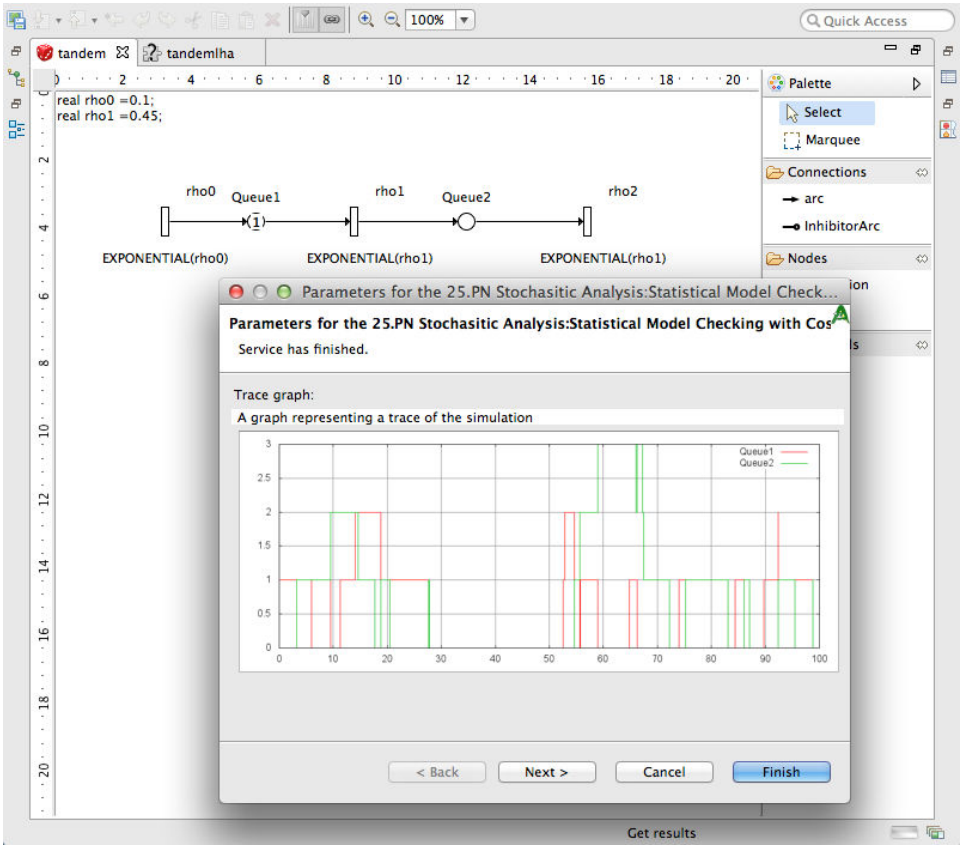



Figure 6.6: Screen-shoot of Coloane calling the tool Cosmos

## 6.5 Stochastic Symmetric Net for Cosmos

Stochastic Symmetric Nets (SSN) are a well-known extension of Petri nets with an exponentially more compact representation. They can also be used to specify parametrized Petri nets in a compact way. In this section, I present contributions related to the implementation of the model-checking of SSNs in Cosmos. This work has been published in [I]. To take full advantage of the SSN model, the specification logic must be extended in order to specify properties on colored tokens. There are two possible approaches for simulating SSNs:

- *Direct simulation* consists in using a simulator which takes colors into account and which simulates directly the semantic of SSNs. As rules for firing transitions become more complex, the simulation time can be increased by this approach.

- *Reduction to stochastic Petri nets* consists in *unfolding* a SSN into a stochastic Petri net with the same semantics and then using a

stochastic Petri net simulator. This approach allows one to use all the optimizations of the Petri net simulators. However, the size of the unfolded Petri net may become huge which can be problematic on some simulators. This approach also requires to "unfold" the specification formula to a formula about stochastic Petri nets.

The two approaches have been implemented in Cosmos and this section provides experimental results which compare them.

### 6.5.1 Extension of HASL

In this section, we present an extension of the HASL logic to SSNs. As SSNs use colored tokens, the HASL logic had to be extended to be able to deal with them.

The HASL logic was extended in several ways, mainly by extending the definition of LHA:

1. Automata variables can now have three different types:

   - *flow variables* take values in $\mathbb{R}$ and correspond to the original variables of HASL. They evolve continuously with a constant rate with respect to the time between two transitions.
   - *stable variables* take values in $\mathbb{R}$ but are only altered by transitions updates. On a single trajectory they take only a countable number of values. As they do not vary between transition firings, they can be used as constants in linear expressions.
   - *color variables* take colors as value and are modified only on transition updates. A color variable can be seen as a subtype of a stable variable.

   Additionally, these three type of variables can be indexed by a color or an integer. In this case they behave like an array.

2. An event read by the LHA is now a pair of a transition identifier and a binding of the SSN color variables. This allows the LHA to synchronize over specific colors of the net. The value of binding can be compared to or stored into color variables.

3. An expression involving some marking of the net specifies a filter on the colors of tokens which can be color variables.

4. Linear expressions of variables are linear combinations of flow variables where coefficients are expressions that can contain stable variables and marking expressions.

The extension of HASL allows one to specify interesting complex properties like property D1 given in Figure 6.7. This property applies to the database model presented in Figure 2.12 on page 41. It computes the CDF of the time for an update to be performed when the system is in a nominal state: the number of concurrent updates is small and the communication speed between the sites is high.

More precisely, the property D1 describes the *cumulative distribution function of the time required for a change request to complete, under the condition that the time to acquire the mutex on the file to be changed does not exceed a given threshold and the time to send all messages from the active site which modified the file to the passive sites is below a given threshold.* The LHA expressing formally D1 starts by waiting in location $l_0$ until $T$ time units elapse, then waits in location $l_1$ for the transition `Start` to occur. The binding of the color variables of the SSN is copied in the LHA color variable. Using the color variable, the automaton waits for the transition `Acquire` in $l_2$ with the same binding as for the `Start` transition. This waiting time is bounded by the constant threshold. The automaton continues to synchronize with the SSN until the transition `Release` occurs and the final state $l_6$ is reached. All trajectories in which a time bound is exceeded are rejected. Other examples of properties using the extension of HASL on different models are given in [I].

Property D1 has been evaluated with Cosmos providing the graph depicted in Figure 6.8.

### 6.5.2 Implementation of SSN in Cosmos

The implementation of SSN in Cosmos has been influenced by the specification language of symmetric nets in the CosyVerif platform. In particular, binding variables are defined globally to the model and not locally for each transitions.

Let us first describe two possible data structures to store colored tokens:

- Each place marking can be implemented as a dynamic structure containing a list of colored tokens with their number of occurrences. This data structure is well suited for places such that the number of tokens for any marking is small but the cardinality of the domain is big. The memory consumption is linear in the number of tokens. Within this approach, the time required for checking the presence of a token with a particular color is linear whereas emptiness of a place can be checked in constant time.

- Alternatively, each place marking can be implemented as an array of integers where each possible colored token is assigned to a cell of the array. This data structure comes with an overhead in the memory consumption when the number of colors is too big. The time required

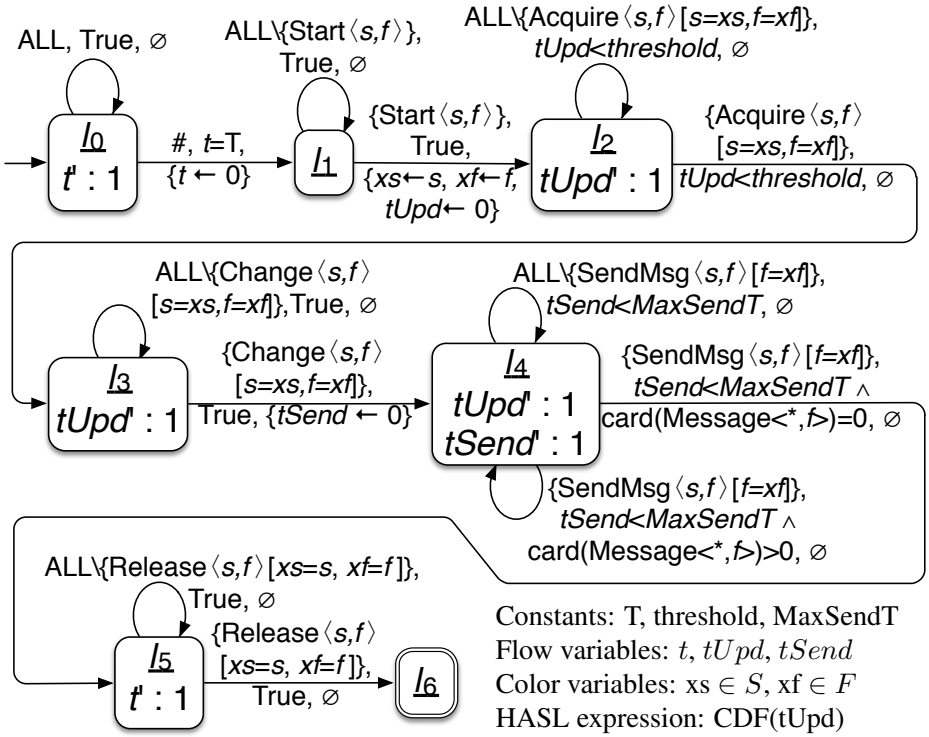



Figure 6.7: HASL formula for the property D1

to check the presence of tokens of a particular color is constant whereas checking for emptiness required a time proportional to the size of the domain of the place.

The second approach is implemented in COSMOS as it allows faster simulation when the number of colors is not excessively large, which is usually the case. This also extends naturally the original implementation of COSMOS for the marking of uncolored tokens and fits well within the code generation algorithm of the tool. Specific classes are generated to handle each color domain and color class. These classes handle color tokens of each domain.

An optimization could be to implement both approaches, and choose for each place how to implement the marking, based on an analysis of the net providing the size of the domain and a possible bound on the maximal number of tokens.

In the SSN implementation of the simulator, events are additionally labeled by the bindings of the color variables of the SSN. Code is generated to enumerate fireable bindings for a given pair of a colored marking and a transition.

The implementation of the extension of HASL in COSMOS is quite straight-

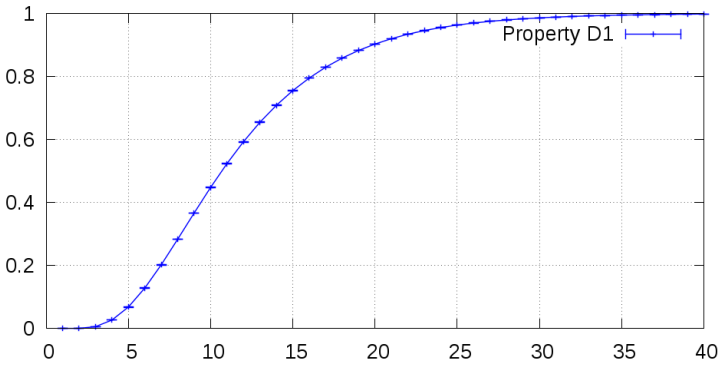



Figure 6.8: Result of the estimation of D1 on the distributed database example

forward, the automaton is synchronized with information on the binding of variables instead of only synchronizing over transitions.

### 6.5.3 Runtime Comparison of Symmetric Net Simulator vs Petri net Simulator

Given a SSN, a Stochastic Petri net can be built such that the set of reachable markings of both nets are in a one-to-one mapping. This transformation known as *unfolding* is implemented in Cosmos. In this section, the time and memory consumptions of Cosmos on a model and on its unfolded counterpart are discussed. The comparison of some performance evaluation measures also asserts the correctness of the SSN simulator.

The number of places and transitions is usually much bigger in the unfolded Petri net than in the symmetric one. For instance on the database example, the place *wait_mutex* whose domain is $S \times F$ needs to be replaced by $|S| \cdot |F|$ places in the unfolded model. Similarly, each transition is replaced by a transition for each possible binding.

As Cosmos relies on code generation to simulate efficiently Petri nets, some code is generated for each place and transition of the net. Thus the time and memory required for the building phase before the simulation depends on the size of the net. When the number of places and transitions is small, this building time is negligible. However when this number becomes bigger than 1000, it may become the bottleneck of the computation.

Figure 6.9 represents a client-server system where each query of a client requires three independent computations. This model uses colored tokens to keep track of the different queries. This client-server system is used with the database example to benchmark the symmetric net simulator.

Figure 6.10 shows the comparison of the time consumption of both the SSN and SPN simulators on the client server example (on the right) and the

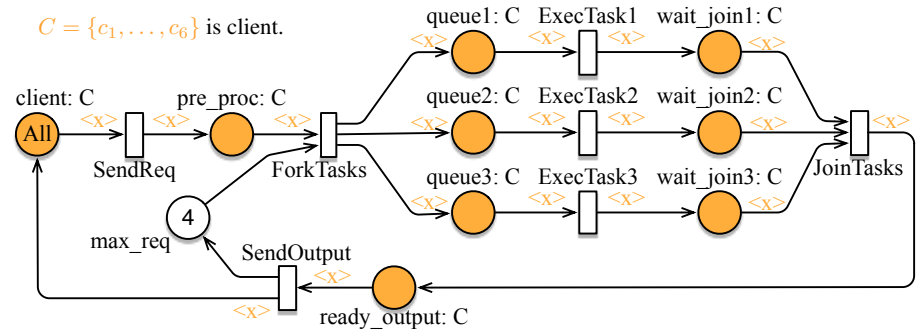



Figure 6.9: SSN representing a client server system

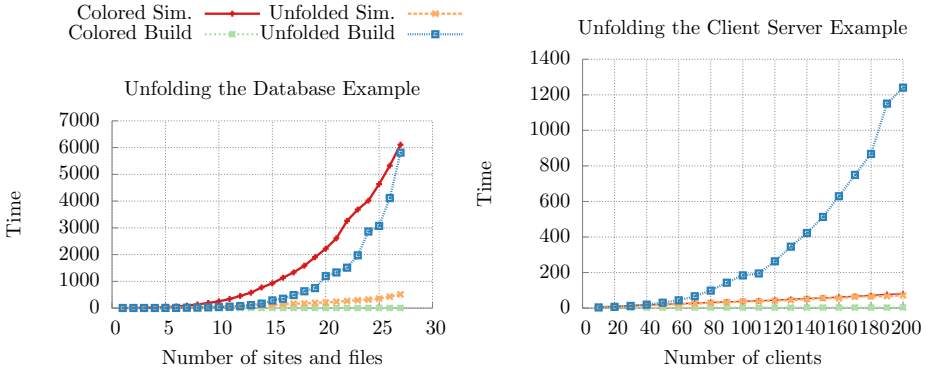



Figure 6.10: Time comparison of SSN and SPT simulators

database example (on the left) as a function of the size of the color class. The time for building the simulator is also reported. The memory consumption in the building phase is proportional to the time consumption and thus is omitted in the graph for the sake of clarity. The memory consumption in the simulation phase is negligible and thus is also omitted. In both examples, the mean number of tokens of a specific color in a specific place has been evaluated (Property C2 for the client-server). The results of all simulations are consistent, with intersecting confidence intervals on SSN and SPN which confirms the correctness of the SSN simulator implementation.

The two examples show different behaviors. The client-server example features simple firing rules for the transitions. In this case, the expressions on the arcs are simple combinations of the binding variables. Therefore, these firing rules are simulated efficiently and the simulation time in the SSN is only slightly bigger than in the SPN. At the same time, as the SPN size increases drastically with the number of clients, the building time for the SPN simulator becomes intractable. The construction time on the SSN is always negligible. Thus on this example, the use of SSN speeds up the

overall computation.

On the database example, the firing rules are more complicated. In particular, arcs labeled by $< All - s, f >$ are naively handled by COSMOS, yielding a large overhead in the simulation time. Even as the building time increases for SPN, for properties which require a large number of trajectories, the additional time required to used SSN is not negligible.

## 6.6 Comparison Between Statistical Model Checkers

In this section, we compare COSMOS to other statistical model checkers with respect to different criteria:

- Focusing on the inputs of the tools, we compare the expressiveness and the conciseness of the formalisms for modeling a system.
- Similarly, we discuss the expressiveness and the conciseness of the language for the specification of a logical property.
- As these tools generate a large number of samples, we compare the time to generate a (random) trajectory in Section 6.6.2.
- We present the statistical procedures included in each of these tools and their impact on the whole computation time.

The following comparison is related to the one of [50] where the authors performed experimental comparisons of numerical and statistical model checkers. Recently, new tools performing SMC have been released justifying new comparisons.

### 6.6.1 Expressiveness Comparison

Numerous tools are available for performing SMC, some of them also performing numerical model checkers. Here is a non-exhaustive list of tools freely available for universities: COSMOS, PLASMA [53], PRISM [63], UPPAAL [17], APMC [48], YMER [87], MRMC [58] and VESTA [81]. We have not found any updates of APMC since 2006. Since 2011, MRMC has not been updated and the corresponding team seems to use UPPAAL. Finally, the link for downloading VESTA is not valid anymore. So we focus on the following tools: YMER, PRISM, UPPAAL, PLASMA and, COSMOS.

#### Ymer

YMER is a statistical model checker taking as input CTMCs or generalized semi-Markov processes, described in the PRISM language. Its specification language is a fragment of CSL (an adaptation of CTL with probabilistic

operators replacing path operators and adding bound to time operators) without the steady-state operator but including the unbounded until.

### Prism

Prism performs numerical and statistical model checking on probabilistic models. The numerical part of Prism is dedicated to discrete and continuous Markov chains, Markov decision processes and probabilistic timed automata. The statistical part deals with discrete and continuous Markov chains. The Prism language defines a probabilistic system as a synchronized product between modules with finite state space and guarded transitions. Using this representation, this language describes big systems in a compact way. The verification procedures of Prism take as input a wide variety of languages for the specification of properties. Most of them are based on CSL or PCTL (a formalism close to CSL).

### Uppaal

Uppaal is a verification tool including many formalisms: timed automata, hybrid automata, priced timed automata, etc. It supports automata-based and game-based verification techniques. Large scale applications have been analyzed with Uppaal. It has recently been enriched with a statistical model checker engine. The corresponding formalism is based on probabilistic timed systems. The probabilistic extension of Uppaal defines distributions as follows: exponential for transitions having guards without upper time bound, and uniform otherwise. The specification language is (P)LTL (*i.e.* an adaptation of LTL with path operators substituted for quantifiers) with bounded until.

### Plasma

Plasma is a platform dedicated for statistical model checking, designed as flexible. The plugin system allows a developer to add extensions and can also be integrated in another software in a library. Prism and languages dedicated to the modeling of biological systems are supported. Prism language is extended with more general distributions. The specification language is a restricted version of PLTL with a single threshold operator.

### Discussion

Formalisms are characterized by different features. First, they can be programming languages oriented like Prism or formal model oriented like Cosmos. In general, formalisms take advantage of the concurrency present in the model. Uppaal combines both approaches and allows one to specify

timing requirements in the system. Application-based languages are also proposed (*e.g.* for biological systems in the case of PLASMA).

Property specifications are either defined by some timed probabilistic temporal logic or by combining hybrid automata with appropriate expressions. Whatever the choice, the main distinctive features are the following ones: presence of the unbounded until, nesting of probabilistic operators and expressiveness of time requirements. Beyond boolean properties and probability computations, COSMOS provides an expressive way to specify performance indices.

### 6.6.2 Runtime Comparison of Statistical Model Checkers

In this section, we perform runtime comparisons of several statistical model checkers. We choose examples taken from classical probabilistic system benchmarks. These examples are easily modeled in the different formalisms of tools.

#### Tandem queues

The first example is a variation of the tandem queue example used as the running example and is given in Figure 6.11. The Tandem Queues System (TQS) is a $M/Cox_2/1$ queue composed with a $M/M/1$ queue, it is available on the PRISM web page [63] (where the Cox distribution is detailed). Please note that this example is different from the tandem example used in other sections of this thesis. Here, the two queues are bounded by a constant $N$. The specification property, denoted $\phi_{TQS}$, is: *The two queues are full before $T$ time units*, where $T$ is some time bound. This property is expressible in all specification languages of the tools. One may be interested either in the satisfaction probability or only in knowing whether it is above 0.3. For the experiments, we use $N = 5$, $\lambda = 20$, $\mu_1 = 0.2$, $\mu_1' = 1.8$, $\mu_2 = 2$ and $\kappa = 4$ as in [63].

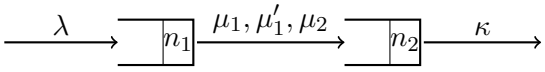



Figure 6.11: Tandem Queues System (TQS)

#### Dining Philosophers

The Dining Philosopher Model (DPM) is a standard benchmark problem for discrete event systems. We have adapted it in a probabilistic setting. In the DPM, $N$ philosophers eat and think around a table. While thinking, a philosopher decides after some random time to pick its right fork, then after some more time, he decides to take its left fork and to start to eat. Later on,

the philosopher starts thinking again and puts back his two forks. A fork is shared between two neighbours at the table. In the probabilistic setting, all the times are exponentially distributed. The property that we study is the following one: *a deadlock occurs before they have been N lunches (possibly by the same philosopher)*. The single deadlock occurs when all philosophers have taken their right fork. The number of states in this system is exponential with respect to the number of philosophers. This benchmark is thus a typical case were SMC is the only way to perform analysis of the system due to the state space explosion problem. For the experiments, we choose 10 for the (common) rate of the exponential distributions.

**Experiment settings**

We have set the following statistical parameters: the confidence level is 0.95 and the width of confidence interval is 0.005, the probability of error is 0.005 for the hypothesis testing and the width of the indifference region is 0.001. These parameters have been chosen in order to obtain a important number of trajectories. The other parameters have been set to their default value. Most tools can take advantage of parallelisation but for simplicity we use only one processor[1] for the comparison.

PRISM, UPPAAL and PLASMA support Chernoff-Hoeffding and hypothesis testing methods. In addition, PRISM provides two sequential methods for confidence intervals where the stopping criterion is unknown to us. COSMOS provides Chernoff-Hoeffding, Chow-Robbins and Gaussian methods. YMER offers the hypothesis testing method.

**DPM experiments.**

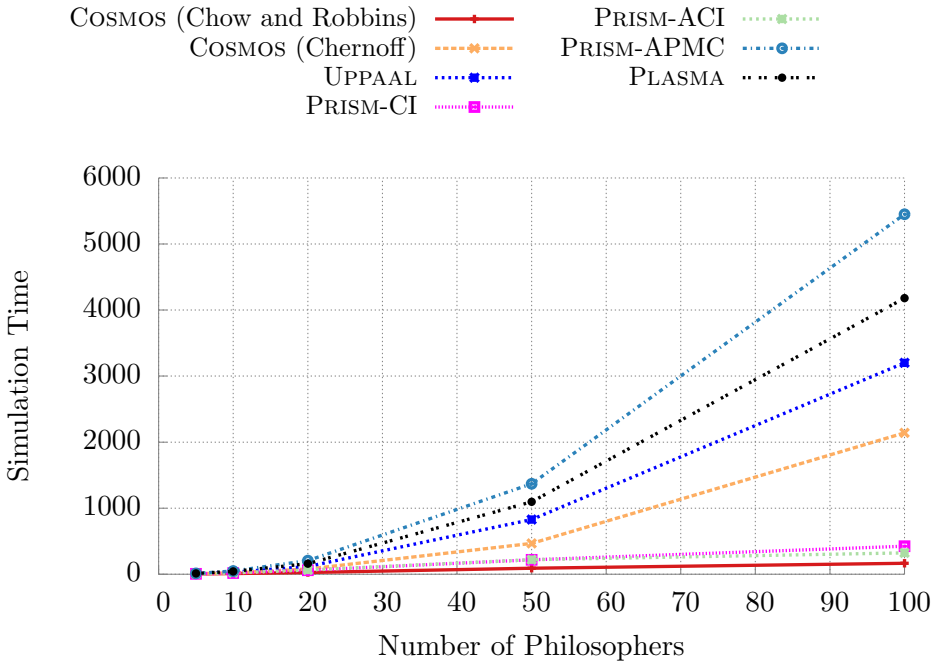
Figure 6.12 refers to the runtime for the DPM as a function of the number of philosophers. The results are also reported in Table 6.1. Among the tools using Chernoff-Hoeffding bounds, COSMOS is the fastest. For 100 philosophers, PLASMA is 1.95 times slower, UPPAAL is 1.5 times slower and PRISM-APMC is 2.5 times slower. Among the tools using sequential procedure, the two versions of PRISM have similar runtime and COSMOS is up to 1.95 times faster.

**TQS experiments with confidence intervals.**

Results about the runtime comparison with different time bounds $T$ are reported in Figure 6.13. The results are also reported in Table 6.2. There are two kinds of behaviors for tools depending on the applied method. For the first kind which corresponds to the Chernoff-Hoeffding method, the

[1]These experiments have been executed on a MacBook Pro, with processor 2.4 GHz Intel Core 2 Duo.

Figure 6.12: Runtime for a Probability Measure of the Philosophers Model



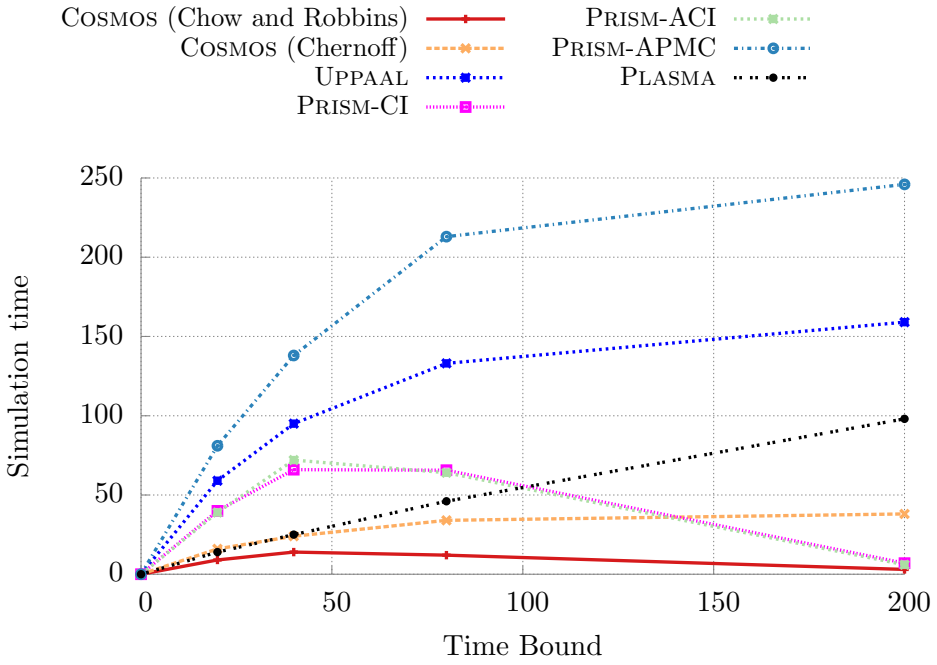

| Philo | Cosmos | Cosmos C-H | Uppaal | Prism CI | Prism ACI | Prism APMC | Plasma |
|---|---|---|---|---|---|---|---|
| 5 | 5 | 8 | 9.4 | 6 | 6 | 13 | 13 |
| 10 | 10 | 21 | 30 | 19 | 20 | 48 | 43 |
| 20 | 26 | 79 | 118 | 60 | 64 | 207 | 159 |
| 50 | 90 | 467 | 827 | 218 | 221 | 1369 | 1097 |
| 100 | 166 | 2140 | 3200 | 424 | 323 | 5450 | 4181 |

Table 6.1: Runtime comparison for the DPM

simulation time increases with respect to the time bound $T$. About 295000 trajectories are required to obtain the specified confidence interval. For the second kind which corresponds to sequential confidence interval methods, the required number of samplings decreases when the time bound goes to infinity. This phenomenon is a consequence of the evolution of the satisfaction probability of $\phi_{TQS}$ that goes to 1 when $T$ goes to infinity.

Among the tools using Chernoff-Hoeffding method, Cosmos is again the fastest. When the time bound is 200, Plasma is 2.58 times slower, Uppaal is 4.18 times slower and Prism-APMC is 6.47 times slower. Among the tools using sequential procedure, when the time bound is 40, the two versions of Prism have similar runtime and Cosmos is up to 2.85 time faster.

Figure 6.13: Runtime for a probability measure of the TQS model




| Time | Cosmos | Cosmos C-H | Uppaal | Prism CI | Prism ACI | Prism APMC | Plasma |
|---|---|---|---|---|---|---|---|
| 20 | 9 | 16 | 59 | 40 | 39 | 81 | 14 |
| 40 | 14 | 24 | 95 | 66 | 72 | 138 | 25 |
| 80 | 12 | 34 | 133 | 65.6 | 64 | 213 | 46 |
| 200 | 3 | 38 | 159 | 7 | 6 | 246 | 98 |

Table 6.2: Runtime comparison for the TQS

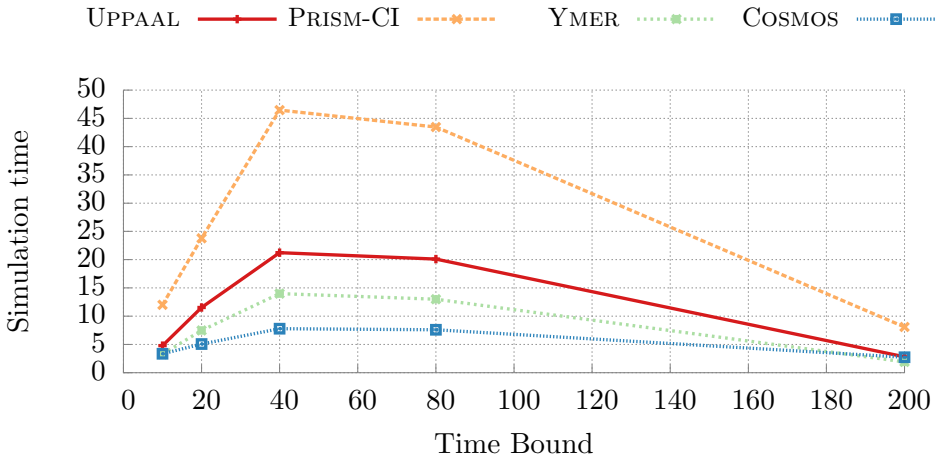
**TQS experiments with sequential hypothesis testing.**

Results on hypothesis testing are reported in Table 6.3. Each value is the mean of 100 experiments. The threshold value for the hypothesis is always very close the numerical value in order to increase the number of trajectories performed by tools. Results confirm that hypothesis testing methods are faster than confidence interval based methods. All tools generate a similar number of trajectories. In most cases Cosmos is the fastest, Ymer being up to 1.8 times slower, Uppaal2.8 times slower, and Prism 6 times slower.

**SMC Comparison Conclusion**

While the formalisms for modeling the system are quite different and address specific kinds of applications, specifications of properties are somewhat

Figure 6.14: Runtime for a probability measure of the TQS model for Sequential Testing




| Time | NumValue | $p \geq ?$ | Uppaal | Prism | Ymer | Cosmos |
|---|---|---|---|---|---|---|
| 10 | 0.17505 | 0.17 | 4.78 | 12.02 | 3.29 | 3.36 |
| 20 | 0.33574 | 0.33 | 11.54 | 23.78 | 7.48 | 5.08 |
| 40 | 0.56931 | 0.564 | 21.23 | 46.47 | 14.00 | 7.78 |
| 80 | 0.81894 | 0.814 | 20.10 | 43.46 | 13.01 | 7.60 |
| 200 | 0.98655 | 0.981 | 2.81 | 8.10 | 1.92 | 2.74 |

Table 6.3: Runtime comparison for the TQS for Sequential Testing

similar. We performed experiments on classical benchmarks using these tools. They have shown that the choice of the statistical test has a big impact on the simulation time. Thus tool developers should provide as many tests as possible to let users choose which one best fits their needs. For the same statistical procedure, we observed that differences of speed can be as big as six times faster. What remains to be done is to perform these comparisons on large scale case studies, contrary to the benchmark we choose here. However some difficulties may arise for the modeling of such case studies in every formalism.

## 6.7 Importance Sampling Benchmark

### 6.7.1 Global overflow in tandem queues

The first example we study is the running example of the tandem queue system. Recall that the running example consists of two waiting queues in tandem ($M/M/1$), where client arrivals are distributed following a Poisson distribution of parameter $\lambda$. Clients are served in the first queue with rate

$\rho_1$ and in the second queue with rate $\rho_2$. We are interested in computing the probability of a global overflow, that is the probability to reach a state where the total number of clients is larger than a threshold $N$, starting from a state with one client in the first queue, without reaching a state where the system is empty. As we are interested in rare-event probability, we suppose that the system is stable, that is $\rho_1 < \lambda$ and $\rho_2 < \lambda$.

We extend this system to $d$ queues in tandem, the probability of interest is unchanged and the reduced model is built by bounding all the queues except the first one by $R$. Figure 6.15 shows Petri nets modeling the system and its reduction. The mapping function $f$ iteratively maps the exceeding clients of each queue to the previous one starting from the last one.

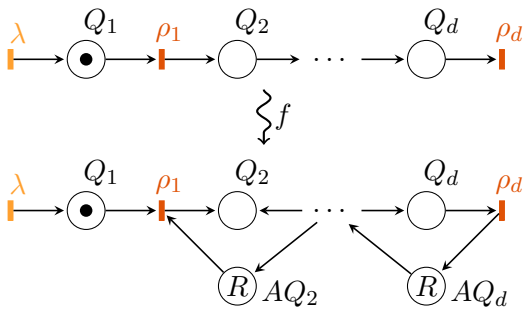



Figure 6.15: Tandem Queues with several queues

This system is a classical benchmark for importance sampling methods on queuing network. Even if this example is simple to describe, several issues arise when designing importance sampling for it. In [71], an empirical state-independent importance sampling is designed. It consists in exchanging the arrival rate and smallest serving rate. It was proved later in [80] that this importance sampling is optimal among the state-independent important samplings. In [24], an exhaustive analysis of state-independent important sampling shows that some values of $\lambda, \rho_1, \rho_2$ in the importance sampling lead to infinite variance. In [29], a method to build efficient state-dependent importance sampling is described. This method is asymptotically optimal when some of its parameters tend to zero.

In Table 6.4, we compare our method to several of these importance samplings. Experiments have been performed for several values of $d$ and $N$. The confidence interval has been computed with a confidence level of 0.95. The following table shows the parameters used depending of the value of $d$, the second column indicates the parameter $R$ used to bound the queues in

| d | N | Exp | Num. | Cosmos | | DSW | | PW |
|---|---|---|---|---|---|---|---|---|
| | | | | Mean | C.I | Mean | C.I | Mean |
| | 30 | $10^{-18}$ | 2.634 | 2.65 | [2.63, 2.66] | 2.73 | [2.37, 3.09] | $9.7{\cdot}10^{-1}$ |
| 2 | 40 | $10^{-24}$ | 1.034 | 1.04 | [1.03, 1.05] | 1.05 | [0.99, 1.11] | $1.4{\cdot}10^{-1}$ |
| | 50 | $10^{-31}$ | 3.801 | 3.79 | [3.75, 3.83] | 3.75 | [3.43, 4.07] | $4.4{\cdot}10^{-1}$ |
| | 20 | $10^{-12}$ | 2.043 | 2.04 | [2.03, 2.05] | 2.05 | [1.97, 2.13] | $7.2{\cdot}10^{-3}$ |
| 4 | 25 | $10^{-16}$ | 5.021 | 5.02 | [4.99, 5.05] | 5.07 | [4.93, 5.21] | $8.5{\cdot}10^{-3}$ |
| | 30 | $10^{-19}$ | 1.099 | 1.11 | [1.10, 1.12] | 1.08 | [1.02, 1.14] | $1.8{\cdot}10^{-3}$ |
| | 20 | $10^{-14}$ | 3.168 | 3.17 | [3.16, 3.18] | 2.93 | [2.47, 3.39] | $7.7{\cdot}10^{-6}$ |
| 9 | 25 | $10^{-19}$ | 9.397 | 9.40 | [9.36, 9.43] | 10.80 | [8.20, 13.1] | $4.4{\cdot}10^{-6}$ |
| | 30 | $10^{-23}$ | 2.154 | 2.14 | [2.13, 2.15] | 1.98 | [1.38, 2.58] | $3.8{\cdot}10^{-7}$ |

Table 6.4: Comparison of different importance samplings for computing the overall overflow probability on tandem queues

the reduced model:

| $d$ | $R$ | $\lambda$ | $\rho_1 = \rho_2 = \cdots = \rho_d$ | Number of trajectories |
|---|---|---|---|---|
| 2 | 4 | 0.1 | 0.45 | $20,000$ |
| 4 | 3 | 0.04 | 0.24 | $20,000$ |
| 9 | 2 | 0.01 | 0.11 | $100,000$ |

For the sake of readability, the exponent has been factorized for each line and is reported in the third column.

- The fourth column reports results computed numerically with the tool Prism.
- Columns 5 and 6 report the result of our method using the tool Cosmos.
- Columns 7 and 8 report experiments performed in [29] using an asymptotically optimal importance sampling.
- The last column reports experiments performed using the importance sampling described in [71], which is an optimal state-independent importance sampling.

We observed that in all cases our method provides tighter confidence intervals than the ones of DSW (at least 5 times tighter and at most 70) and contains the real value. For PW, we observed a classical caveat of importance sampling, that is underestimation of the value. As the value is underestimated, confidence interval computed in this case were not correct and thus were not reported. This caveat will be discussed in more details on experiments where guaranteed variance reduction cannot be obtained.

We now analyze the memory consumption of the different variants of our method compared to numerical computations. The number of states in the model is bounded by $N^d$ and the number of states in the reduced model is bounded by $N \cdot (R+1)^{d-1}$. Table 6.5 reports results of experiments with

| d | Size of $\mathcal{C}$ | Prism | | Reduce Model, Prism | | | Cosmos | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Mem. | Time | Size | Time | Mem. | Time S | Time N | Mem. |
| 1 | 51 | 57 | 0.052 | 51 | 2.0 | 57 | 7 | 8 | 67 |
| 2 | 1,326 | 59 | 0.22 | 198 | 1.7 | 58 | 11 | 12 | 69 |
| 3 | 23,426 | 68 | 0.73 | 768 | 1.7 | 60 | 17 | 20 | 70 |
| 4 | 316,251 | 92 | 1.4 | 2,976 | 1.7 | 75 | 27 | 29 | 72 |
| 5 | 3,478,761 | 181 | 39 | 11,520 | 1.7 | 102 | 47 | 51 | 73 |
| 6 | 32,468,436 | 747 | 580 | 44,544 | 2.0 | 187 | 101 | 109 | 74 |
| 7 | 264,385,836 | 5,174 | 6,635 | 172,032 | 4.1 | 199 | 341 | 372 | 76 |
| 8 | $1.9 \cdot 10^9$ | 36,736 | 75,940 | 663,552 | 12 | 229 | 1,666 | 1,773 | 103 |
| 9 | $12.6 \cdot 10^9$ | Out of Mem. | | 2,555,904 | 50 | 279 | 10,470 | 11,536 | 388 |
| 10 | $75.4 \cdot 10^9$ | | | 9,830,400 | 217 | 313 | 51,377 | 51,840 | 1,491 |
| 11 | $418.1 \cdot 10^9$ | | | 37,748,736 | 964 | 787 | 284,726 | 256,165 | 6,432 |

Table 6.5: Time of memory consumption for the computation of global overflow for the tandem queues

$N = 50, R = 3$ and with increasing values of $d$. The number of samples used by Cosmos is equal to $100,000$. The second column reports the size of the DTMC. Columns 3 and 4 report the memory (in megabytes) and time (in seconds) required by Prism to perform the computation. Column 5 reports the size of the DTMC of the reduced model. Columns 6 and 7 report the time and memory required by Prism to compute the vector $\mu^\bullet$. Finally columns 8, 9 and 10 report the time and memory used by Cosmos to perform the simulations using either important sampling with sink state (S) or important sampling with normalization (N). As the memory consumption of these two importance samplings are equal, it is reported only once.

The size of the DTMC grows exponentially with $d$ as well as the time and memory required by Prism to perform the computation. For $d = 9$, the required memory exceeds the maximum of 42GB available on the computer used for the experimentation. The size of the reduced model also grows exponentially with $d$ but at a much slower pace. The computation of $\mu^\bullet$ is therefore always tractable. The time required by Cosmos for the simulation also increases but is much smaller than the time required by Prism when the model becomes big ($d \geq 6$). Times required for the importance sampling with sink state and with normalization are similar.

Table 6.6 and 6.7 show statistical results. The first column represents the number of queues. The exponent has been factorized to the second column to save space. The third column reports numerical results when available. The fourth column reports the value of $\mu^\bullet(s_0^\bullet)$ computed with Prism on the reduced model. For the two importance samplings, the mean value, the confidence interval and the confidence interval width are reported. For the importance sampling with sink state, confidence intervals are computed using bounds on Bernouilli law. For the importance sampling with normalization, Chernoff-Hoeffding bounds are used and intervals computed using Gaussian analysis are also provided.

| d | Exp | $\mu(s_0)$ | $\mu^\bullet(s_0^\bullet)$ | I.S. with Sink State | | |
|---|---|---|---|---|---|---|
| | | | | Mean | CI | Width |
| 1 | $10^{-52}$ | 8.519 | 8.52 | 8.519 | [8.518, 8.519] | 0.001 |
| 2 | $10^{-50}$ | 4.251 | 5.29 | 4.256 | [4.238, 4.273] | 0.034 |
| 3 | $10^{-48}$ | 1.082 | 1.35 | 1.081 | [1.077, 1.086] | 0.009 |
| 4 | $10^{-47}$ | 1.871 | 2.27 | 1.866 | [1.858, 1.873] | 0.014 |
| 5 | $10^{-46}$ | 2.474 | 2.93 | 2.471 | [2.463, 2.480] | 0.017 |
| 6 | $10^{-45}$ | 2.667 | 3.09 | 2.669 | [2.661, 2.678] | 0.017 |
| 7 | $10^{-44}$ | 2.440 | 2.78 | 2.440 | [2.433, 2.448] | 0.015 |
| 8 | $10^{-43}$ | 1.948 | 2.18 | 1.947 | [1.942, 1.953] | 0.011 |
| 9 | $10^{-42}$ | | 1.53 | 1.383 | [1.379, 1.387] | 0.007 |
| 10 | $10^{-42}$ | | 9.76 | 8.904 | [8.882, 8.927] | 0.045 |
| 11 | $10^{-41}$ | | 5.70 | 5.243 | [5.231, 5.256] | 0.025 |

Table 6.6: Numerical and statistical results for the computation of global overflow for the tandem queues using importance sampling with sink state.

The observed confidence intervals are always very tight around the mean value and contain the numerical value when it is available. The first line $d = 1$ is reported for completeness but is not relevant as the reduced model is equal to the initial model.

On this model, the importance sampling with normalization with Chernoff-Hoeffding bounds is less effective than the one with sink state. This is due to the Chernoff-Hoeffding bounds which are very conservative. Asymptotical confidence intervals computed using Gaussian analysis are also provided. On this set of experiments, the Gaussian intervals are tighter than the Chernoff-Hoeffding and Bernouilli ones, and still contain the numerical value when available.

### 6.7.2 Local Overflow in Tandem Queues

We consider the tandem queue system with two queues with a different property to check: `The second queue contains` $N$ `clients` ($n_2 = N$) `before the second queue is empty` ($n_2 = 0$). The state space is $S = \mathbb{N} \times [0..N]$ with initial state $(0, 1)$. Contrary to the first example, $\mathcal{C}$ is now infinite but $\mathcal{C}^\bullet$ must be finite in order to apply the numerical model checker.

The reduced model behaves as the original one until the first queue contains $R$ clients. Then the model behaves as if there was an infinite number of clients in the first queue. The Petri net is depicted in Figure 6.16(a). The corresponding reduction function $f$ (whose effect on the original chain is represented in Figure 6.16(b)) is defined by:

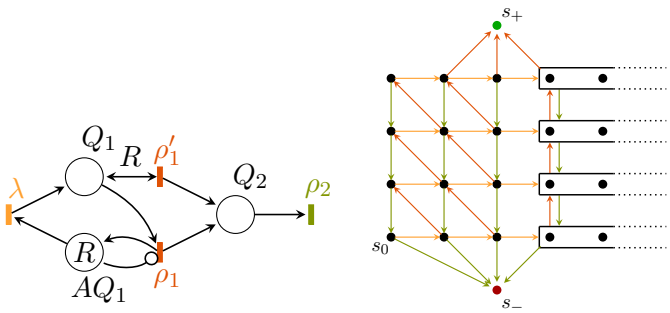
$$f(n_1, n_2) = \begin{cases} (n_1, n_2) & \text{if } n_1 \leq R \\ (R, n_2) & \text{otherwise} \end{cases}$$

| d | Exp | $\mu(s_0)$ | $\mu^\bullet(s_0^\bullet)$ | Mean | Chernoff-Hoeffding | | Gaussian | |
|---|---|---|---|---|---|---|---|---|
| | | | | | CI | width | CI | width |
| 1 | $10^{-52}$ | 8.519 | 8.52 | 8.519 | [8.478, 8.519] | 0.041 | [8.518, 8.519] | 0.001 |
| 2 | $10^{-50}$ | 4.251 | 5.29 | 4.251 | [4.226, 4.277] | 0.051 | [4.244, 4.258] | 0.014 |
| 3 | $10^{-48}$ | 1.082 | 1.35 | 1.081 | [1.075, 1.087] | 0.013 | [1.080, 1.082] | 0.002 |
| 4 | $10^{-47}$ | 1.871 | 2.27 | 1.869 | [1.859, 1.880] | 0.022 | [1.867, 1.871] | 0.004 |
| 5 | $10^{-46}$ | 2.474 | 2.93 | 2.474 | [2.460, 2.488] | 0.028 | [2.472, 2.476] | 0.004 |
| 6 | $10^{-45}$ | 2.667 | 3.09 | 2.667 | [2.652, 2.682] | 0.030 | [2.665, 2.669] | 0.004 |
| 7 | $10^{-44}$ | 2.440 | 2.78 | 2.439 | [2.426, 2.453] | 0.027 | [2.438, 2.441] | 0.003 |
| 8 | $10^{-43}$ | 1.948 | 2.18 | 1.948 | [1.938, 1.959] | 0.021 | [1.947, 1.949] | 0.002 |
| 9 | $10^{-42}$ | | 1.53 | 1.385 | [1.378, 1.392] | 0.015 | [1.384, 1.386] | 0.001 |
| 10 | $10^{-42}$ | | 9.76 | 8.908 | [8.861, 8.955] | 0.094 | [8.904, 8.911] | 0.007 |
| 11 | $10^{-41}$ | | 5.70 | 5.245 | [5.218, 5.272] | 0.055 | [5.243, 5.247] | 0.004 |

Table 6.7: Numerical and statistical results for the computation of global overflow for the tandem queues using importance sampling with normalization.

Table 6.8: Experimental results for the local overflow in tandem queues

| N | R | T(s) | Size of | Exp $\mu^\bullet(f(s_0))$ | | Cosmos | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\mathcal{C}^\bullet$ | $\mathcal{C}^\bullet$ | | | $\mu(s_0)$ | C.I. | T(s) | Nb Traj. |
| 25 | 12 | $\approx 0$ | 338 | $10^{-6}$ | 11.6 | 1.48 | 0.283 | 2 | 5000 |
| 50 | 29 | $\approx 0$ | 1530 | $10^{-11}$ | 29.8 | 3.81 | 0.719 | 13 | 5000 |
| 100 | 66 | 1.44 | 6767 | $10^{-20}$ | 18.7 | 4.22 | 0.734 | 17 | 3000 |
| 500 | 370 | 1770 | 185871 | $10^{-91}$ | 10.3 | 6.63 | 0.805 | 37 | 2000 |
| 1000 | 740 | 24670 | 741741 | $10^{-179}$ | 324 | 3.95 | 4.0 | 180 | 3000 |




(a) Reduced model (b) Reduction chain

Figure 6.16: Petri net for the local overflow in tandem queues ($R = 3, N = 5$)

We found by running experiments on small values of $N$ and $R$ that for $R \geq 0.74 \times N$ we have $\mu(s_0) \geq \mu^\bullet(f(s_0))/10$. This example shows that we can apply our method on an infinite model subject to the specification of a finite reduced model. Observe that computation time reductions w.r.t. standard statistical model checking are still impressive.

### 6.7.3 Bottleneck in Tandem Queues

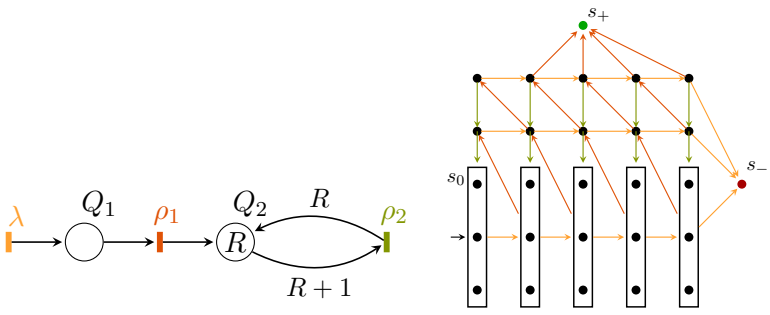
We consider the tandem queue system with another property to check: `The second queue is full` ($n_2 = N$) `before the first one` ($n_1 = N$). The reduced model is obtained by considering that the second queue is full when it contains $N - R$ clients or in an equivalent way that the second queue always contains at least $R$ clients. The corresponding Petri net is depicted in Figure 6.17(a). The reduction function (whose effect on the original chain is represented in Figure 6.17(b)) is defined by:

$$f(n_1, n_2) = \begin{cases} (n_1, R) & \text{if } n_2 \leq R \\ (n_1, n_2) & \text{otherwise} \end{cases}$$

However, the experimental results are not satisfactory since $\mu(s_0) \ll \mu^\bullet(f(s_0))$ when $R$ is small compared to $N$. This shows that designing a reduced model with relevant computation time reduction is sometimes tricky (and remains to be done for this example).

### 6.7.4 Parallel Random Walk

The Petri net depicted in Figure 6.18 models a parallel random walk of $N$ walkers. A walk is done between position 1 and position $L$ starting in position $L/2$ and ends up in the extremal positions. At every round, some random walker can randomly go in either direction. However when walkers $i$ and $i + 1$ are in the same position, walker $i$ can only go toward 1. We




(a) Reduced model (b) Reduction function

Figure 6.17: Petri net for the tandem queues ($R = 2, N = 5$)

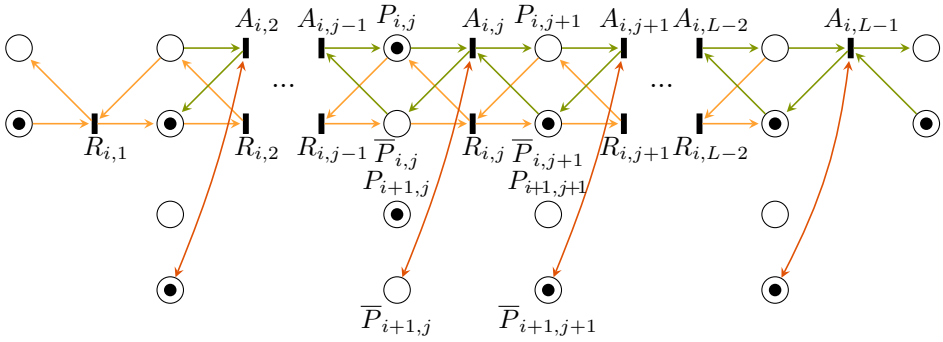



Figure 6.18: The Petri net for the parallel random walk

represent on this figure the walker $i$ and its interactions with walker $i+1$. Transition $A_{i,j}$ (resp. $R_{i,j-1}$) corresponds to a step toward $L$ (resp. 1).

This model is a paradigm of failure tolerant systems in which each walker represents a process which finishes its job when it reaches position 1. Failures may occur and drive the process away from its goal. When position $L$ is reached, the job is aborted. We want to evaluate the probability that a majority of walkers reaches position $L$.

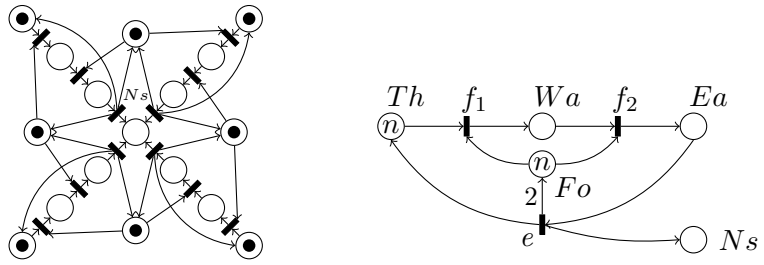
This model has $L^N$ states. In order to get a reduced model, we remove all synchronisations between walkers. Behaviors of all walkers are now independent and thus a state of the reduced system is now defined by the number of walkers in each position. The size of the reduced system is $\binom{N+L-1}{L-1}$.

Proposition 3 holds for this reduced model. Intuitively, removing synchronisation between walkers increases the probability to reach position $L$.

Table 6.9 shows the experimental results with the following parameters $p = 0.3$ , $q = 0.7$, $L = 15$. We stop the simulation when the confidence interval width reaches one tenth of the estimated value. Our method handles

Table 6.9: Experimental results for the parrallel random walk

| N | size of | PRISM Num. | | PRISM Stat. | | | COSMOS | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\mathcal{C}$ | Time | $\mu(s_0)$ | Time | $\mu(s_0)$ | C.I. | Nb Traj. | T $\mathcal{C}^\bullet$(s) | T(s) | $\mu(s_0)$ | Conf. Int. |
| 1 | 15 | $\approx 0$ | 0.00113 | 12 | 1.15E-3 | 1E-4 | 1 | $\approx$0 | $\approx$0 | 0.00113 | 0 |
| 5 | $7.5 \cdot 10^5$ | 6 | 1.88E-9 | 21 | 0 | # | 18000 | 0.5 | 13 | 1.94E-9 | 1.89E-10 |
| 6 | $1.1 \cdot 10^7$ | 127 | 1.14E-12 | No Accepted Path. | | | 53000 | 1 | 57 | 1.17E-12 | 1.17E-13 |
| 7 | $1.7 \cdot 10^8$ | 2248 | 2.93E-12 | | | | 50000 | 2.8 | 186 | 2.92E-12 | 2.89E-13 |
| 8 | $2.0 \cdot 10^9$ | Out of Mem. | | | | | 145000 | 7.9 | 1719 | 1.86E-15 | 1.86E-16 |
| 9 | $3.8 \cdot 10^{10}$ | | | | | | 128000 | 24 | 3800 | 4.7E-15 | 4.75E-16 |
| 10 | $5.7 \cdot 10^{11}$ | | | | | | 371000 | 71 | 26000 | 3.12E-18 | 3.11E-19 |
| 11 | $8.0 \cdot 10^{12}$ | | | | | | 321000 | 228 | 67000 | 7.90E-18 | 7.89E-19 |




(a) $\mathcal{M}$ with $N = 4$ (b) the reduce model $\mathcal{M}^\bullet$

Figure 6.19: Petri net for the dining philosophers

huge models (with size up to 8 $10^{12}$) with very small probabilities (8 $10^{-18}$) whereas the standard statistical model checking and numerical model checking fail due to either the low probability or the size of the system.

### 6.7.5 The Dining Philosophers

The last example is the usual dining philosophers problem which illustrates a problem of concurrency for sharing resources. The transition rate for a philosopher to take his right fork is $\lambda_1$, the one to take its left fork is $\lambda_2$ and the one to put back both forks on the table and start to think again is $\rho$. We are interested in the following property: `All the philosophers are blocked because they have all taken their right fork, before` $r$ `meals are eaten`. The model of the system is a Petri net depicted in Figure 6.19(a) in the particular case of four philosophers. The place in the center of the net ($Ns$) is used to compute the number of eaten meals ($r$).

In the reduced model, philosophers are not sitting while they think. When a philosopher wants to eat, he sits down on the left of another philosopher and takes his right fork, if no philosopher is already sitting, he sits down anywhere. Only the leftmost philosopher is then able to take his left fork. This model is depicted Figure 6.19(b). In this model, the deadlock is more frequent than in the original one. We define the rate as $f_1 = m(Th) \times \lambda_1$, $f_2 = min(m(Wa), m(Fo)) \times \lambda_2$ and $e = m(Ea) \times \rho$ where $m$ is the current marking. The number of states in the original model is at least $2^n \cdot n$ whereas

Table 6.10: Experimental results for the philosophers

| N | size of $\mathcal{C}$ | $\mu(s_0)$ | T (s) numerical | size of $\mathcal{C}^\bullet$ | $\mu^\bullet(f(s_0))$ | $\mu(s_0)$ | T (s) simulation |
|---|---|---|---|---|---|---|---|
| 3 | 56 | $5.822 \cdot 10^{-4}$ | 0.007 | 24 | $5.822 \cdot 10^{-4}$ | $5.822 \cdot 10^{-4}$ | 0.22 |
| 5 | 492 | $1.590 \cdot 10^{-6}$ | 0.028 | 72 | $9.950 \cdot 10^{-7}$ | $1.604 \cdot 10^{-6}$ | 705 |
| 10 | $7.3 \cdot 10^4$ | $2.358 \cdot 10^{-11}$ | 3.45 | 396 | $1.853 \cdot 10^{-12}$ | $6.006 \cdot 10^{-12}$ | 1400 |
| 15 | $8.8 \cdot 10^6$ | $3.402 \cdot 10^{-16}$ | 845 | 1152 | $3.979 \cdot 10^{-17}$ | $5.868 \cdot 10^{-19}$ | 4150 |
| 18 | $1.4 \cdot 10^8$ | | | 1900 | $1.416 \cdot 10^{-19}$ | $2.212 \cdot 10^{-22}$ | 6400 |
| 30 | $9.4 \cdot 10^{12}$ | | | 7936 | $1.566 \cdot 10^{-27}$ | $1.051 \cdot 10^{-38}$ | 16000 |

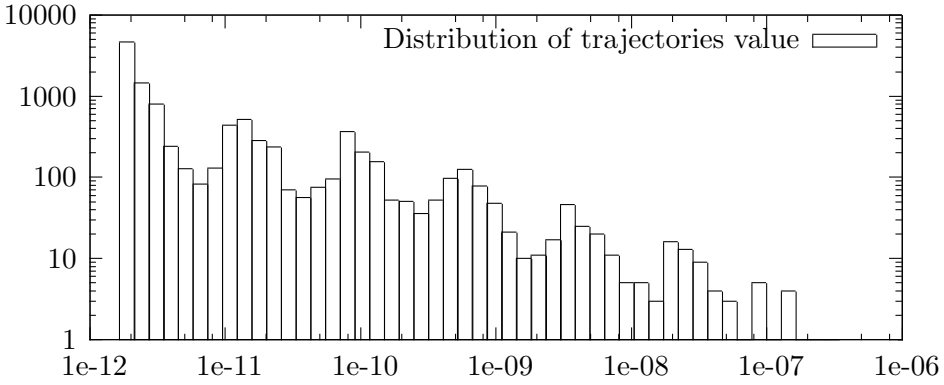
the number of states in the reduced model is at most $n^3$. There is an exponential gap in between the size of the two models which allows one to easily compute $\mu^\bullet$ even on big models.

For this example, our method returns good approximations for small values of $N$ but for larger values the likelihood associated to some rare trajectories is large as can be seen on Histogram 6.20(b) where the impact of trajectories are there frequency multiplied by there likelihood. Thus in this histogram one can observe that a single trajectory with a likelihood of $10^-6$ contribute more to the observable average value of $W_{s_0}$ than the 5000 trajectories with a likelihood of $2 \cdot 10^{-12}$.

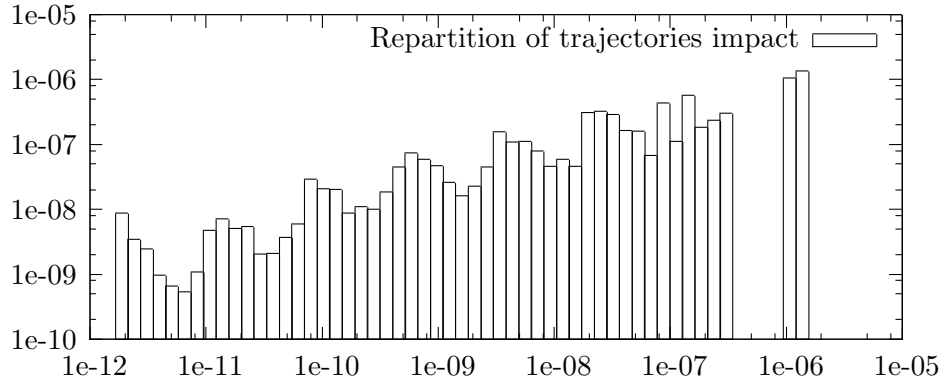
## 6.8 Conclusion

In this chapter, we provided some implementation details and the results of some benchmarks and experiments for the tool Cosmos. From these observations, we can state that Cosmos is well adapted for the analysis of stochastic Petri nets and higher formalisms as it offers most functionalities users are interested in. Its performances in terms of simulation speed are among the best available on typical benchmarks. Thanks to its integration in the verification platform CosyVerif, it benefits from a graphical user interface and can be run as a virtual machine over all common operating systems.

We performed experiments on the importance sampling methods described in Chapter 4 and found that they are often more efficient than other importance samplings methods. For different kinds of models where the state space is too large to be analyzed with numerical methods, these importance sampling methods efficiently estimate the rare-event probability of interest.




(a) Value of trajectories




(b) Impact of trajectories

Figure 6.20: Histogram of the value of trajectory for 10 philosophers

# Chapter 7

# Signaling Cascade

## 7.1 Introduction

In this chapter a case study of biology is presented. This is a join work with Serge Haddad, Monika Heiner and Claudine Picaronny [III].

The case study we consider in this chapter is an example of biological regulatory networks which models how several chemicals species, more specifically proteins and genes, interact to regulate a biological function. These network models systems with continuous behaviors but they can be accurately modeled as discrete systems [76]. As a discrete system they have been widely studied using computer science techniques [72, 47]

Signaling processes is an example of biological regulatory networks playing a crucial role for the regulatory behavior of living cells. They mediate input signals, i.e. the extracellular stimuli received at the cell membrane, to the cell nucleus, where they enter as output signals the gene regulatory system. Understanding signaling processes is still a challenge in cell biology. To approach this research area, biologists design and explore signaling networks, which are likely to be building blocks of the signaling networks of living cells. Among them are the *Ras/Raf/MEK/ERK* pathway which has been extensively studied (see for example this review [60]). This pathway is involved in the control of cell proliferation, differentiation and apoptosis.

A signaling cascade is a set of reactions which can be grouped into levels. At each level a particular enzyme is produced (e.g. by phosphorylation); the level generally also includes the inverse reactions (e.g., dephosphorylation). The system constitutes a cascade since the enzyme produced at some level is the catalyzer for the reactions at the next level. The catalyzer of the first level is usually considered to be the input signal, while the catalyzer produced by the last level constitutes the output signal. The transient behavior of such a system presents a characteristic shape, the quantity of every enzyme increases to some stationary value. In addition, the increases are temporally ordered w.r.t. the levels in the signaling cascade. This behavior can be

viewed as a signal traveling along the levels, and there are many interesting properties to be studied like the traveling time of the signal, the relation between the variation of the enzymes of two consecutive levels, etc.

In [45], it has been shown how such a system can be modeled by a Petri net which can either be equipped with continuous transition firing rates leading to a continuous Petri net which determines a set of differential equations or by stochastic transition firing rates leading to a stochastic Petri net. This approach emphasizes the importance of Petri nets which, depending on the chosen semantics, permit to investigate particular properties of the system. In this paper we wish to explore the influence of stochastic features on the signaling behavior, and thus we focus on the use of stochastic Petri nets.

Analysis of stochastic Petri nets can be performed either numerically or statistically. The former approach is much faster than the latter and provides exact results up to numerical approximations, but its application is limited by the memory requirements due to the combinatory explosion of the state space.

In this chapter we consider three families of properties for signaling cascades that are particularly relevant for the study of their behavior and that are (depending on a scaling parameter) potentially rare events. From an algorithmic point of view, this case study raises interesting issues since the combinatorial explosion of the model quickly forbids the use of numerical solvers and its intricate (quantitative) behavior requires elaborated and different abstractions depending on the property to be checked.

Due to these technical difficulties, the signaling cascade analysis has led us to substantially improve our method and in particular the way we obtain the final confidence interval. From a biological point of view, experiments have pointed out interesting dependencies between the scaling parameter of the model and the probability of satisfying a property.

In Section 7.2 we present the biological background, the signaling cascade under study and the properties to be studied. Then in Section 7.3, we model signaling cascades by stochastic Petri Nets (SPN). Then in Section 7.4 we report and discuss the results of our experiments. Finally in Section 7.5, we conclude and give some perspectives to this work.

## 7.2 Biological Background

In technical terms, signaling cascades can be understood as networks of biochemical reactions transforming input signals into output signals. In this way, signaling processes determine crucial decisions a cell has to make during its development, such as cell division, differentiation, or death. Malfunction of these networks may potentially lead to devastating consequences on the organism, such as outbreak of diseases or immunological abnormalities. Therefore, cell biology tries to increase our understanding of how signaling

cascades are structured and how they operate. However, signaling networks are generally hard to observe and often highly interconnected, and thus signaling processes are not easy to follow. For this reason, typical building blocks are designed instead, which are able to reproduce observed input/output behaviors.

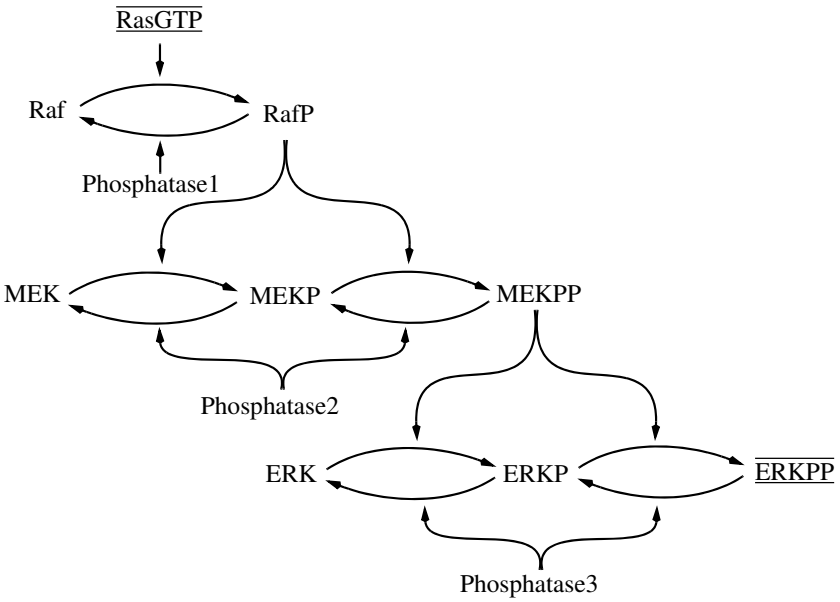



Figure 7.1: The general scheme of the considered three-level signaling cascade; RasGTP serves as input signal and ERKPP as output signal.

The case study we have chosen for our paper is such a signaling building block: the mitogen-activated protein kinase (MAPK) cascade [67]. This is the core of the ubiquitous ERK/MAPK network that can, among others, convey cell division and differentiation signals from the cell membrane to the nucleus. The description starts at the RasGTP complex which acts as an enzyme (kinase) to phosphorylate Raf, which phosphorylates MAPK/ERK Kinase (MEK), which in turn phosphorylates Extracellular signal Regulated Kinase (ERK). We consider RasGTP as the input signal and ERKPP (activated ERK) as the output signal. This cascade (RasGTP $\rightarrow$ Raf $\rightarrow$ MEK $\rightarrow$ ERK) of protein interactions is known to control cell differentiation, while the strength of the effect depends on the ERK activity, i.e. concentration of ERKPP.

The scheme in Figure 7.1 describes the typical modular structure for such a signaling cascade, see [18]. Each layer corresponds to a distinct protein species. The protein Raf in the first layer is only singly phosphorylated. The proteins in the two other layers, MEK and ERK respectively, can be singly as well as doubly phosphorylated. In each layer, forward reactions are catalysed by kinases and reverse reactions by phosphatases (Phosphatase1,

Phosphatase2, Phosphatase3). The kinases in the MEK and ERK layers are the phosphorylated forms of the proteins in the previous layer. Each phosphorylation/dephosphorylation step applies mass action kinetics according to the pattern $A + E \rightleftharpoons AE \to B + E$. This pattern reflects the mechanism by which enzymes act: first building a complex with the substrate, which modifies the substrate to allow for forming the product, and then disassociating the complex to release the product; for details see [16].

Figure 7.2 depicts the evolution of the mean number of proteins with time. At time zero there is a only hundred RasGTP proteins, then we observe the transmission of the signal as successively the number of RafP MEKPP and ERKPP increases.

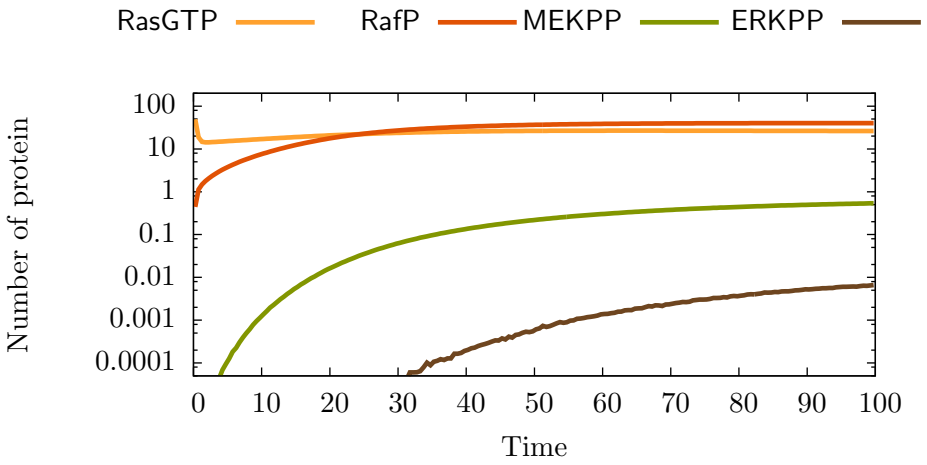



Figure 7.2: Transmission of the signal in the signalling cascade

Having the wiring diagram of the signaling cascade, a couple of interesting questions arise whose answers would shed some additional light on the subject under investigation. Among them are an assessment of the signal strength in each level, and specifically of the output signal. We will consider these properties in Sections 7.4.1 and 7.4.2. The general scheme of the signaling cascade also suggests a temporal order of the signal propagation in accordance with the level order. What cannot be derived from the structure is the extent to which the signals are simultaneously produced; we will discuss this property in Section 7.4.3.

## 7.3 Petri net modeling

We now explain how to model our running case study in the Petri net framework. The signaling cascade is made of several phosphorylation/dephosphorylation steps, which are built on mass/action kinetics. Each step follows the pattern $A + E \rightleftharpoons AE \to B + E$ and is modeled by a small

| N | number of states | N | number of states |
|---|---|---|---|
| 1 | 24,065 (4) | 6 | 769,371,342,640 (11) |
| 2 | 6,110,643 (6) | 7 | 5,084,605,436,988 (12) |
| 3 | 315,647,600 (8) | 8 | 27,124,071,792,125 (13) |
| 4 | 6,920,337,880 (9) | 9 | 122,063,174,018,865 (14) |
| 5 | 88,125,763,956 (10) | 10 | 478,293,389,221,095 (14) |

Table 7.1: Development of the state space for increasing $N$.

Petri net component depicted in Figure 7.3. The mass action kinetics is expressed by the rate of the transitions. The marking-dependent rate of each transition is equal to the product of the number of tokens in all its incoming places up to a multiplicative constant given by the biological behavior (summing up dependencies on temperature, pressure, volume, etc.).

The whole reaction network based on the general scheme of a three-level double phosphorylation cascade, as given in Figure 7.1, is modeled by the Petri net in Figure 7.4. The input signal is the number of tokens in the place RasGTP, and the output signal is the number of tokens in the place ERKPP.

This signaling cascade model represents a self-contained and closed system. It is covered with place invariants (see the appendix), specifically each layer in the cascade forms a P-invariant consisting of all states a protein can undergo; thus the model is bounded. Assuming an appropriate initial marking, the model is also live and reversible; see [45] for more details, where this Petri net has been developed and analyzed in the qualitative, stochastic and continuous modeling paradigms. In our paper we extend these analysis techniques for handling properties corresponding to rare events.

We introduce a scaling factor $N$ to parameterize how many tokens are spent to specify the initial marking. Increasing the scaling parameter can be interpreted in two different ways: either an increase of the biomass circulating in the closed system (if the biomass value of one token is kept constant), or an increase of the resolution (if the biomass value of one token inversely decreases, called level concept in [45]). The kind of interpretation does not influence the approach we pursue in this paper.

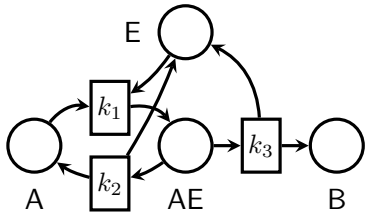



Figure 7.3: Petri net pattern for mass action kinetics $A + E \rightleftharpoons AE \to B + E$.

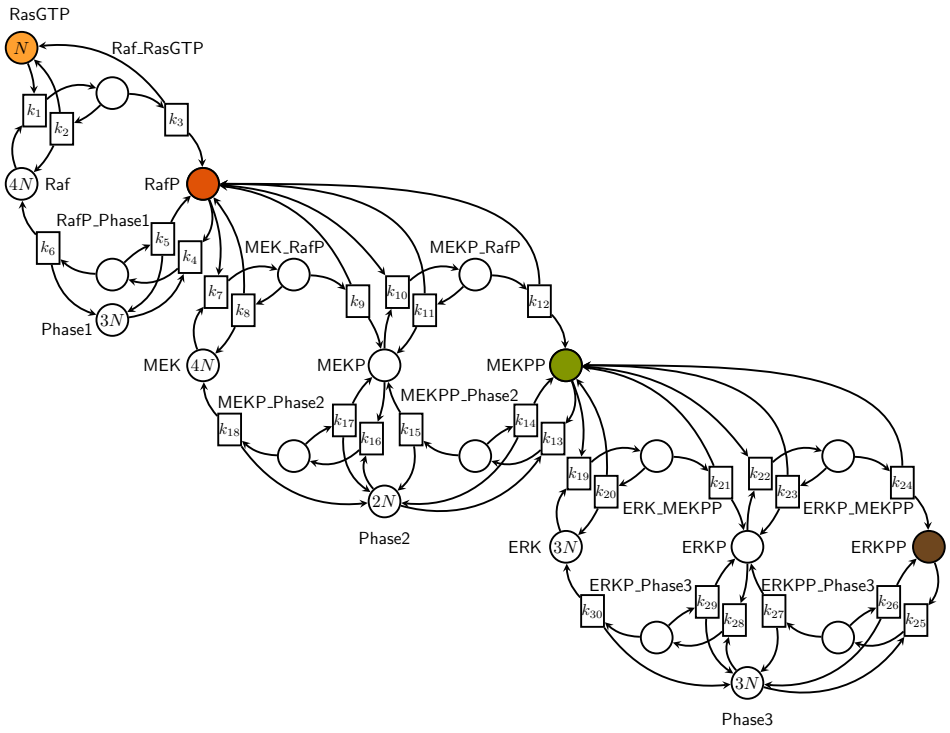



Figure 7.4: A Petri net modelling the three-level signaling cascade given in Figure 7.1; $k_i$ are the kinetic constants for mass action kinetics, $N$ the scaling parameter.

Increasing $N$ means to increase the size of the state space and thus of the CTMC, as shown in Table 7.1 which has been computed with the symbolic analysis tool Marcie [46]. As expected, the explosion of the state space prevents numerical model checking for higher $N$ and thus calls for statistical model checking.

Furthermore, increasing the number of states means to actually decrease the probabilities to be in a certain state, as the total probability of 1 is fixed. With the distribution of the probability mass of 1 over an increasingly huge number of states, we obtain sooner or later states with very tiny probabilities, and thus rare events. Neglecting rare events is usually appropriate when focusing on the averaged behavior. But they become crucial when certain jump processes such as mutations under rarely occurring conditions are of interest.

## 7.4 Experiments

We have analysed three properties, the last two are inspired by [45]. Recall that the initial marking of the model is parametrized by a scaling factor

$N$. For the first two properties, the reduced model is the same model but with local smaller scaling factors on the different layers of phosphorylation. Every state of the initial model is mapped (by $f$) to a state of the abstract model which has the "closest" proportion of chemical species. For instance let $N = 4$ which corresponds to 16 species of the first layer, a state with 6 tokens in Raf and 10 tokens in RafP is mapped, for a reduced model with $N = 3$, to a state with $4 = \lfloor 6 \times 3/4 \rfloor$ tokens in Raf and $8 = \lceil 10 \times 3/4 \rceil$ tokens in RafP (see the appendix for a specification of $f$).

All statistical experiments have been carried out with our tool Cosmos [9]. Cosmos is a statistical model checker for the HASL logic [9]. It takes as input a Petri net (or a high-level Petri net) with general distributions for transitions. It performs an efficient statistical evaluation of the stochastic Petri net by generating a code per model and formula. In the case of importance sampling, it additionally takes as inputs the reduced model and the mapping function specified by a $C$ function and returns the different confidence intervals.

All experiments have been performed on a machine with 16 cores running at 2 GHz and 32 GB of memory both for the statistical evaluation of Cosmos and the numerical evaluation of Marcie.

### 7.4.1 Maximal peak of the output signal

The first property is expressed as a time-bounded reachability formula assessing the strength of the output signal of the last layer: "What is the probability to reach within 10 time units a state where the total mass of ERK is doubly phosphorylated?", associated with probability $p_1$ defined by:

$$p_1 = \mathbb{P}(\text{True } \mathbf{U}^{\leq 10}(\mathsf{ERKPP} = 3N))$$

The inner formula is parametrized by $N$, the scaling factor of the net (via its initial marking). The reduced model that we design for Cosmos uses different scaling factors for the three layers in the signaling cascade. The first two layers of phosphorylation which are based on Raf and MEK always use a scaling factor of 1, whereas the last layer involving ERK uses a scaling factor of $N$. The second column of Table 7.2 shows the ratio between the number of reachable states of the original and the reduced models.

**Experimental Results.**

We have performed experiments with both Cosmos and Marcie. The time and memory consumptions for increasing values of $N$ are reported in Table 7.2. For each value of $N$ we generate one million trajectories with Cosmos. We observe that the time consumption significantly increases between $N = 3$ and $N = 4$. This is due to a change of strategy in the space/time trade-off in order to not exceed the machine memory capacity. Marcie suffers an

| N | Cosmos | | | Marcie | |
|---|---|---|---|---|---|
| | Reduction factor | time | memory | time | memory |
| 1 | - | - | - | 4 | 514MB |
| 2 | 38 | 20,072 | 3,811MB | 326 | 801MB |
| 3 | 558 | 15,745 | 15,408MB | 43,440 | 13,776MB |
| 4 | 4667 | 40,241 | 3,593MB | Out of Memory: >32GB | |
| 5 | 27353 | 51,120 | 19,984MB | | |

Table 7.2: Computational complexity related to the evaluation of $p_1$

| N | Cosmos | | | Marcie |
|---|---|---|---|---|
| | Gaussian CI | Chernoff CI | MinMax CI | Output |
| 1 | | | | $2.07{\cdot}10^{-12}$ |
| 2 | $[3.75{\cdot}10^{-27}, 5.88{\cdot}10^{-26}]$ | $[3.75{\cdot}10^{-27}, 4.54{\cdot}10^{-25}]$ | $[3.75{\cdot}10^{-27}, 1.57{\cdot}10^{-23}]$ | $8.18{\cdot}10^{-26}$ |
| 3 | $[4.34{\cdot}10^{-42}, 1.72{\cdot}10^{-39}]$ | $[4.34{\cdot}10^{-42}, 1.82{\cdot}10^{-38}]$ | $[4.43{\cdot}10^{-42}, 1.87{\cdot}10^{-37}]$ | $2.56{\cdot}10^{-39}$ |
| 4 | $[1.54{\cdot}10^{-57}, 8.54{\cdot}10^{-56}]$ | $[1.54{\cdot}10^{-57}, 1.98{\cdot}10^{-55}]$ | $[1.78{\cdot}10^{-57}, 7.05{\cdot}10^{-55}]$ | Out of Memory |
| 5 | $[3.97{\cdot}10^{-73}, 2.33{\cdot}10^{-70}]$ | $[3.97{\cdot}10^{-73}, 7.30{\cdot}10^{-70}]$ | $[5.44{\cdot}10^{-73}, 2.24{\cdot}10^{-69}]$ | |

Table 7.3: Numerical values associated with $p_1$

exponential increase w.r.t. both time and space resources. When $N = 3$, it is slower than Cosmos and it is unable to handle the case $N = 4$.

Table 7.3 depicts the values returned by the two tools: Marcie returns a single value, whereas Cosmos returns three confidence intervals (discussed above) with a confidence level set to 0.99. We observe that confidence intervals computed by the Gaussian analysis neither contain the result, the ones computed by Chernoff-Hoeffding do not contain it for $N = 3$, and the most conservative ones always contain it (when this result is available). An analysis of the likelihood $L_{s_0}$ is detailed in the appendix.

Figure 7.5 illustrates the dependency of $p_1$ with respect to the scaling factor $N$. It appears that the probability $p_1$ depends on $N$ in an exponential way. The constants occurring in the formula could be interpreted by biologists.

**Mapping function.**

We describe here formally the reduction function $f$. The reduction function must map each marking of the Petri net to a marking of the reduced Petri net.

First we observe that the signaling cascades SPN contains three places invariants of interest:

- The total number of tokens in the set of places {Raf,Raf_RasGTP,RafP_Phase1,RafP, MEK_RafP, MEKP_RafP} is equal to $4N$.

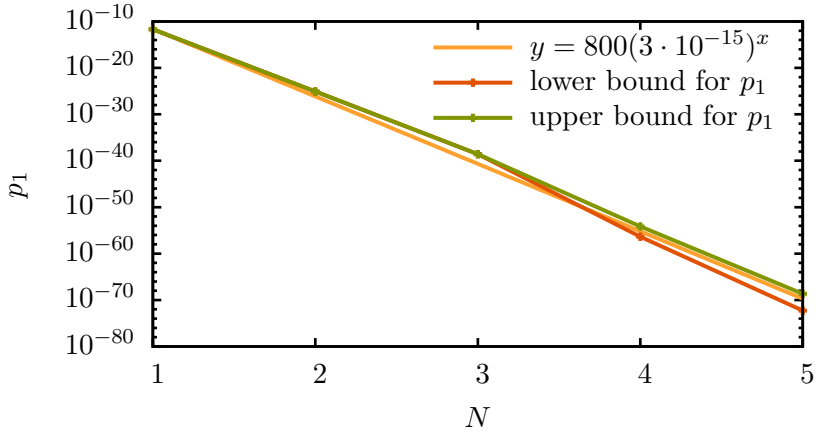



Figure 7.5: Highlighting an exponential dependency

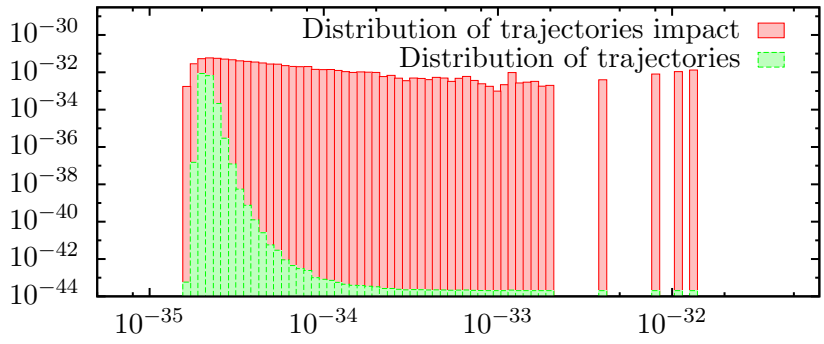



Figure 7.6: Distribution of trajectories and their contribution

- The number of tokens in the set of places {MEK, MEK_RafP, MEKP_Phase2, MEKP, MEKP_RafP, MEKPP_Phase2, MEKPP, ERK_MEKPP, ERKP_MEKPP} is equal to $2N$
- The number of tokens in the set of places {ERK, ERK_RafP, ERKP_Phase2, ERKP, ERKP_RafP, ERKPP_Phase2, ERKPP} is equal to $3N$

We also introduce three sequences of places one for each layer of phosphorylation.

- $S_1$ = [Raf,Raf_RasGTP,RafP_Phase1,RafP]
- $S_2$ = [MEK, MEK_RafP, MEKP_Phase2, MEKP, MEKP_RafP, MEKPP_Phase2,MEKPP]
- $S_3$ = [ERK, ERK_RafP, ERKP_Phase2, ERKP, ERKP_RafP, ERKPP_Phase2, ERKPP]

Let us remark that a marking of the SPN $\mathcal{N}$ is uniquely determined by its values on places in $S_1$, $S_2$ and $S_3$.

We define a function $g$ such that: for all positive integer $m$, positive real number $p$ and vector of integers of size $k$, $\mathbf{v} = (v_i)_1^k$, $g\left(p, m, \mathbf{v}\right)$ is the vector of integers of size $k$, $\mathbf{u} = (u_i)_1^k$, defined by:

$$\forall i > 1,\ u_i = \min\left(\lceil v_i \cdot p \rceil, m - \sum_{l=i+1}^{k} u_l\right) \text{ and } u_1 = m - \sum_{l=2}^{k} u_l$$

One can see that the $g$ is properly defined and that the sum of the components of $\mathbf{u}$ are equal to $m$.

The reduction function $f$ for the two properties is a mapping from the set of states of SPN $\mathcal{N}$ to the set of states of the reduced SPN $\mathcal{N}^\bullet$. This function takes as input the marking of a sequence of places that uniquely define the state. This sequence can be decompose on the three layers of phosphorylation, that is $S_1$ for the first layer, $S_2$ for the second layer and $S_3$ for the last layer.

Recall that layers are not independent one from the others because proteins of one layer are used to activate the following layer; this can be seen on the P-invariant that contains places of the following layer. The mapping function that we construct preserve these P-invariants.

Roughly, on each layer $S_i$, this function $f$ applies a function of the form $g(p_i, m_i, -)$.

Precisely, given a scaling factor $N$ and a scaling factor for each of the three layers of the reduced model, respectively $N_1, N_2$ and $N_3$, the reduction function $f$ maps the marking $m$ on the marking $m^\bullet$ defined as follow:

- $(m^\bullet(p))_{p\in S_3} = g\left(\frac{N_3}{N}, 3N_3, (m(p))_{p\in S_3}\right)$

- $(m^\bullet(p))_{p\in S_2} = g\big(\frac{N_2}{N}, 2N_2 - m^\bullet(\mathsf{ERK_MEKPP}) - m^\bullet(\mathsf{ERKP_MEKPP}), (m(p))_{p\in S_2}\big)$
- $(m^\bullet(p))_{p\in S_1} = g\big(\frac{N_1}{N}, 4N_1 - m^\bullet(\mathsf{MEK_RafP}) - m^\bullet(\mathsf{MEKP_RafP}), (m(p))_{p\in S_1}\big)$

One can see that the three P-invariants are preserved in the reduced model by $f$. We choose $N_1 = N_2 = 1$ and $N_3 = N$.

**Experimental analysis of the likelihood.**

We describe here some technical details of the simulation done for evaluating probability $p_1$. Recall the likelihood of a trajectory requires the distribution of the random variable $W_{s_0}$ (see subsection 4). Proposition 4 ensures that $W_{s_0}$ takes values in $\{0\} \cup [\mu^\bullet_{n^+}(f(s)), \infty[$. Values taken by $L_{s_0}$ are taken by $W_{s_0}$ when at the end of a successful trajectory, therefore these values are in $[\mu^\bullet_{n^+}(f(s)), \infty[$.

We simulate the system for the first formula with $N = 2$ and a discrete horizon of 615 (615 is the right truncation point given by Fox-Glynn algorithm). The result of the simulation is represented as an histogram shown in Figure 7.6. The total number of trajectories is 69000, 49001 of them are not successful. We observe that most of the successful trajectories end with a value close to $2.10^{-35}$, and that a few trajectories have a value close to $10^{-32}$. This is represented by an histogram which is shown as the green part of Figure 7.6 (with a logarithmic scale for the abscissa). We also represent the histogram of the contribution of the trajectories for the estimation of the mean of $L_{s_0}$, that is the red part of the figure (with a logarithmic scale for the ordinate). We observe that the contribution to this mean is almost uniform. Thus a trajectory ending with a likelihood close to $10^{-32}$ have a larger impact than one ending with a likelihood close to $10^34$. This means that an estimator of the mean value of $L_{(s_0,u)}$ will underestimate the expectation of $L_{(s_0,u)}$. To produce a framing of the result, one has to use a very conservative method to avoid underestimating the result.

### 7.4.2 Conditional maximal signal peak

The network structure of each layer in the signaling cascade presents a cyclic behavior, i.e. phosphorylated proteins, serving as signal for the next layer, can also be dephosphorylated again, which corresponds to a decrease of the signal strength. Thus an interesting property of the signaling cascade is the probability of a further increase of the signal strength under the condition that a certain strength has already been reached. We estimate this quantity for the first layer in the signaling cascade, i.e. RafP, and ask specifically for the probability to reach its maximal strength, $4N$: "What is the probability of the concentration of RafP to continue its increase and reach $4N$, when

| $N$ | $L$ | COSMOS | | MARCIE | | |
|---|---|---|---|---|---|---|
| | | confidence interval | time | result | time | memory |
| 2 | 2 | $[2.39{\cdot}10^{-13}$ , $1.07{\cdot}10^{-9}]$ | 31 | $5.55{\cdot}10^{-10}$ | 90 | 802 MB |
| 2 | 3 | $[2.18{\cdot}10^{-10}$ , $6.92{\cdot}10^{-8}]$ | 110 | $6.64{\cdot}10^{-8}$ | 136 | 816 MB |
| 2 | 4 | $[9.33{\cdot}10^{-8}$ , $3.54{\cdot}10^{-5}]$ | 256 | $3.01{\cdot}10^{-6}$ | 276 | 798 MB |
| 2 | 5 | $[1.16{\cdot}10^{-5}$ , $6.08{\cdot}10^{-4}]$ | 1000 | $7.16{\cdot}10^{-5}$ | 759 | 801 MB |
| 2 | 6 | $[5.42{\cdot}10^{-4}$ , $1.21{\cdot}10^{-3}]$ | 5612 | $1.27{\cdot}10^{-3}$ | 3180 | 804 MB |
| 3 | 5 | $[1.82{\cdot}10^{-12}$ , $9.78{\cdot}10^{-9}]$ | 459 | Time > 48 hours | | |
| 3 | 6 | $[3.41{\cdot}10^{-10}$ , $9.66{\cdot}10^{-8}]$ | 1428 | | | |
| 3 | 7 | $[1.81{\cdot}10^{-8}$ , $2.23{\cdot}10^{-6}]$ | 7067 | | | |
| 3 | 8 | $[8.72{\cdot}10^{-7}$ , $2.71{\cdot}10^{-6}]$ | 4460 | | | |
| 3 | 9 | $[1.42{\cdot}10^{-6}$ , $4.59{\cdot}10^{-5}]$ | 4301 | | | |
| 3 | 10 | $[2.69{\cdot}10^{-4}$ , $9.34{\cdot}10^{-4}]$ | 6420 | | | |
| 4 | 10 | $[5.12{\cdot}10^{-9}$ , $2.75{\cdot}10^{-8}]$ | 8423 | Memory > 32GB | | |
| 4 | 11 | $[8.23{\cdot}10^{-8}$ , $2.97{\cdot}10^{-7}]$ | 7157 | | | |
| 4 | 12 | $[9.84{\cdot}10^{-7}$ , $1.86{\cdot}10^{-6}]$ | 18730 | | | |

Table 7.4: Numerical values associated with $p_2$

starting in a state where the concentration is for the first time at least $L$?". This is a special use case of the general pattern introduced in [45].

$$p_2 = \mathbb{P}_\pi((\mathsf{RafP} \geq L) \ \mathbf{U} \ (\mathsf{RafP} \geq 4N))$$

where $\pi$ is the distribution over states when satisfying for the first time the state formula $\mathsf{RafP} \geq L$ (previously called a filter).

This formula is parametrized by threshold $L$ and scaling factor $N$. The results for increasing $N$ and $L$ are reported in Table 7.4 (confidence intervals are computed by Chernoff-Hoeffding method). As before, MARCIE cannot handle the case $N = 3$, the bottleneck being here the execution time.

It is clear that $p_2$ is an increasing function of $L$. More precisely, experiments point out that $p_2$ increases approximatively exponentially by at least one magnitude order when $L$ is incremented. However this dependency is less clear than the one of the first property.

The reduced model is the one used for the first property except for the values of the following parameters: here we choose $N_1 = 1$, $N_2 = N$ and $N_3 = 0$.

### 7.4.3 Signal propagation

To demonstrate that the increases of the signals are temporally ordered w.r.t. the layers in the signaling cascade, and by this way proving the traveling of the signals along the layers, we explore the following property: "What is the probability that, given the initial concentrations of RafP, MEKPP and

ERKPP being zero, the concentration of RafP rises above some level $L$ while the concentrations of MEKPP and ERKPP remain at zero, i.e. RafP is the first species to react?". While this property has its focus on the beginning of the signaling cascade, it is obvious how to extend the investigation by further properties covering the entire signaling cascade.

$$p_3 = \mathbb{P}((\mathsf{MEKPP} = 0) \wedge (\mathsf{ERKPP} = 0)) \,\mathbf{U}\, (\mathsf{RafP} > L))$$

This formula is parametrized by $L$. Due to the lack of space only some values of $L$ in $[0, 4N[$ are reported. The results for increasing $N$ and $L$ are given in Table 7.4.3. As can be observed, the probability to satisfy this property is not a rare event thus no importance sampling is required. Instead results are obtained by a plain Monte Carlo simulation generating 10 millions of trajectories. For $N > 3$ MARCIE requires more than $32GB$ of memory thus the computation was stopped. On the other hand, the memory requirement of COSMOS is around $50MB$ for all experiments.

We also observed that as expected the probability exponentially decreases with respect to $L$.

## 7.5 Conclusion

We have studied rare events in signaling cascades with the help of an improved importance sampling method implemented in COSMOS. As demonstrated by means of our scalable case study, our method has been able to cope with huge models that could not be handled neither by numerical computations nor by standard simulations. In addition, analysis of the experiments has pointed out interesting dependencies between the scaling parameter and the quantitative behavior of the model.

In future work we intend to incorporate other types of quantitative properties, such as the mean time a signal needs to exceed a certain threshold, the mean traveling time from the input to the output signal, or the relation between the variation of the enzymes of two consecutive levels. We also plan to analyse other biological systems for which the evaluation of tiny probabilities might be relevant like mutation rates in growing bacterial colonies [38].

| N | L | Cosmos | | Marcie | | |
|---|---|---|---|---|---|---|
| | | confidence interval | time | result | time | memory |
| 2 | 2 | [0.8018,0.8024] | 4112 | 0.8021 | 75 | 730MB |
| 2 | 3 | [0.4201,0.4209] | 7979 | 0.4205 | 137 | 723MB |
| 2 | 4 | [0.1081,0.1086] | 10467 | 0.1084 | 163 | 725MB |
| 2 | 5 | [0.0122,0.0124] | 11122 | 0.0123 | 123 | 725MB |
| 2 | 6 | $[6.20\cdot10^{-4},6.61\cdot10^{-4}]$ | 11185 | $6.32\cdot10^{-4}$ | 129 | 725MB |
| 2 | 7 | $[1.02\cdot10^{-5},1.61\cdot10^{-5}]$ | 11194 | $1.24\cdot10^{-5}$ | 156 | 725MB |
| 3 | 6 | [0.0136,0.0138] | 14648 | 0.0137 | 17420 | 10.3GB |
| 3 | 7 | $[1.45\cdot10^{-3},1.51\cdot10^{-3}]$ | 14752 | $1.48\cdot10^{-3}$ | 18155 | 10.3GB |
| 3 | 8 | $[9.99\cdot10^{-5},1.17\cdot10^{-4}]$ | 14739 | $1.06\cdot10^{-4}$ | 18433 | 10.3GB |
| 3 | 9 | $[3.53\cdot10^{-6},7.36\cdot10^{-6}]$ | 14734 | $4.86\cdot10^{-6}$ | 18353 | 10.3GB |
| 3 | 10 | $[1.03\cdot10^{-8},9.27\cdot10^{-7}]$ | 14743 | $1.29\cdot10^{-7}$ | 18355 | 10.3GB |
| 3 | 11 | $[0\ ,5.30\cdot10^{-7}]$ | 14766 | $1.48\cdot10^{-9}$ | 18047 | 10.3GB |
| 4 | 8 | $[1.47\cdot10^{-3},1.53\cdot10^{-3}]$ | 17669 | Out of Memory | | |
| 4 | 9 | $[1.52\cdot10^{-4},1.73\cdot10^{-4}]$ | 17628 | | | |
| 4 | 10 | $[9.99\cdot10^{-6},1.59\cdot10^{-5}]$ | 17656 | | | |
| 4 | 11 | $[1.54\cdot10^{-7},1.57\cdot10^{-6}]$ | 17632 | | | |
| 4 | 12 | $[0\ ,5.30\cdot10^{-7}]$ | 17664 | | | |
| 5 | 8 | $[6.92\cdot10^{-3},7.06\cdot10^{-3}]$ | 20367 | | | |
| 5 | 9 | $[1.13\cdot10^{-3},1.19\cdot10^{-3}]$ | 20421 | | | |
| 5 | 10 | $[1.46\cdot10^{-4},1.67\cdot10^{-4}]$ | 20419 | | | |

Table 7.5: Experiments associated with $p_3$

# Chapter 8

# Conclusion and Perspectives

## 8.1 Conclusion

In this thesis, a new method to deal with the quantitative model checking of rare events was presented. This method was thus designed to compute extremely small probabilities with a conservative confidence interval which is of particular interest for critical systems. The rare-event problem was recalled in Chapter 3. To achieve this computation, this method relies on the combination of two well-know methods: Monte Carlo simulations and numerical model checking which were both presented in Chapter 2. This combination is performed by first applying numerical methods over an abstraction of the model and then using the outcoming results to bias the Monte Carlo simulation with importance sampling as shown in Chapter 4. Contrary to other methods using Monte Carlo simulations, the method presented provides given some hypotheses, a *conservative* confidence interval. Additionally, it also guarantees the reduction of the variance with respect to a plain Monte Carlo simulation so that the width of this conservative confidence interval is also smaller.

This method can be used directly on any finite discrete or continuous Markov chain. However, the abstraction of the model has then to be defined manually which can be extremely tedious as the reader might have noticed. To circumvent this difficulty, a framework for probabilistic models was provided in Chapter 5 in which the abstraction is derived automatically.

This method has been implemented in the tool Cosmos. Experiments and a case study have then been performed to demonstrate its efficiency in comparison with numerical methods, plain Monte Carlo simulation and other rare-event methods for Monte Carlo simulation. Details of the implementation and the results of these analyses can be found in Chapters 6 and 7.

In this thesis, we have conducted the design, implementation and testing of an alternative method to deal with the rare-event problem. We have provided a tool which is freely available, and is integrated in the CosyVerif

platform and has been thoroughly tested. Indeed, this work is not finished and many additional features could be introduced. In the following, I provide a non-exhaustive list of these improvements.

## 8.2 Perspectives

Several perspectives were presented all along this manuscript either concerning the theoretical developments, the implementation or the experiments.

Concerning the theoretical aspects, the method with guaranteed variance reduction detailed in Chapter 4 presents several limitations either on its range of application or on its ease of use:

- In its actual formulation, the model to study can be infinite but its reduced model must be finite. This restriction was introduced in order to be able to use numerical model checking algorithms based on linear algebra on the abstraction. However, there exist methods to compute reachability probabilities in infinite systems like quasi-birth-death processes. Using these methods would therefore enable the handling of infinite reduced models and therefore produce better importance sampling.

- Moreover, the models are requested to be Markovian while it is often not the case of real-life systems. As Monte Carlo simulations can deal with non-Markovian models, it will thus be very useful to extend the method to this framework. However, one should ensure that although the initial model is not Markovian, the reduced one is, in order to take advantage of efficient numerical methods on Markovian models. Conditions on the reduced model should then be adapted to maintain the guaranteed variance reduction.

- Finally, from a pragmatic point of view, designing the reduced model and establishing the proof that it satisfies the guaranteed variance property is in practice a difficult and tedious task. Methods to help the construction and/or the proof of such reduced models are thus strongly needed in order for these methods to be used more widely. One solution to this issue was presented in Chapter 5.A natural perspective is to extend the range of this framework by, for instance, letting processes fork and join when changing zone or by allowing processes to be created or destroyed as it is the case in open systems.

Concerning the implementation of the method detailed in Chapter 6, several technical issues still need to be solved. Initially, the tool Cosmos was designed as a simulator. Numerical model-checking methods have been introduced afterwards to deal with rare events. However, they use a naive

representation of the underlying Markov chain while tools dedicated to numerical model checking like Prism or Marcie rely on more involved symbolic methods derived from binary decision diagrams and are thus much more space efficient. For time-unbounded reachability problems, the numerical and simulation parts of the computation are successive. Therefore, the numerical part of the computation can be done by a numerical tool and its results can then be given as input to Cosmos for the simulation part. For time-bounded reachability problems however, the simulation and numerical computation must be performed at the same time and cannot be performed by two different tools. Improving the numerical model checker of Cosmos or developing its interface with an external numerical model checker is thus critical in order to be able to efficiently conduct these calculations. However, both of them will require significant development efforts.

Moreover, experiments presented in Chapter 6 and 7 showed that, even though the methods described in this manuscript can in principle compute probabilities of rare events which were out of reach of classical simulation-based or numerical methods, the design of the reduced model is in practice challenging in order to ensure the guaranteed variance property. Without this property, one can only compute asymptotical confidence intervals that are wider and not conservative. Additional experiments and case studies should be conducted to establish in which domain the method applies the best. Moreover, using the framework described in Chapter 5, additional testing would provide guidelines to decide which kind of generalizations of the framework would be helpful or required to model different types of systems.

Finally, for these methods to reach their full potential, they should be used not only by researchers in computer science but also by other scientists both in academics or in the industry. In order for that to become possible, it implies that much effort should be put into the automation of the critical steps and to the development of a user-friendly interface. A first step has been done in this direction with the development of the CosyVerif platform but much remains to be done. It will require theoretical efforts to design new algorithms, their implementation in efficient and well-documented tools, and also the development of an environment that will help modelers to build systems.

# List of Publications


[I] E. G. Amparore, B. Barbot, M. Beccuti, S. Donatelli, and G. Franceschinis. Simulation-based verification of hybrid automata stochastic logic formulas for stochastic symmetric nets. In *Proceedings of the 2013 ACM SIGSIM conference on Principles of advanced discrete simulation*, SIGSIM-PADS '13, pages 253–264, New York, NY, USA, 2013. ACM.

[II] É. André, B. Barbot, C. Démoulins, L. M. Hillah, F. Hulin-Hubard, F. Kordon, A. Linard, and L. Petrucci. A modular approach for reusing formalisms in verification tools of concurrent systems. In *Proceedings of the 15th International Conference on Formal Engineering Methods (ICFEM'13)*, pages 199–214, 2013.

[III] B. Barbot, S. Haddad, M. Heiner, and C. Picaronny. Rare event handling in signalling cascades. In A. Arisha and G. Bobashev, editors, *Proceedings of the 6th International Conference on Advances in System Simulation (SIMUL'14)*, pages 126–131, Nice, France, Oct. 2014. XPS.

[IV] B. Barbot, S. Haddad, and C. Picaronny. Échantillonnage préférentiel pour le model checking statistique. In *MSR'11*, volume 45 of *Journal Européen des Systèmes Automatisés*, pages 237–252, Lille, France, 2011. Hermès.

[V] B. Barbot, S. Haddad, and C. Picaronny. Coupling and importance sampling for statistical model checking. In *Proceedings of Tools and Algorithm for the Construction and Analysis of Systems (TACAS'12)* Flanagan and König, pages 331–346.

[VI] B. Barbot, S. Haddad, and C. Picaronny. Importance sampling for model checking of continuous time Markov chains. In P. Dini and P. Lorenz, editors, *Proceedings of the 6th International Conference on Advances in System Simulation (SIMUL'13)*, pages 30–35, Lisbon, Portugal, Novembre 2012. IARIA.

# Bibliography


[1] P. A. Abdulla and A. Nylén. Timed Petri nets and BQOS. In *Applications and theory of Petri nets 2001*, pages 53–70. Springer, 2001.

[2] O. Abu-Amsha and J.-M. Vincent. An algorithm to bound functionals of Markov chains with large state space. In *4th INFORMS Conference on Telecommunications*, number 3.2, pages 3–2, 1998.

[3] O. Akin and J. Townsend. Efficient simulation of TCP/IP networks characterized by non-rare events using DPR-based splitting. In *GLOBECOM*. IEEE, 2001.

[4] R. J. Allen, D. Frenkel, and P. R. Ten Wolde. Simulating rare events in equilibrium or nonequilibrium stochastic systems. *The Journal of chemical physics*, 124:024102, 2006.

[5] G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM.

[6] C. Baier, L. Cloth, B. R. Haverkort, M. Kuntz, and M. Siegle. Model checking Markov chains with actions and state labels. *Software Engineering, IEEE Transactions on*, 33(4):209–224, 2007.

[7] C. Baier and J.-P. Katoen. *Principles of model checking*, volume 950. MIT press, 2008.

[8] P. Ballarini, H. Djafri, M. Duflot, S. Haddad, and N. Pekergin. Cosmos: A statistical model checker for the hybrid automata stochastic logic. In *Quantitative Evaluation of Systems (QEST), 2011 Eighth International Conference on*, pages 143–144. IEEE, 2011.

[9] P. Ballarini, H. Djafri, M. Duflot, S. Haddad, and N. Pekergin. HASL: An expressive language for statistical verification of stochastic models. In P. H. Samson Lasaulce, Dieter Fiems and L. Vandendorpe, editors, *Proceedings of the 5th International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'11)*, pages 306–315, Cachan, France, May 2011. ICST.

[10] B. Barbot, P. Ballarini, and H. Djafri. http://www.lsv.ens-cachan.fr/Software/cosmos/.

[11] B. Barbot, T. Chen, T. Han, J.-P. Katoen, and A. Mereacre. Efficient ctmc model checking of linear real-time objectives. In P. A. Abdulla and K. R. M. Leino, editors, *TACAS*, volume 6605 of *Lecture Notes in Computer Science*, pages 128–142. Springer, 2011.

[12] G. Behrmann, A. David, K. G. Larsen, P. Pettersson, and W. Yi. Developing uppaal over 15 years. *Software: Practice and Experience*, 41(2):133–142, 2011.

[13] B. Bérard, F. Cassez, S. Haddad, D. Lime, and O. H. Roux. Comparison of different semantics for time Petri nets. In *Automated Technology for Verification and Analysis*, pages 293–307. Springer, 2005.

[14] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Transactions on Software Engineering*, 17(3):259–273, 1991.

[15] J. H. Blanchet, P. W. Glynn, P. Lécuyer, W. Sandmann, and B. Tuffin. Asymptotic robustness of estimators in rare-event simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 20(1):6, 2010.

[16] R. Breitling, D. Gilbert, M. Heiner, and R. Orton. A structured approach for the engineering of biochemical network models, illustrated for signalling pathways. *Briefings in bioinformatics*, 9(5):404–421, 2008.

[17] P. E. Bulychev, A. David, K. G. Larsen, M. Mikucionis, D. B. Poulsen, A. Legay, and Z. Wang. Uppaal-smc: Statistical model checking for priced timed automata. In H. Wiklicky and M. Massink, editors, *QAPL*, volume 85 of *EPTCS*, pages 1–16, 2012.

[18] V. Chickarmane, B. N. Kholodenko, and H. M. Sauro. Oscillatory dynamics arising from competitive inhibition and multisite phosphorylation. *Journal of Theoretical Biology*, 244(1):68–76, 2007.

[19] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic well-formed colored nets and symmetric modeling applications. *IEEE Transaction Computers*, 42(11):1343–1360, 1993.

[20] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. Greatspn 1.7: Graphical editor and analyzer for timed and stochastic Petri nets. *Perform. Eval.*, 24(1-2):47–68, 1995.

[21] Y. S. Chow and H. Robbins. On the asymptotic theory of fixed-width sequential confidence intervals for the mean. *The Annals of Mathematical Statistics*, pages 457–462, 1965.

[22] E. M. Clarke and P. Zuliani. Statistical model checking for cyber-physical systems. *Automated Technology for Verification and Analysis*, pages 1–12, 2011.

[23] C. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, pages 404–413, 1934.

[24] P.-T. De Boer. Analysis of state-independent importance-sampling measures for the two-node tandem queue. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 16(3):225–250, 2006.

[25] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[26] M. Diaz. *Petri Nets: Fundamental models, verification and applications*. Wiley, 2010.

[27] H. Djafri. *Numerical and Statistical Approaches for Model Checking of Stochastic Processes*. PhD thesis, École Normal Supérieur de Cachan, juin 2012.

[28] S. Donatelli, S. Haddad, and J. Sproston. Model checking timed and stochastic properties with $\text{CSL}^{\text{TA}}$. *IEEE Transactions on Software Engineering*, 35(2):224–240, Mar.-Apr. 2009.

[29] P. Dupuis, A. D. Sezer, and H. Wang. Dynamic importance sampling for queueing networks. *The Annals of Applied Probability*, 17(4):1306–1346, 2007.

[30] E. A. Emerson. Temporal and modal logic. *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, 995:1072, 1990.

[31] E. A. Emerson and E. M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *ICALP*, LNCS 85, 1980.

[32] R. P. Feynman. Personal observations on the reliability of the shuttle. *Report of the Presidential Commission on the Space Shuttle Challenger Accident*, 2:1–5, 1986.

[33] J. Fourneau, M. Lecoz, and F. Quessette. Algorithms for an irreducible and lumpable strong stochastic bound. *Linear Algebra and its Applications*, 386:167–185, 2004.

[34] J.-M. Fourneau and N. Pekergin. An algorithmic approach to stochastic bounds. In M. Calzarossa and S. Tucci, editors, *Performance Evaluation of Complex Systems: Techniques and Tools*, volume 2459 of *Lecture Notes in Computer Science*, pages 64–88. Springer, 2002.

[35] B. L. Fox and P. W. Glynn. Computing poisson probabilities. *Commun. ACM*, 31(4):440–445, 1988.

[36] M. J. Garvels, J.-K. C. Van Ommeren, and D. P. Kroese. On the importance function in splitting simulation. *European Transactions on Telecommunications*, 13(4):363–371, 2002.

[37] M. J. J. Garvels. *The splitting method in rare event simulation*. PhD thesis, Universiteit Twente, 2000.

[38] D. Gilbert, M. Heiner, F. Liu, and N. Saunders. Colouring space-a coloured framework for spatial modelling in systems biology. In *Application and Theory of Petri Nets and Concurrency*, pages 230–249. Springer, 2013.

[39] P. Glasserman and Y. Wang. Counterexamples in importance sampling for large deviations probabilities. *The Annals of Applied Probability*, 7(3):731–746, 1997.

[40] P. W. Glynn and D. L. Iglehart. Importance sampling for stochastic simulations. *Management Science*, pages 1367–1392, 1989.

[41] S. Haddad and P. Moreaux. Sub-stochastic matrix analysis for bounds computation - theoretical results. *European Journal of Operational Research*, 176(2):999–1015, 2007.

[42] J. H. Halton. Sequential Monte Carlo. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 58, pages 57–78. Cambridge Univ Press, 1962.

[43] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal aspects of computing*, 6(5):512–535, 1994.

[44] R. He, P. Jennings, S. Basu, A. P. Ghosh, and H. Wu. A bounded statistical approach for model checking of unbounded until properties. In C. Pecheur, J. Andrews, and E. D. Nitto, editors, *ASE*, pages 225–234. ACM, 2010.

[45] M. Heiner, D. Gilbert, and R. Donaldson. Petri nets for systems and synthetic biology. In M. Bernardo, P. Degano, and G. Zavattaro, editors, *SFM 2008*, volume 5016 of *LNCS*, pages 215–264. Springer, 2008.

[46] M. Heiner, C. Rohr, and M. Schwarick. MARCIE - Model checking And Reachability analysis done effiCIEntly. In J. Colom and J. Desel, editors, *Proc. PETRI NETS 2013*, volume 7927 of *LNCS*, pages 389–399. Springer, June 2013.

[47] M. Herajy and M. Heiner. Towards a computational steering and Petri nets framework for the modelling of biochemical reaction networks. In *CS&P*, pages 147–159, 2012.

[48] T. Herault, R. Lassaigne, and S. Peyronnet. Apmc 3.0: Approximate verification of discrete and continuous time Markov chains. In *Quantitative Evaluation of Systems, 2006. QEST 2006. Third International Conference on*, pages 129–130. IEEE, 2006.

[49] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[50] D. Jansen, J.-P. Katoen, M. Oldenkamp, M. Stoelinga, and I. Zapreev. How fast and fat is your probabilistic model checker? an experimental performance comparison. In K. Yorav, editor, *Hardware and Software: Verification and Testing*, volume 4899 of *Lecture Notes in Computer Science*, pages 69–85. Springer, 2008.

[51] D. N. Jansen. Understanding Fox and Glynn's "computing poisson probabilities". Technical Report ICIS-R11001, Nijmegen: Radboud Universiteit, 2011.

[52] C. Jégourel, A. Legay, and S. Sedwards. Cross-entropy optimisation of importance sampling parameters for statistical model checking. *CoRR*, abs/1201.5229, 2012.

[53] C. Jégourel, A. Legay, and S. Sedwards. A platform for high performance statistical model checking - plasma. In C. Flanagan and B. König, editors, *TACAS*, volume 7214 of *Lecture Notes in Computer Science*, pages 498–503. Springer, 2012.

[54] C. Jegourel, A. Legay, and S. Sedwards. Importance splitting for statistical model checking rare properties. In *Computer Aided Verification*, pages 576–591. Springer, 2013.

[55] S. Juneja and P. Shahabuddin. Rare-event simulation techniques: an introduction and recent advances. *Handbooks in operations research and management science*, 13:291–350, 2006.

[56] H. Kahn and T. E. Harris. Estimation of particle transmission by random sampling. *National Bureau of Standards applied mathematics series*, 12:27–30, 1951.

[57] D. Kartson, G. Balbo, S. Donatelli, G. Franceschinis, and G. Conte. *Modelling with generalized stochastic Petri nets*. Wiley, 1994.

[58] J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen. The ins and outs of the probabilistic model checker MRMC. *QEST*, pages 167–176, 2009.

[59] J. Kemeny and J. Snell. *Finite Markov chains*. Springer, 1960.

[60] W. Kolch. Meaningful relationships: the regulation of the Ras/Raf/MEK/ERK pathway by protein interactions. *Biochem. J*, 351:289–305, 2000.

[61] C. Kollman, K. Baggerly, D. Cox, and R. Picard. Adaptive importance sampling on discrete Markov chains. *Annals of Applied Probability*, pages 391–412, 1999.

[62] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, pages 79–86, 1951.

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

[64] M. Z. Kwiatkowska, G. Norman, and D. Parker. Prism: Probabilistic symbolic model checker. In T. Field, P. G. Harrison, J. T. Bradley, and U. Harder, editors, *Computer Performance Evaluation / TOOLS*, volume 2324 of *Lecture Notes in Computer Science*, pages 200–204. Springer, 2002.

[65] P. L'ecuyer, V. Demers, and B. Tuffin. Rare events, splitting, and quasi-Monte Carlo. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 17(2):9, 2007.

[66] P. L'Ecuyer and B. Tuffin. Approximating zero-variance importance sampling in a reliability setting. *Annals of Operations Research*, 189(1):277–297, 2011.

[67] A. Levchenko, J. Bruck, and P. W. Sternberg. Scaffold proteins may biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties. *Proceedings of the National Academy of Sciences*, 97(11):5818–5823, 2000.

[68] T. Lindvall. *Lectures on the coupling method*. Dover, 2002.

[69] N. A. Lynch. *Distributed algorithms*. Morgan Kaufmann, 1996.

[70] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, 1998.

[71] S. Parekh and J. Walrand. A quick simulation method for excessive backlogs in networks of queues. *Automatic Control, IEEE Transactions on*, 34(1):54–66, 1989.

[72] L. Paulevé, M. Magnin, and O. Roux. Refining dynamics of gene regulatory networks in a stochastic $\pi$-calculus framework. In *Transactions on computational systems biology XIII*, pages 171–191. Springer, 2011.

[73] B. Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. In *Proceedings of the 1985 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '85, pages 147–154, New York, NY, USA, 1985. ACM.

[74] A. Pnueli. The temporal logic of programs. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 46–57. IEEE, 1977.

[75] D. Reijsbergen, P.-T. de Boer, W. Scheinhardt, and B. Haverkort. Automated rare event simulation for stochastic Petri nets. In *Quantitative Evaluation of Systems*, pages 372–388. Springer, 2013.

[76] A. Richard, J.-P. Comet, and G. Bernot. Formal methods for modeling biological regulatory networks. In *Modern Formal Methods and Applications*, pages 83–122. Springer, 2006.

[77] A. Ridder. Asymptotic optimality of the cross-entropy method for Markov chain problems. *Procedia Computer Science*, 1(1):1571–1578, 2010.

[78] G. Rubino and B. Tuffin. *Rare Event Simulation using Monte Carlo Methods*. Wiley, 2009.

[79] R. Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

[80] J. S. Sadowsky. Large deviations theory and efficient simulation of excessive backlogs in a GI/GI/m queue. *Automatic Control, IEEE Transactions on*, 36(12):1383–1394, 1991.

[81] K. Sen, M. Viswanathan, and G. A. Agha. VESTA: A statistical model-checker and analyzer for probabilistic systems. In *QEST*, pages 251–252, 2005.

[82] P. Shahabuddin. Importance sampling for the simulation of highly reliable Markovian systems. *Management Science*, 40(3):333–352, 1994.

[83] N. N. Taleb. *The Black Swan: The Impact of the Highly Improbable Fragility*. Random House LLC, 2010.

[84] M. Y. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Foundations of Computer Science, 1985., 26th Annual Symposium on*, pages 327–338. IEEE, 1985.

[85] A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 06 1945.

[86] H. Younes and R. Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Information and Computation*, 204(9):1368–1409, 2006.

[87] H. L. S. Younes. Ymer: A statistical model checker. In K. Etessami and S. K. Rajamani, editors, *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 429–433. Springer, 2005.