Statistical Inference and Verification of Chemical Reaction Networks

Mahmoud Bentriou

université
PARIS-SACLAY

Statistical Inference and Verification of Chemical Reaction Networks

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 573, Interfaces
Spécialité de doctorat: Mathématiques appliquées
Graduate School : Sciences de l'ingénierie et des systèmes
Référent : CentraleSupélec

Thèse préparée dans l'unité de recherche Mathématiques et Informatique pour la Complexité et les Systèmes (Université Paris-Saclay, CentraleSupélec) sous la direction de Paul-Henry Cournède, Professeur, et le co-encadrement de Paolo Ballarini, Maitre de conférences.

Thèse soutenue à Paris-Saclay, le 8 décembre 2021, par

Mahmoud BENTRIOU

Composition du jury

Pascale Le Gall Professeure, MICS, CentraleSupélec	Présidente de jury
Adeline Leclercq-Samson Professeure des universités (HDR), Laboratoire Jean Kuntzmann, Université Grenoble-Alpes	Rapportrice et examinatrice
Blaise Genest Directeur de recherche CNRS (HDR), IRISA, Université de Rouen	Rapporteur et examinateur
Benoît Barbot Maître de conférences, LACL, Université Paris-Est Créteil	Examinateur
Paul-Henry Cournède Professeur, MICS, CentraleSupélec	Directeur de thèse
Paolo Ballarini Maître de conférences, MICS, CentraleSupélec	Co-encadrant

THESE DE DOCTORAT
NNT : 2021UPAST137

"*La clé de notre rapport aux nombres n'est pas à chercher chez eux, mais entre nous. Cette clé n'est pas mathématique, mais historique et sociale. En d'autres termes, le rapport des hommes aux chiffres reflète la relation des hommes entre eux.*"

Olivier Rey

"*C'est dans le vide de la pensée que s'inscrit le mal.*"

Hannah Arendt

Cette thèse est dédiée à Saadia Ali, ma mère.

"C'est la seule qui m'a assumé,
Mais, c'est aussi la seule qui a su m'aimer.
Donc j'vais continuer de lui dire que c'est la plus belle dame
Avec des cheveux gris,
Tant que son âme et que ses deux yeux brillent."

Limsa d'Aulnay

Remerciements

Tout d'abord, je tenais à remercier mes directeurs de thèse Paul-Henry Cournède et Paolo Ballarini pour l'opportunité de réaliser cette thèse stimulante par ses différents aspects complèxes et variés. Merci Paul-Henry de m'avoir accordé ta confiance dés le début de notre rencontre, d'avoir su être à l'écoute durant toutes ces années pour les concrétiser en l'obtention de mon doctorat, en simultanéité avec ta promotion de Directeur de la Recherche de CentraleSupélec (encore félicitations).

Merci aux rapporteur.trice.s de thèse Adeline Leclercq-Samson et Blaise Genest d'avoir accepté de faire parti du jury, tant pour leurs retours pertinents sur le manuscrit que les questions soulevées impregnées d'intérêt durant la soutenance. Merci à Benoît Barbot qui a rapidement accepté d'examiner le manuscrit après une indisponibilité d'un des membres du jury.

Ces années auraient été bien plus tristes sans l'ambiance et la bienveillance des membres du laboratoire MICS. J'ai été plus que chanceux d'avoir partagé mon bureau avec ces deux personnes brillantes et drôles que sont Brice et Antonin. Bien entendu, je remercie la team Kwak pour les afterworks de qualité. Mathilde, on a commencé ensemble, désolé d'avoir pris un peu de retard. Elvire, bon courage pour la dernière ligne droite. Merci à l'équipe du mésocentre (Laurent, Rémi, Guillaume) qui a permis de rendre le travail de cette thèse bien plus agréable. Merci à Sylvain pour tes meilleures recommandations de festivals, Gautier pour tes tentatives journalières manquées de me faire rendre le manuscrit à temps. Merci au babyfoot de m'avoir permis d'infliger de lourdes défaites aux sus-cités ainsi qu'à Ludovic, Yoann, Romain et d'autres (c'est ma page de remerciement, j'y écris ce que je veux). Merci à Fabienne pour ta disponibilité ainsi que nos discussions. Mais l'expérience au laboratoire est un tout. C'est pour cela que je compte remercier tous ceux qui ont rendu la vie du labo agréable, les anciens (Benoit, Chloé, Pierre, Jean-Christophe, Xiangtuo, Alexandre, Erwan, Sylvie), ceux arrivés après (Walid, Théo, Gurvan, Laura, Agathe, Stefania), ainsi que tous ceux ayant fait un bout de chemin avec moi au laboratoire.

Je remercie bien entendu tous mes amis qui m'ont permis de me sentir très bien entouré pendant ce long périple. Félicitations aux docteur.e.s Mezghani et Nadjahi, deux brillant.e.s chercheur.e.s en devenir, merci pour ce voyage commun traversé ensemble. Merci aux WOATs (Maxime, Antoine, Pierre, Maximilien) pour les SR, le SS, et autres activités périscolaires. Merci Pierre d'avoir assisté sans t'endormir à la soutenance, et d'assurer ma reconversion en tant qu'influenceur sur les réseaux sociaux. Merci Greg d'avoir accompagné les "quelques" coups pris à Paris, merci Mathias pour les concerts et l'initiation à l'escalade malgré mon niveau claqué.

Merci l'Etat français de faciliter l'accès aux études supérieures, espérons que cela reste encore le cas quelques temps.

Pour finir - *last but not least* comme disent les américains -, je remercie ma famille pour tout leur amour et leur soutien indéfectibles, c'est aussi un peu votre diplôme. Merci Maman, pour tout, ta persévérance et ta résilience sont un modèle qui me servira de lanterne toute ma vie durant.

Abstract

Chemical Reaction Networks (CRN) constitute a formalism used to model biological processes. When the population number is not significant and the system is well-stirred, a Continuous-Time Markov Chain describes its stochastic dynamics. This class of model is characterised by the memoryless property: the future state of the system only depends on the current state.

Statistical inference of such CTMCs is complex: likelihood computations are generally intractable. Approximate Bayesian Computation is a recent class of likelihood-free methods for Bayesian inference that allows approximating the posterior distribution with Monte Carlo simulations. It has proven its efficiency in the case of CTMCs.

Model-checking was initially developed for assessing hardware and software systems' reliability. There is a growing interest in the verification of models from Systems Biology to understand the complex molecular interactions within a biological system. Unfortunately, the state space of a CTMC modelled by a CRN quickly explodes or is infinite, which renders its complete exploration infeasible in practice. Statistical model checking methods have been developed to overcome this issue. They simulate the model and compute the ratio of simulations that fulfils a property. Recently, Hybrid Automata Stochastic Logic (HASL) has been introduced for the statistical verification of stochastic models. This temporal logic inherently adopts the statistical point of view of model checking.

In this thesis, we focus on statistical inference and verification of CTMCs defined by CRNs. Our main contribution consists in the new formulation of an Approximate Bayesian Computation procedure combined with HASL called automaton-ABC. We apply this high-level method on several tasks of statistical inference and verification for biological CTMCs, including oscillatory models and time-bounded reachability problems. The implementation of our algorithms is documented and has led to a package in the Julia Programming language.

Résumé

Les réseaux de réactions chimiques (CRN) constituent un formalisme utilisé pour modéliser des processus biologiques. Quand la population est de taille modérée et supposée bien mélangée, le processus stochastique sous-jacent pour décrire ses dynamiques est une chaîne de Markov en temps continu (CTMC). Ce processus est dit sans mémoire: l'état futur du système ne dépend que de l'état courant.

L'inférence statistique de ce type de CTMC est complexe: le calcul de la vraisemblance est en général difficile à résoudre. Les méthodes ABC (Approximate Bayesian Computation) forment une classe de méthodes bayésiennes sans calcul de vraisemblance qui permettent d'approcher la distribution postérieure avec des simulations de Monte-Carlo.

La vérification de modèles, qui fut à l'origine développée pour garantir la fiabilité de systèmes et logiciels informatiques, se penche de plus en plus sur la biologie des systèmes. En effet, il y a un réel besoin de comprendre les interactions complexes entre molécules dans les systèmes biologiques. Malheureusement, l'espace d'états d'un CTMC défini par un CRN explose généralement, voir est infini. Pour palier à cela, des méthodes de vérification statistiques ont été développées. Le principe est de simuler un certain nombre de fois le modèle et de calculer le ratio des simulations qui ont vérifié une propriété. Récemment, une logique temporelle appelée HASL a été introduite pour la vérification statistique de modèles: elle adopte intrinsèquement le point de vue statistique de la vérification.

Dans cette thèse, nous nous intéressons à l'inférence statistique et la vérification statistique de chaines de Markov en temps continu définies par un modèle de réseaux de réactions chimiques. Notre contribution tient principalement dans la formulation d'un algorithme ABC combiné avec le formalisme HASL appelé automaton-ABC. Nous appliquons cette méthode haut niveau sur plusieurs tâches d'inférence statistique et de vérification pour des CTMCs issus de systèmes biologiques, impliquant notamment des modèles oscillatoires et des problèmes d'atteignabilité bornés en temps. L'implémentation des méthodes présentées est documentée et a conduit au développement d'une bibliothèque dans le langage de programmation Julia.

Contents

List of Figures

List of Tables

List of Scripts

List of Listings

Notations

Classical probability theory

ω	An elementary event
Ω	Sample space
$\mathbb{P}_*$	A probability measure
$\mathcal{F}_X$	A σ-field / σ-algebra generated by X
$(\Omega, \mathcal{F}, \mathbb{P})$	A probability space
2^X	Power set of X
$\mathbb{E}[X]$	Expectation of the random variable X
$\mathbb{V}[X]$	Variance of the random variable X
δ_*	Dirac distribution
$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2
$\mathcal{P}(\lambda)$	Poisson distribution with mean λ
$Exp(\lambda)$	Exponential distribution with mean $\frac{1}{\lambda}$

Markov Chains

$[[p, q]]$	The set of integers $\{p, \ldots, q-1, q\}$
$\mathbb{N}$	Space of integers
$\mathbb{R}_{\geq 0}$	Space of non-negative real numbers
CTMC	Abbreviation for Continuous-Time Markov Chain
$\mathbf{S} \subseteq \mathbb{N}^d$	State space of a Markov Chain
$\mathbf{s} \in \mathbf{S}$	State of a CTMC
$\mathbf{s}[i]$	i-th component of $\mathbf{s}$

S	A Markov Chain stochastic process, discrete or continuous time
$(S_n)_{n\in\mathbb{N}}$	A Discrete-Time Markov Chain stochastic process
$(S_t)_{t\in\mathbb{R}_{\geq 0}}$	A Continuous-Time Markov Chain stochastic process
$\mathcal{M}$	A Continuous-Time Markov Chain defined with a transition rate matrix
Q	A transition rate matrix of a CTMC
$P(\mathbf{s}, \mathbf{s}')$	A transition probability
$T_{\mathbf{s}}$	The random sojourn time of state $\mathbf{s}$
$E(\mathbf{s})$	The total exit rate of state $\mathbf{s}$

Paths of a CTMC

$Path(\mathcal{M})$	Set of paths of the CTMC $\mathcal{M}$
$Pr_{\mathcal{M}}, \mathbb{P}_S$	Probability measure induced by a CTMC over the set of paths
$\sigma, S(\omega)$	A path / trajectory of a CTMC
$\delta(\sigma, i)$	Sojourn time of the i-th state of a trajectory
$\sigma[i]$	i-th state of a trajectory
$\sigma[t\rangle$	The trajectory σ shifted at time t

Chemical Reaction Networks

$\mathcal{S}$	Set of species
M	Number of the reaction channels
R_j	A reaction channel
$\eta_j(\mathbf{s}, \theta)$	The kinetic rate of the reaction channel R_j
ν_j	The stoichiometric vector of the reaction channel R_j

Statistical observations

$i.i.d$	independent and identically distributed
$1:N$	The set $\{1, \ldots, N\}$
$X^{(1)}, \ldots, X^{(N)}$	N i.i.d random observations
$x^{(1)}, \ldots, x^{(N)}$	A realisation of N i.i.d observations

$x_{1:N}$	N correlated realisations of observations, e.g. from a Markov Chain

Probability density function

pdf	probability density function
$\propto$	proportional to
π	a pdf
$\tilde{\pi}$	a function proportional to π
$\widehat{\pi}$	an estimation of π
q	a proposal density
K	a kernel density function

Approximate Bayesian Computation

ϵ	Tolerance level
y_{exp}	A dataset / collection of observations
ρ	The distance function
η	The summary statistics function

Model checking

ϕ	A CSL state formula
φ	A CSL path formula or an MITL formula
$\mathsf{true}, \mathsf{false}$	true / false formulae
$\mathbf{F}, \mathbf{G}$	Eventually operator, Globally operator
$Pr(\varphi; \mathcal{M})$	Satisfaction probability of φ
f_{φ}	Satisfaction probability function of φ
(G)SPN	(Generalized) Stochastic Petri Net
$\mathcal{A}$	A Linear Hybrid Automaton (LHA)

Chapter 1

Introduction

1.1 Context

At the time of writing, the world has been hit for about a year by a pandemic of a new coronavirus disease called Covid-19. The pandemic has impacted society a lot, and the media has seized upon two types of questions only scientists were interested in so far:

1. How can one infer the spread of the disease based on previously infected people data?
2. How can we specify properties such as in which region vigilance should be reinforced, the epidemic is stable or over?

The first problem is related to *statistical inference* (Casella and Berger, 2001). Based on probability theory, statistical inference constructs estimators of properties of a general population from samples, quantifies the uncertainty of such estimation and predicts possible future observations based on available data.

The second problem is related to *model checking* (Baier and Katoen, 2008). Based on theoretical computer science, this field groups techniques to check if a model fulfils some specification φ expressed by formal logic. It provides methods that explore the state-space model efficiently in order to assess whether the model verifies the specification φ or not. Initially, it was developed to assess hardware and software systems' reliability, but there is a growing interest in model checking of Systems Biology (Kwiatkowska, Norman, and Parker, 2008).

In this thesis, we focus on statistical inference and verification of stochastic chemical kinetics. Chemical Reaction Network is a formalism that provides a way to model the interactions between entities. A reaction is an event that can occur in the system,

which modifies each entity's quantities. Even if this formalism is originally used to describe chemical systems (Feinberg, 2019), it is also used in gene regulatory networks (Karlebach and Shamir, 2008) or systems of interactions between humans. An illustrative example that is a guideline model in the thesis is the SIR epidemiology model:

$$\begin{aligned} Infection : S + I \xrightarrow{k_i} 2I \\ Recovery : I \xrightarrow{k_r} R \end{aligned} \tag{1.1}$$

This Chemical Reaction Network has three types of entities (living beings in this context): S is the population that can be infected, I is the infected people, whereas R is the recovered/removed population. The system contains two reactions: either an infection or an immunisation/death can occur.

Chemical Reaction Networks are algebraic models, but they induce mathematical models. Traditionally, their dynamics are described by a system of Ordinary Differential Equations (ODE). Such kind of dynamics is continuous and deterministic. If the deterministic model is relevant under some assumptions, such as a high enough species density, they are not adapted to populations of small sizes. With relaxed assumptions, the stochastic dynamics are described by a *Continuous-Time Markov Chain* (CTMC). The main property of such a stochastic process is the *memoryless property* (also known as Markov property): the system's future state only depends on the current state. For example, genetic networks have proven their stochastic nature (Thattai and Van Oudenaarden, 2001).

Statistical inference of CTMCs has been widely studied in the literature, particularly Bayesian statistics (Craciun et al., 2013; Schnoerr, Sanguinetti, and Grima, 2017; Loskot, Atitey, and Mihaylova, 2019). A general issue raised by such a complex stochastic model is the evaluation of the likelihood: it is intractable in most cases. The Bayesian framework allows the use of preexisting expertise on models via the prior distribution and quantifies the uncertainty of parameter inference with the posterior distribution. Also, it offers groups of likelihood-free methods, which allows approximating the posterior distribution. Approximate Bayesian Computation, a specific family of likelihood-free methods, is specifically studied in this thesis as our models have intractable likelihoods. It relies on Monte Carlo simulation methods: the general idea is to draw a parameter θ, simulate the model $y \sim p(.|\theta)$, and keep the sample if y is close enough to observations y_{exp}. Since its first appearance in the literature of population genetics (Tavaré et al., 1997; Pritchard et al., 1999) ABC methods have received much attention from the statistician community (Marin et al., 2011) (Sisson, Fan, and Beaumont, 2018). They have proven efficiency in the case of CTMCs (Warne, Baker, and Simpson, 2019; Alharbi, 2018).

Verification of a CTMC $\mathcal{M}$ has been studied for about two decades since the appearance of CSL temporal logic (Aziz et al., 1996). As CTMCs are stochastic, verification of a CTMC considers the probability of verifying a specification φ. This kind of methods is divided into two approaches. First, the *qualitative approach* assesses whether the probability is above (or below) a *threshold*. Second, the *quantitative approach* is related to the *estimation* of such probabilities. Numerical methods have been developed for these purposes (Baier et al., 2003). To face the problem of state space explosion, statistical methods for verification have been considered since 2002 (Legay, Delahaye, and Bensalem, 2010; Legay et al., 2019). This was enhanced by the growing interest of Systems Biology field in verification methods: the state-space of CTMCs defined by a Chemical Reaction Network quickly explodes or is infinite. Initially, these methods were developed for a single CTMC, but now verification methods tackle model checking with CTMCs indexed by parameters (*parametric CTMC*). The qualitative approach for parametric CTMC is called parameter synthesis. An approximate numerical method has been first proposed (Han, Katoen, and Mereacre, 2008), and also statistical methods (Bortolussi and Silvetti, 2018). For the quantitative approach, statistical methods have been developed (Bortolussi, Milios, and Sanguinetti, 2016).

1.2 Outline

This thesis focuses on statistical inference and verification of parametric Continuous-Time Markov Chains defined by Chemical Reaction Networks. Our main methodological contribution relies on the formulation of a general algorithm called *automaton-ABC*, which uses HASL formalism within the ABC method. With this high-level algorithm, we address several tasks of statistical inference and statistical verification. The considered tools and problems are both issued from model checking and statistical inference, which positions this thesis at the frontier between these two fields. Efforts have been made to be intelligible for scientists of both fields.

The first three chapters are related to the literature.

- Chapter 2 presents the basics of Markov Chains theory. The main properties of Continuous-Time Markov Chains (CTMC) are discussed. We define the probability measure over the set of *càdlàg* functions based on a CTMC. Then, Chemical Reaction Network formalism is defined. We see how a CTMC describes the stochastic dynamics, we discuss the other possible equations of the

dynamics with additional assumptions, and we detail simulation algorithms for CTMC.

- Chapter 3 recalls concepts on statistical methods in inference with a particular focus on Monte Carlo simulations. First, some basic concept about statistical inference and Bayesian inference are discussed. Second, the main Monte Carlo methods are discussed. Third, we describe the Approximate Bayesian Computation (ABC) methods, detail several ABC samplers and discuss ABC applications' stakes. Finally, Kernel Density Estimation is presented.
- Chapter 4 discusses model checking for CTMCs. First, we detail two temporal logics: CSL and MITL. Then, verification of CTMCs are discussed (model checking for a single CTMC $\mathcal{M}$, or parametric CTMC with a collection of CTMCs $(\mathcal{M}_\theta)_{\theta\in\Theta}$) with a particular focus on statistical methods. Finally, the HASL formalism is detailed.

The last two chapters expose contributions.

- Chapter 5 details the statistical inference framework of CTMCs. We present the automaton-ABC algorithm, which combines the synchronised simulation offered by HASL formalism and ABC. We present two applications based on automaton-ABC: detection of oscillatory trends in parametric CTMC and faster implementation of classical ABC inference.
- Chapter 6 details a new statistical method based on automaton-ABC for the quantitative approach of model checking parametric CTMC in time-bounded reachability problems.

The new methods lead to a package in the cross-platform Julia language (Bezanson et al., 2014). The implementation is detailed in Appendix A. Efforts have been made to make it easy to use, efficient and tested. The Scripts List showed above in the thesis groups the scripts that mainly produce the results based on the package. The git repository of the thesis is available at:

`https://gitlab-research.centralesupelec.fr/2017bentrioum/phd-thesis`

A Notations List is available before the introduction. It groups the common notations used in this thesis.

Chapter 2

Markov Chains and Chemical Reaction Networks

This chapter is an introduction to Markov Chains theory. The idea is to present Markov Chains with stochastic process and model checking point of views and see how they describe the dynamics of Chemical Reaction Networks, a formalism that defines the interactions between chemical species.

Markov Chains are a specific class of stochastic models. They are characterised by the *memoryless* property: if a value is known at a specific time, the future is independent of the older values. In our work, this class of models is involved in several tasks:

- Continuous-Time Markov Chains, also called Markov Jump Processes, is the underlying probabilistic model involved in Chemical Reaction Networks, our main study subject.
- Discrete-Time Markov Chains, sometimes simply called Markov Chains, are involved in advanced statistical methods (Markov Chain Monte Carlo, Hidden Markov Models).

In Section 2.1.1, we recall definitions and basic properties of Markov Chains in a classical way for stochastic processes. Section 2.1.2 gives a Markov Chain description frequently used in model checking and based on the transition rate matrix. We describe the set of paths of a CTMC. In Section 2.2, we construct a probability measure over the set of paths of a CTMC. This section is a requisite for the definition of verification tasks of CTMC. In the last Section 2.3, we describe Chemical Reaction Networks formalism for the description of chemical kinetics. We show that the underlying probabilistic model that rules the stochastic dynamics is a CTMC. We

also detail the equations that rule the dynamics, under new assumptions or not, and expose simulation algorithms of trajectories.

The notations related to this chapter are described in the Notations part at the beginning of the thesis. We assume the reader is familiar with classical probability and measure theory notions such as random variable, σ-algebra and probability measure. For further exploration of the theory of Markov Chains, we refer the reader to the books (Bladt and Nielsen, 2017), (Stroock, 2005), (Kulkarni, 1998).

In the whole chapter, Ω is the sample space with $\omega \in \Omega$ an elementary event, $\mathcal{F}$ a σ-field of Ω and $(\Omega, \mathcal{F}, \mathbb{P})$ a probability space. $\mathcal{F}_X$ is the σ-algebra generated by a set X. $\mathbf{S} = \{(\mathbf{s}^{(i)})_{i\in I}\} \subseteq \mathbb{N}^d$ with $I \subseteq \mathbb{N}$ is the countable discrete state set. An element $\mathbf{s} \in \mathbf{S}$ is a d-dimensional vector of integers, and for $i \in \{1, \dots, d\}$, $\mathbf{s}[i]$ is the i-th component. $(S_t)_{t\in\mathbb{T}} \in \mathbf{S}$ is a stochastic process (a collection of random variables) defined on the measurable space $(\Omega, \mathcal{F}, \mathbb{P})$ with $\mathbb{T}$ a countable or continuous set.

2.1 Different perspectives on Markov Chains

This section aims to define discrete space Markov Chains in discrete time and continuous time and detail their main properties. Proof and details of these results can be found in (Bladt and Nielsen, 2017).

2.1.1 Markov Chains as a stochastic process

Discrete-Time Markov Chain (DTMC)

A Discrete-Time Markov Chain is a stochastic process indexed by an integer ($\mathbb{T} = \mathbb{N}$) that fulfils the Markov property. This property states that the current state of the chain only depends on the last state of the chain.

Definition 2.1.1 (Discrete-Time Markov Chain)
$(S_n)_{n\in\mathbb{N}}$ is a time-homogeneous Discrete-Time Markov Chain with state space $\mathbf{S} \subseteq \mathbb{N}^d$ if

1. $\forall n \in \mathbb{N}, \forall \mathbf{s}, \mathbf{s}_n, \dots, \mathbf{s}_0 \in \mathbf{S}, \mathbb{P}(S_{n+1} = \mathbf{s} | S_n = \mathbf{s}_n, \dots, S_0 = \mathbf{s}_0) = \mathbb{P}(S_{n+1} = \mathbf{s} | S_n = \mathbf{s}_n)$ *(Markov property)*
2. $\forall n \in \mathbb{N}, \forall \mathbf{s}, \mathbf{s}' \in \mathbf{S}, \mathbb{P}(S_{n+1} = \mathbf{s}' | S_n = \mathbf{s}) = \mathbb{P}(S_1 = \mathbf{s}' | S_0 = \mathbf{s})$ *(Time-homogeneity)*

In the following, we introduce some classical notations about Markov Chains. They are involved in helpful results such as ergodic theorems.

Transition probabilities

Definition 2.1.2 (Transition matrix for a DTMC)
For a time-homogeneous DTMC $(S_n)_{n\in\mathbb{N}}$ *with state space* $\mathbf{S} = \{(\mathbf{s}^{(i)})_i\} \subseteq \mathbb{N}^d$*, the matrix*

$$P = (p_{ij})_{ij} = (\mathbb{P}(S_1 = \mathbf{s}^{(j)} | S_0 = \mathbf{s}^{(i)}))_{\mathbf{s}^{(i)},\mathbf{s}^{(j)}\in\mathbf{S}}$$

is called the transition matrix of the DTMC. $P(\mathbf{s}^{(i)}, \mathbf{s}^{(j)}) = p_{ij}$ *is called a transition probability.*

p_{ij} is the probability that, being at the state $\mathbf{s}^{(i)}$, the chain transitions to the state $\mathbf{s}^{(j)}$. One can generalise this definition with n-step transitions.

Definition 2.1.3 (n-step transition probability)
Let $(S_n)_{n\in\mathbb{N}}$ *be a time-homogeneous DTMC. For* $n \in \mathbb{N}^*$*, the probability*

$$p_{ij}^{(n)} = \mathbb{P}(S_n = \mathbf{s}^{(j)} | S_0 = \mathbf{s}^{(i)})$$

is called a n-step transition probability. We denote $P^{(n)} = (p_{ij}^{(n)})_{ij}$*.*

Any transition probability $p_{ij}^{(n+m)}$ can be computed with intermediate probabilities by Kolmogorov's equations.

Theorem 2.1.1 (Chapman-Kolmogorov equations)
Let $(S_n)_{n\in\mathbb{N}}$ *be a time-homogeneous DTMC with associated transition matrices* $(P^{(n)})_{n\in\mathbb{N}}$ *2.1.3. Then:*

$$\forall n, m \in \mathbb{N}, P^{(n+m)} = P^{(n)}P^{(m)}.$$

Ergodic chain A stationary distribution is a distribution invariant by the transition probability matrix (see Definition 2.1.10 further). The existence of such distributions is an important result of Markov Chains. Under adequate hypotheses, the matrix $P^{(n)}$ converges to the unique stationary distribution of the Markov Chain as n grows to infinity. These results are known as ergodic theorems. They are especially useful in advanced Bayesian statistical methods, as we will see in the next chapter. We proceed by describing these hypotheses related to the states of the chain. We say that a chain verifies a property if all its states verify this property.

Definition 2.1.4 (Initial passing time)
The random time of first entrance in a state $\mathbf{s}^{(i)}$*, also called initial passing time or first hit time, is defined as:*

$$\tau_{\mathbf{s}^{(i)}} = \inf\{t \in \mathbb{N}, t > 1 / S_t = \mathbf{s}^{(i)}\}$$

Definition 2.1.5 (Recurrent state)
A state $\mathbf{s}^{(i)}$ *is recurrent if* $\sum_{n=1}^{+\infty} p_{ii}^{(n)} = +\infty$*. Otherwise it is called transient.*

This means that as n tends to infinity, the state will be visited by the Markov Chain regularly. Indeed $\sum_{n=1}^{+\infty} p_{ii}^{(n)}$ is the expectation of the random variable $N_{\mathbf{s}^{(i)}}$ that counts the number of times the chain sojourns in $\mathbf{s}^{(i)}$ when starting from $\mathbf{s}^{(i)}$, i.e. $N_{\mathbf{s}^{(i)}} = \sum_{n=1}^{+\infty} \mathbb{1}(S_n = \mathbf{s}^{(i)})$.

Definition 2.1.6 (Positive recurrent state)
A recurrent state $\mathbf{s}^{(i)}$ *is positive recurrent if* $\mathbb{E}[\tau_{\mathbf{s}^{(i)}} |\ S_0 = \mathbf{s}^{(i)}] < +\infty$*. Otherwise it is called null recurrent.*

A Markov Chain is positive recurrent if all its states are positive recurrent.

Definition 2.1.7 (Communication of states - Irreducibility)
A state $\mathbf{s}^{(i)}$ *leads to* $\mathbf{s}^{(j)}$ *(*$\mathbf{s}^{(i)} \to \mathbf{s}^{(j)}$*) if* $\exists n \in \mathbb{N}, p_{ij}^{(n)} > 0$*. States* $\mathbf{s}^{(i)}$ *and* $\mathbf{s}^{(j)}$ *communicate (*$\mathbf{s}^{(i)} \leftrightarrow \mathbf{s}^{(j)}$*) if* $\mathbf{s}^{(i)} \to \mathbf{s}^{(j)}$ *and* $\mathbf{s}^{(j)} \to \mathbf{s}^{(i)}$*.*

A Markov Chain is irreducible if all its states communicate.

Definition 2.1.8 (Period of a state and aperiodicity)
The period of a state $\mathbf{s}^{(i)}$ *is defined as:*

$$gcd\{n \geq 1 / p_{ii}^{(n)} > 0\}$$

where gcd is the greatest common divisor. If the period is 1, $\mathbf{s}^{(i)}$ *is called aperiodic.*

A Markov Chain is aperiodic if all its states are aperiodic.

We obtain the notion of ergodicity by gathering all these properties.

Definition 2.1.9 (Ergodic Markov Chain)
A Markov Chain $(S_n)_{n \in \mathbb{N}}$ *is ergodic if it is positive reccurent, irreducible and aperiodic.*

Ergodic Markov Chains have useful properties, which are mainly embodied by the ergodic theorem.

Definition 2.1.10 (Stationary distribution)
A stationary distribution $(\pi_i)_{i\in\mathbb{N}/\mathbf{s}^{(i)}\in\mathbf{S}}$ *of a Markov Chain with transition matrix* P *is a non-zero vector where for all* $i \in \mathbb{N}$ *corresponding to a state* $\mathbf{s}^{(i)} \in \mathbf{S}$, $0 \leq \pi_i < 1$, $\pi = \pi P$ *and* $\sum_{i\in\mathbb{N}/\mathbf{s}^{(i)}\in\mathbf{S}} \pi_i = 1$.

Proposition 2.1.1 (Stationary distribution of a Markov Chain)
An ergodic Markov Chain admits a unique stationary distribution π.

In fact, it is sufficient for the Markov Chain to be irreducible and positive recurrent for Proposition 2.1.1 to be true. We have stated the required definitions and properties for the ergodic theorem.

Theorem 2.1.2 (Ergodic theorem)
Let $(S_n)_{n\in\mathbb{N}}$ *be an ergodic Markov Chain with n-step transition probabilities* $p_{ij}^{(n)}$ *and its stationary distribution* π. *Then the n-step transition probabilities converge to the stationary distribution, i.e*

$$\forall i, \sup_j |p_{ij}^{(n)} - \pi_j| \underset{n\to+\infty}{\to} 0.$$

This result is generalisable with continuous state spaces. They are especially useful in Markov Chain Monte Carlo statistical methods, as we will see in Chapter 3.

Continuous-Time Markov Chain

We now extend the definitions described in the previous section to the case of a continuous-time set $\mathbb{T} = \mathbb{R}_{\geq 0}$ (state space $\mathbf{S}$ is still discrete).

Definition 2.1.11 (Continuous-Time Markov Chain (CTMC))
$(S_t)_{t\in\mathbb{R}_{\geq 0}}$ *is a time-homogeneous CTMC if it satisfies the following properties:*

1. *The trajectories* $(t \to S_t(\omega))_{\omega\in\Omega}$ *are right-continuous*
2. $\forall t, s \in \mathbb{T}, \mathbb{P}(S_t|S_v, v \in [0, s]) = \mathbb{P}(S_t|S_s)$ *(Markov property)*
3. $\forall t, v \in \mathbb{R}_{\geq 0}, t > v, \mathbb{P}(S_t|S_v) = \mathbb{P}(S_{t-v}|S_0)$ *(Time-homogeneity).*

Remark 2.1.1 (Time-homogeneity of a Markov Chain)
In our applications, unless otherwise stated, Markov Chains will be considered time-homogeneous.

We define $T_{\mathbf{s}}$ ($\mathbf{s} \in \mathbf{S}$) the random sojourn time in state $\mathbf{s}$, i.e. the time before a state transition occurs while in $\mathbf{s}$:

$$\mathbb{P}(T_{\mathbf{s}} \geq t) = \mathbb{P}(\{\omega \in \Omega, \forall t' \in [0,t], S_{t'}(\omega) = \mathbf{s}\})$$

Proposition 2.1.2 (Exponential times of a CTMC.)
The random variable $T_{\mathbf{s}}$ verifies the memoryless property, i.e. it follows an exponential law.

As $T_{\mathbf{s}}$ follows an exponential law, we denote $E(\mathbf{s})$ its parameter, which is called *total exit rate*. Then, $T_{\mathbf{s}} \sim Exp(E(\mathbf{s}))$ and $\mathbb{E}(T_{\mathbf{s}}) = \frac{1}{E(\mathbf{s})}$. The event $\{S_{T_{\mathbf{s}}} = \mathbf{s}'\}$ is defined as the set of elementary events ω so that the trajectory $S(\omega)$ reaches $\mathbf{s}'$ after a jump from $\mathbf{s}$.

Definition 2.1.12 (Transition probability for a CTMC)
Let $(S_t)_{t \in \mathbb{R}_{\geq 0}}$ be a CTMC.

$$\forall \mathbf{s}, \mathbf{s}' \in \mathbf{S}, \mathbf{s} \neq \mathbf{s}', P(\mathbf{s}, \mathbf{s}') = \mathbb{P}(S_{T_{\mathbf{s}}} = \mathbf{s}' | S_0 = \mathbf{s})$$

is called a transition probability.

Then, we can prove the following properties based on the first-order Taylor's expansion of exponential and on the Markov property:

$$\begin{aligned} \forall t \in \mathbb{T}, \forall \mathbf{s}, \mathbf{s}' \in \mathbf{S}, \mathbf{s} \neq \mathbf{s}', &\mathbb{P}(S_{t+h} = \mathbf{s}' | S_t = \mathbf{s}) = P(\mathbf{s}, \mathbf{s}')E(\mathbf{s})h + o(h) \\ \forall t \in \mathbb{T}, \forall \mathbf{s} \in \mathbf{S}, &\mathbb{P}(S_{t+h} = \mathbf{s} | S_t = \mathbf{s}) = 1 - \sum_{\mathbf{s}' \neq \mathbf{s}} P(\mathbf{s}, \mathbf{s}')E(\mathbf{s})h + o(h). \end{aligned} \tag{2.1}$$

If we are in state $\mathbf{s}$ at t, the probability that the Markov process jumps to $\mathbf{s}'$ after a time h only depends on h and $\mathbf{s}'$, but does not depend on t. Let $\{S_{T_{\mathbf{s}_0}} = \mathbf{s}'\}$ the event where trajectories go to $\mathbf{s}'$ after a jump from $\mathbf{s}_0$. We obtain the following result:

Proposition 2.1.3
For $\mathbf{s} \in \mathbf{S}$ and $I \subseteq \mathbb{R}_{\geq 0}$, the events $\{S_{T_{\mathbf{s}}} = \mathbf{s}'\}$ and $\{T_{\mathbf{s}} \in I\}$ are independent.

This property allows to define the *transition rate* matrix (or infinitesimal generator matrix) Q as:

$$\begin{aligned} \forall \mathbf{s}, \mathbf{s}' \in \mathbf{S}, \mathbf{s} \neq \mathbf{s}', Q(\mathbf{s}, \mathbf{s}') &= P(\mathbf{s}, \mathbf{s}')E(\mathbf{s}) \\ \forall \mathbf{s} \in \mathbf{S}, Q(\mathbf{s}, \mathbf{s}) &= -\sum_{\mathbf{s}' \neq \mathbf{s}} P(\mathbf{s}, \mathbf{s}')E(\mathbf{s}) \end{aligned} \tag{2.2}$$

In fact, Property 2.1 is sufficient for the definition of a CTMC.

Proposition 2.1.4 (Infinitesimal definition of a time-homogeneous CTMC)
A stochastic process $(S_t)_{t \in \mathbb{R}_{\geq 0}}$ with state space $\mathbf{S} = \{(\mathbf{s}^{(i)})_i\} \subseteq \mathbb{N}^d$ is a time-homogenenous CTMC if and only if there exists a matrix $Q = (q_{ij})_{ij}$ called transition rate matrix so that:

$$\forall t \in \mathbb{R}_{\geq 0}, h > 0, \mathbf{s}^{(i)}, \mathbf{s}^{(j)} \in \mathbf{S}, \mathbb{P}(S_{t+h} = \mathbf{s}^{(j)} | S_t = \mathbf{s}^{(i)}) = \delta_{ij} + q_{ij}h + o(h).$$

Remark 2.1.2 (Description of a CTMC.)
A CTMC is fully defined by its initial distribution S_0 and transition rate matrix Q. $E(\mathbf{s}) = \sum_{\mathbf{s}' \neq \mathbf{s}} Q(\mathbf{s}, \mathbf{s}')$ is the total exit rate of the state $\mathbf{s}$.

The transition rate matrix Q is involved in the evolution of the chain's state over time. Let us define the transient probability vector.

Definition 2.1.13 (Transient state probability)
Considering the countable state space $\mathbf{S} = \{(\mathbf{s}^{(i)})_i\}$, we define the transient state probability vector of a CTMC as the vector $P^{(t)} = (P_i^{(t)})_i$ where

$$P_i^{(t)} = \mathbb{P}(S_t = \mathbf{s}^{(i)}).$$

The dynamics of the Continuous-Time Markov Chain is described by Kolmogorov's forward equations.

Proposition 2.1.5 (Kolmogorov's forward equations)
Let P be the transient state probability vector (Definition 2.1.13). Then P verifies the Kolmogorov forward equation:

$$\frac{d}{dt} P^{(t)} = P^{(t)} Q.$$

This equivalence will be proven later in the context of Chemical Reaction Networks (Section 2.3.3).

Remark 2.1.3
For a finite state space ($Card(\mathbf{S}) < \infty$), the solution of the Kolmogorov's forward

equations is

$$P^{(t)} = e^{Qt}.$$

The analytical form in 2.1.3 is convenient for describing the dynamics of a CTMC system with few states, but the more the number of states increases, the less the computation of the matrix exponential is feasible. This is known as *the state-space explosion problem*, which justifies the use of statistical and approximate methods for CTMC. The bypass of the state-space explosion problem is the driving force of our work.

2.1.2 CTMC as an oriented graph

In this thesis, we are interested in verification of CTMC, which aims at verifying if a set of logical properties are satisfied by the stochastic process. The common representation of CTMCs used in this field is based on the transition rate matrix (see Remark 2.1.2). They are represented by a tuple $\mathcal{M} = (\mathbf{S}, \alpha, Q)$ where:

- $\mathbf{S}$ is the set of states
- $\alpha : \mathbf{S} \to [0, 1]$ with $\sum_{\mathbf{s}\in\mathbf{S}} \alpha(\mathbf{s}) = 1$ is the initial distribution. It corresponds to the distribution of S_0, $\mathbb{P}_{S_0}$.
- $Q : \mathbf{S} \times \mathbf{S} \to \mathbb{R}$ is the transition rate matrix where $\forall \mathbf{s}, \sum_{\mathbf{s}'\in\mathbf{S}} Q(\mathbf{s}, \mathbf{s}') = 0$.

Remark 2.1.4
In our applications, unless otherwise stated, the initial distribution is combined with a fixed initial state $\mathbf{s}_0$*, which means* $\alpha = \delta_{\mathbf{s}_0}$*. This case leads to a tuple* $\mathcal{M} = (\mathbf{S}, \mathbf{s}_0, Q)$*.*

In this case, a CTMC is illustrated by an oriented graph whose nodes are states and edges/transitions represent the transition rate matrix Q.

Figure 2.1 depicts an example of a CTMC with two states ($\mathbf{S} = \{\text{Sleep, Awake}\}$). It describes the sleep cycle of a human. The transition rate matrix is defined by $Q(\text{Sleep, Awake}) = \frac{1}{8}$, $Q(\text{Awake, Sleep}) = \frac{1}{16}$. As each row of Q should sum to zero, the two others coefficients equal $Q(\text{Sleep, Sleep}) = -\frac{1}{8}$ and $Q(\text{Awake, Awake}) = -\frac{1}{16}$. In this CTMC, only one edge arises from the Sleep state. So the sojourn time of the Sleep state is $T_{\text{Sleep}} \sim Exp(Q(\text{Sleep, Awake}))$. With a time scale of hours, this modelisation considers that the average time of sleep is 8 hours.

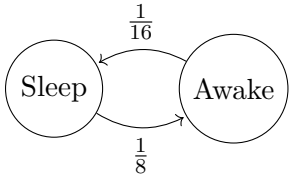


FIGURE 2.1: CTMC of a sleep cycle

Depending on the literature, the property $\sum_{\mathbf{s}' \in \mathbf{S}} Q(\mathbf{s}, \mathbf{s}') = 0$ can be either included in the definition or not. We assume this property to hold. Simply speaking, we do not allow self transitions ($\mathbf{s} \xrightarrow{t} \mathbf{s}$).

2.2 Probability measure of CTMCs.

In the following, we describe the set of trajectories of a CMTC and construct a probability measure over this set. It is a requirement for verification of CTMCs: the satisfaction of a formula is based on the computation of this probability measure. We need to ensure that this probability measure is well defined.

2.2.1 Paths/Trajectories of a CTMC.

Definition 2.2.1 (Path/Trajectory of a CTMC)
A path (or trajectory) σ of a CTMC $\mathcal{M} = (\mathbf{S}, \alpha, Q)$ is a (possibly infinite) sequence $\sigma = \mathbf{s}_0 \xrightarrow{t_0} \dots \xrightarrow{t_{k-1}} \mathbf{s}_k \dots$, with $t_i \in \mathbb{R}_{\geq 0}$ being the sojourn time in state $\mathbf{s}_i \in \mathbf{S}$ such that $\alpha(\mathbf{s}_0) > 0$ and $\forall i, Q(\mathbf{s}_i, \mathbf{s}_{i+1}) > 0$.

$Path(\mathcal{M})$ is the set of possible right-continuous paths of a CTMC $\mathcal{M}$.

Several operators are associated with a path $\sigma \in Path(\mathcal{M})$. For $i \in \mathbb{N}$ and $t \in \mathbb{R}_{\geq 0}$, $\sigma[i] = \mathbf{s}_i$ is the i-th state of σ, $\delta(\sigma, i) = t_i$ the sojourn time of σ in the i-th state, $\sigma@t$ the state of σ at time t and $\sigma[t\rangle$ the truncated path of σ at time t ($\sigma[t\rangle@0.2 = \sigma@(t + 0.2)$).

For example, if $\sigma = \mathbf{s}_0 \xrightarrow{0.25} \mathbf{s}_1 \xrightarrow{0.5} \mathbf{s}_2 \xrightarrow{0.15} \mathbf{s}_3 \xrightarrow{1} \dots$ we have $\sigma[1] = \mathbf{s}_1$, $\delta(\sigma, 2) = 0.15$, $\sigma@0.1 = \mathbf{s}_1$. A finite path can be considered as an infinite path with $t_k = +\infty$.

Remark 2.2.1 (About the notation of the set of paths)
In the literature, the initial distribution is sometimes not included in the definition of a CTMC. In this context, Path is used for the set of all paths with any possible initial state and $Path(\mathbf{s})$ for the paths originating in $\mathbf{s}$. In our work, we deal with collections

of CTMCs. Hence we do not want the reference of the CTMC to be implicit. As the initial distribution is included in the representation of Section 2.1.2, we do not mention the initial distribution in the notation. When necessary, an auxiliary CTMC with a different initial state but the same transition matrix will be defined (for example, in Definition 4.1.4 of the CSL satisfaction operator $\models$).

2.2.2 Probability measure over the set of paths.

Let $(S_t)_{t\in\mathbb{R}_{\geq 0}}$ a CTMC. We need a probability measure over the set of paths to quantify how probable a path/trajectory is from a CTMC. In statistical inference, such a probability measure is defined by the pushforward measure of $\mathbb{P}$ under S.

Let Φ_S be the function:

$$\begin{aligned}\Phi_S : (\Omega, \mathcal{F}) &\to (\mathbf{S}^{\mathbb{R}}_{\geq 0}, \mathcal{F}_{\mathbf{S}^{\mathbb{R}}_{\geq 0}}) \\ \omega &\to S(\omega)\end{aligned}$$

with $\mathbf{S}^{\mathbb{R}_{\geq 0}}$ the set of right-continuous functions $f : \mathbb{R}_{\geq 0} \to \mathbf{S}$, and $\mathcal{F}_{\mathbf{S}^{\mathbb{R}_{\geq 0}}}$ the σ-algebra generated by $\mathbf{S}^{\mathbb{R}}_{\geq 0}$. Then $S\mathbb{P} = \mathbb{P} \circ \Phi_S^{-1}$ defines a probability measure $\mathbb{P}_S$ on the measurable set $(\mathbf{S}^{\mathbb{R}}_{\geq 0}, \mathcal{F}_{\mathbf{S}^T})$. We have:

$$\mathbb{P}_S = S\mathbb{P} = \mathbb{P} \circ \Phi_S^{-1}$$

If S is a CTMC, for $\omega \in \Omega$, $S(\omega)$ is a trajectory (like σ). If A is a subset of trajectories, $\mathbb{P}_S(A) = \mathbb{P}(\{\omega \in \Omega, S(\omega) \in A\})$.

However, in the representation $\mathcal{M} = (\mathbf{S}, \alpha, Q)$ of a CTMC, the notion of probability does not appear clearly (except the distribution over the initial state). We only define a transition rate matrix Q, which defines the transitions between states and the timing information of a transition. Intuitively, the elements $Q(\mathbf{s}, \mathbf{s}')$ represent the velocity at which the transition $\mathbf{s} \to \mathbf{s}'$ occurs. For verification purpose of these systems, the probability measure over the set of trajectories is constructed explicitely by induction. It is based on the assumption that the time before an outgoing transition from a non-absorbing state has an exponential time.

In the following, we define a probability measure $Pr_{\mathcal{M}}$, based on the CTMC $\mathcal{M} = (\mathbf{S}, \alpha, Q)$, over the measurable space $(Path(\mathcal{M}), \mathcal{F}_{Path(\mathcal{M})})$, where $\mathcal{F}_{Path(\mathcal{M})}$ is the σ-algebra generated by $Path(\mathcal{M})$.

Let $\mathbf{s}_i, \mathbf{s}_j \in \mathbf{S}$ so that $\mathbf{s}_i \neq \mathbf{s}_j$ and $E(\mathbf{s}_i) > 0$. The probability that $\mathbf{s}_i \to \mathbf{s}_j$ occurs within $t \in \mathbb{T}$ is given by:

$$P(\mathbf{s}_i, \mathbf{s}_j, t) = P(\mathbf{s}_i, \mathbf{s}_j)(1 - e^{-E(\mathbf{s}_i)t}) \tag{2.3}$$

where $P(\mathbf{s}_i, \mathbf{s}_j) = \frac{Q(\mathbf{s}_i, \mathbf{s}_j)}{E(\mathbf{s}_i)}$ is called a *transition probability* and $(1 - e^{-E(\mathbf{s}_i)t})$ is the probability that the transition occurs within t.

We define the cylinder set $C(\mathbf{s}_0, I_0, \ldots, I_{k-1}, \mathbf{s}_k)$ with $I_j =]a_j, b_j[$ as:

$$C(\mathbf{s}_0, I_0, \ldots, I_{k-1}, \mathbf{s}_k) = \{\sigma \in Path(\mathcal{M}), \forall i \in \{0, \ldots, k\}, \sigma[i] = \mathbf{s}_i, \delta(\sigma, i) \in I_i\}$$

If $\sigma \in C(\mathbf{s}_0, I_0, \ldots, I_{k-1}, \mathbf{s}_k)$, the sequence of states $\mathbf{s}_0 \to \ldots \to \mathbf{s}_k$ that a trajectory traverses is fixed but the times spent in each state can vary. A way of visualising $C(\mathbf{s}_0, I_0, \ldots, I_{k-1}, \mathbf{s}_k)$ is $\mathbf{s}_0 \xrightarrow{I_0} \ldots \xrightarrow{I_{k-1}} \mathbf{s}_k$.

The probability measure $Pr_{\mathcal{M}}$ is defined by induction:

$$\begin{aligned} Pr_{\mathcal{M}}(C(\mathbf{s}_0)) =& \alpha(\mathbf{s}_0) = P(\mathbf{s}_0) \\ Pr_{\mathcal{M}}(C(\mathbf{s}_0, I_0, \ldots, I_{k-1}, \mathbf{s}_k)) =& Pr_{\mathcal{M}}(C(\mathbf{s}_0, I_0, \ldots, I_{k-2}, \mathbf{s}_{k-1})) \\ & \cdot P(\mathbf{s}_{k-1}, \mathbf{s}_k)(e^{-a_k E(\mathbf{s}_{k-1})} - e^{-b_k E(\mathbf{s}_{k-1})}) \end{aligned}$$

By denoting $T_{\mathbf{s}_k} \sim Exp(E(\mathbf{s}_k))$,

$$Pr_{\mathcal{M}}(C(\mathbf{s}_0, I_0, \ldots, I_{k-1}, \mathbf{s}_k)) = Pr_{\mathcal{M}}(C(\mathbf{s}_0)) \cdot \prod_{i=1}^{k} P(\mathbf{s}_i, \mathbf{s}_{i+1}) \mathbb{P}(T_{\mathbf{s}_i} \in I_i)$$

The probability measure is thus defined over a subset of $\mathcal{F}_{Path(\mathcal{M})}$. It is sufficient to define the measure over these cylinder sets: the measure is naturally extended over the whole space (see Appendix B for details).

Proposition 2.2.1 (Consistency of the probability measures)

$$\forall A \in \mathcal{F}_{Path(\mathcal{M})}, Pr_{\mathcal{M}}(A) = \mathbb{P}_S(A)$$

Proof. It is sufficient to prove the equality of the measures over any cylinder set $C(\mathbf{s}_0, I_0, \dots, I_{k-1}, \mathbf{s}_k)$. This is a consequence of Caratheodory's extension theorem (Klenke, 2008, Chapter 1). We refer the readers interested in the probabilistic details of measure constructions to the Appendix B.

Let $A = C(\mathbf{s}_0, I_0, \dots, I_{k-1}, \mathbf{s}_k)$.

Let us prove this by induction over $p \in \{0, \dots, k\}$. Let $p \geq 1, A_p = C(\mathbf{s}_0, I_0, \dots, I_{p-1}, \mathbf{s}_p)$. Then $A_k = A$.

We set these events:

- $E_0 = \{S_0 = \mathbf{s}_0\}$
- $E_1 = \{T_{\mathbf{s}_0} \in I_0, S_{T_{\mathbf{s}_0}} = \mathbf{s}_1\}$
- $j \geq 2, E_j = \{a_{j-1} + \sum_{i=0}^{j-2} T_{\mathbf{s}_i} \leq T_{\mathbf{s}_{j-1}} + \sum_{i=0}^{j-2} T_{\mathbf{s}_i} \leq b_{j-1} + \sum_{i=0}^{j-2} T_{\mathbf{s}_i}, S_{\sum_{i=0}^{j-1} T_{\mathbf{s}_i}} = \mathbf{s}_j\} = \{a_{j-1} \leq T_{\mathbf{s}_{j-1}} \leq b_{j-1}, S_{\sum_{i=0}^{j-1} T_{\mathbf{s}_i}} = \mathbf{s}_j\}$

E_j is the event that after spending a time $T_{\mathbf{s}_{j-1}} \in I_{j-1}$ in $\mathbf{s}_{j-1}$, we have a jump from s_{j-1} to s_j.

By time-homogeneity of the Markov Chain, $\mathbb{P}_S(A_p) = \mathbb{P}(E_0, \dots, E_p)$.

Let us reason by finite induction on $p \in \{1, \dots, k\}$.

$p = 1$:

$$\begin{aligned}
\mathbb{P}(A_1) &= \mathbb{P}(E_1|E_0)\mathbb{P}(E_0) \\
&= \mathbb{P}(T_{\mathbf{s}_0} \in I_0, S_{T_{\mathbf{s}_0}} = \mathbf{s}_1|S_0 = \mathbf{s}_0)\mathbb{P}(S_0 = \mathbf{s}_0) \\
&= \mathbb{P}(T_{\mathbf{s}_0} \in I_0)\mathbb{P}(S_{T_{\mathbf{s}_0}} = \mathbf{s}_1|S_0 = \mathbf{s}_0) \\
&= P(\mathbf{s}_0, \mathbf{s}_1)\mathbb{P}(T_{\mathbf{s}_0} \in I_0)
\end{aligned}$$

$p + 1$:

$$\begin{aligned}
\mathbb{P}_S(A_{p+1}) &= \mathbb{P}(E_0, \dots, E_{p+1}) \\
&= \mathbb{P}(E_{p+1}|E_0, \dots, E_p)\mathbb{P}(E_0, \dots, E_p) \\
&= \mathbb{P}(E_{p+1}|E_0, \dots, E_p) \prod_{i=1}^{p} P(\mathbf{s}_i, \mathbf{s}_{i+1})\mathbb{P}(T_i \in I_i)
\end{aligned}$$

Then,

$$\begin{aligned}\mathbb{P}(E_{p+1}|E_0,\dots,E_p) &= \mathbb{P}(T_{\mathbf{s}_p} \in I_p, S_{\sum_{i=0}^{p} T_{\mathbf{s}_i}} = \mathbf{s}_{p+1}|E_0,\dots,E_p) \\ &= \mathbb{P}(T_{\mathbf{s}_p} \in I_p, S_{\sum_{i=0}^{p} T_{\mathbf{s}_i}} = \mathbf{s}_{p+1}|S_0 = \mathbf{s}_0, T_{\mathbf{s}_0} \in I_0,\dots,T_{\mathbf{s}_{p-1}} \in I_{p-1}, S_{\sum_{i=0}^{p-1} T_{\mathbf{s}_i}} = \mathbf{s}_p)\end{aligned}$$

Since the value of the next state only depends on the previous value, we have:

$$\mathbb{P}(E_{p+1}|E_0,\dots,E_p) = \mathbb{P}(T_{\mathbf{s}_p} \in I_p, S_{\sum_{i=0}^{p} T_{\mathbf{s}_i}} = \mathbf{s}_{p+1}|S_{\sum_{i=0}^{p-1} T_{\mathbf{s}_i}} = \mathbf{s}_p)$$

Hence, as the CTMC is time-homogeneous, we translate the time by $-\sum_{i=0}^{p-1} T_{\mathbf{s}_i}$, and we obtain by the independence of $\{T_{\mathbf{s}_0} \in I\}$ and $\{S_{T_{\mathbf{s}_0}} = \mathbf{s}'\}$ (Proposition 2.1.3):

$$\mathbb{P}(E_{p+1}|E_0,\dots,E_p) = P(\mathbf{s}_p,\mathbf{s}_{p+1})\mathbb{P}(T_{\mathbf{s}_p} \in I_p)$$

By induction the result is proven. □

Remark 2.2.2 ($Pr_{\mathcal{M}}$ or $\mathbb{P}_S$?)
If these notations appoints the same probability measure, should we drop one of them? In this thesis, we think both notations are useful. In model checking literature, the representation of Section 2.1.2 and the construction of $Pr_{\mathcal{M}}$ are frequently presented (e.g. (Aziz, 2000) or (Baier et al., 2003)). For a CTMC $\mathcal{M} = (\mathbf{S}, \alpha, Q)$, $Pr_{\mathcal{M}}$ is often denoted Pr_α (the transition rate matrix is implicit due to the fact we only consider one instance of CTMC in classical model checking).

However, it is convenient to use $\mathbb{P}_S$ in a statistical context. Also, $Pr_{\mathcal{M}}$ does not allow probability calculus with random variables, which deprives us of a useful tool. For example, if we denote A the set of trajectories so that the value at time t_1 is above x_1, one can rewrite $Pr_{\mathcal{M}}(A)$ as $\mathbb{P}_S(A) = \mathbb{P}(S_{t_1} \geq x_1)$.

2.3 Chemical Reaction Networks

In this section, we describe a specific formalism called Chemical Reaction Networks. They are used to represent a class of biological models that describe the kinetics of chemical phenomena.

In this kind of models, we consider an amount of different species molecules in a specific constant volume: these molecules can collide and provoke transformations. These phenomena are expressed by reactions:

$$R_i : \underbrace{a_{i1}X_1 + \ldots + a_{id}X_d}_{reactants} \rightarrow \underbrace{b_{i1}X_1 + \ldots b_{id}X_d}_{products}$$

Due to the complexity of the interactions between the species, the modelling of Chemical Reaction Networks is stochastic. Under particular assumptions, the underlying probabilistic model is a Continuous-Time Markov Chain. This stochastic description is especially useful when the numbers of molecules are not significant. Most models consider molecules that can collide and react, such as gene regulatory networks (Karlebach and Shamir, 2008) or cell signaling pathways (Eungdamrong and Iyengar, 2004). But this framework is also helpful in the study of interactions between humans or animals such as the SIR (Kermack and McKendrick, 1927) or Lokta-Volterra (Lotka, 1932) models.

2.3.1 Definition of a Chemical Reaction Network

In the most detailed description of chemical phenomena, one should describe the position and velocity of each molecule in time and fires a reaction when two molecules collide. Stochastic semantics of Chemical Reaction Network are based on two assumptions:

- The system is well-stirred in a constant volume. At any time t, the system evolution is only represented by a state $\mathbf{s} \in \mathbb{N}^d$ where each component $\mathbf{s}[i]$ is the number of species. Thus, the system evolution can be described by a stochastic process $(S_t)_{t\in\mathbb{R}_{\geq 0}}$.
- Each possible change of the population occurs through a reaction channel R_j. It is defined by two quantities: ν_j the stoichiometric vector and η_j the kinetic rate.

The stoichiometric vector ν_j represents the variation in number for each species when R_j occurs. The kinetic rate $\eta_j = \eta_j(\mathbf{s}, \theta)$ of a reaction channel R_j depends both on the state $\mathbf{s}$ and some parameters θ. This quantity is defined so that given a state $\mathbf{s}$, the probability that the reaction j occurs within the infinitesimal interval $[t, dt[$ is $\eta_j(\mathbf{s}, \theta)dt$. In other words,

$$\mathbb{P}(R_j \text{ occurs within } [t, t+dt] | \text{ population at time t is } \mathbf{s}) \approx \eta_j(\mathbf{s}, \theta)dt \tag{2.4}$$

The function $\mathbf{s} \rightarrow \eta_j(\mathbf{s}, \theta)dt$ is called the *propensity function* (also called the hazard function).

Definition 2.3.1 (Chemical Reaction Network)
A Chemical Reaction Network is fully described by:

- *The set of species $\mathcal{S}$ of cardinal d.*
- *A set $R_1, \ldots, R_m$ of reaction channels where each R_j is characterised by a pair $R_j : (\nu_j, \eta_j)$ with $\boldsymbol{\nu_j} = [\nu_{1j}, \ldots, \nu_{dj}]$ the stoichiometric vector, representing the variation in number for each species determined by the occurrence of R_j, and $\eta_j = \eta_j(\mathbf{s}, \theta)$ the kinetic rate of the reaction R_j.*
- *A p-dimensional vector of parameters $\theta = [\theta_1, \ldots, \theta_p]$ affecting the kinetic rate of the reaction channels, with θ in $\Theta \subset \mathbb{R}^p$.*

The Equation 2.4 recalls the characterisation of a CTMC in Proposition 2.1.4. Indeed, the assumptions on the Chemical Reaction Network system evolution leads to a Continuous-Time Markov Chain.

Remark 2.3.1 (CTMC description of a Chemical Reaction Network)
The system evolution of a CRN is described by a CTMC $(S_t)_{t \in \mathbb{R}_{\geq 0}}$:

- *A discrete state space $\mathbf{S} \subseteq \mathbb{N}^d$ whose elements are vectors $\mathbf{s} = [\mathbf{s}[1], \ldots, \mathbf{s}[d]] \in \mathbf{S}$ where $\mathbf{s}[i]$ is the population, in terms of number of molecules of the i-th species.*
- *An initial state $\mathbf{s}_0$.*
- *A transition rate matrix Q_θ defined by $\forall j \in \{1, \ldots, M\}, Q_\theta(\mathbf{s}, \mathbf{s} + \nu_j) = \eta_j(\mathbf{s}, \theta)$, and 0 for the states that does not have the form $\mathbf{s} + \nu_j$.*

We deduce the transitition probabilities and total exit rates:

- $P_\theta(\mathbf{s}_t, \mathbf{s}_t + \nu_j) = \frac{\eta_j(\mathbf{s}_t, \theta)}{\eta_{sum}(\mathbf{s}_t, \theta)}$
- $E_\theta(\mathbf{s}) = \eta_{sum}(\mathbf{s}_t, \theta) = \sum_{j=1}^{M} \eta_j(\mathbf{s}_t, \theta)$.

A CTMC whose a state represents the number of individuals of several categories/species is also often called a Markov Population Process.

A common assumption about the kinetic rate is the *law of mass action*. This law states that the kinetic rate is proportional to the multiplication of the numbers of reactant species.

Remark 2.3.2
Unless stated, the considered Chemical Reaction Networks follows the mass-action law.

2.3.2 Example

Let us illustrate the definitions with an example. Let $\mathcal{S} = \{A, B, C, D\}$ be the set of species, $\mathbb{N}^4$ be the state space. The i-th component $i \in \{1, \dots, 4\}$ of a state is the number of molecules of the species i. We consider the following set of reactions with constant reaction rates $\theta = (k_1, k_2, k_3)$:

$$\begin{aligned} R_1&: \quad A + C \xrightarrow{k_1} D \quad (\textit{second order reaction}) \\ R_2&: \quad A \xrightarrow{k_2} B \qquad\quad (\textit{first order reaction}) \\ R_3&: \quad \emptyset \xrightarrow{k_3} C \qquad\quad (\textit{zero order reaction}) \end{aligned}$$

Then the associated stoichiometric vectors are:

$$\begin{aligned} \nu_1 &= [\ \underbrace{-1}_{\substack{\text{A is a reactant} \\ \text{in } R_1}}, 0, -1, \underbrace{1}_{\substack{\text{D is a product} \\ \text{in } R_1}}\ ] \\ \nu_2 &= [-1, 1, 0, 0] \\ \nu_3 &= [0, 0, 1, 0]. \end{aligned}$$

The mass-action law leads to the kinetic rates:

$$\begin{aligned} \eta_1(\mathbf{s}, \theta) &= k_1 \cdot |A| \cdot |C| = k_1 \cdot \mathbf{s}[1] \cdot \mathbf{s}[3] \\ \eta_2(\mathbf{s}, \theta) &= k_2 \cdot |A| = k_2 \cdot \mathbf{s}[1] \\ \eta_3(\mathbf{s}, \theta) &= k_3. \end{aligned}$$

2.3.3 Different representations of the system evolution

As we saw in the previous section, the formalism of Chemical Reaction Networks leads to a CTMC $(S_t)_{t \in \mathbb{R}_{\geq 0}}$ to describe the system evolution. In the following, we describe the equations that govern $(S_t)_{t \in \mathbb{R}_{\geq 0}}$ and how, with additional assumptions, approximated equations can be deduced.

Chemical Master Equation

The Chemical Master Equation of a Chemical Reaction Network, also known as the Kolmogorov equation in probability theory, is the set of equations that describes the probability of each state at any time t.

Let $\mathbf{s} \in \mathbf{S}$ a state. We want to describe the evolution of the probability to be in this state over time.

Let the events $E_I^{(0)}$ = "no reaction occurs in the interval I", $\forall j \in \{1, \dots, M\}, E_I^{(j)}$ = "the reaction j occurs in the interval time I". Let $t > 0$ be a time and $dt > 0$ be an infinitesimal time. Then,

$$\mathbb{P}(S_{t+dt} = \mathbf{s}) = \underbrace{\mathbb{P}(E^{(0)}_{[t,t+dt]} |\ S_t = \mathbf{s})}_{\text{no reaction occurs}} \mathbb{P}(S_t = \mathbf{s}) + \underbrace{\sum_{j=1}^{M} \mathbb{P}(E^{(j)}_{[t,t+dt]} |\ S_t = \mathbf{s} - \nu_j)}_{\text{one of the reactions occurs}} \mathbb{P}(S_t = \mathbf{s} - \nu_j)$$

Intuitively, this equation states that the probability of being in state $\mathbf{s}$ after an infinitesimal time dt can be decomposed into the sum of two probabilities:

- The probability of being in state $\mathbf{s}$ at t and no reaction occurs until $t + dt$.
- The probability of being in another state so that one of the reactions occurs within dt and brings the system in the state $\mathbf{s}$.

Based on the CTMC framework of a CRN 2.3.1,

$$\mathbb{P}(E^{(0)}_{[t,t+dt]} |\ S_t = \mathbf{s}) = (1 - \sum_{j=1}^{M} \eta_j(\mathbf{s}, \theta) dt)$$
$$\forall j \in \{1, \dots, M\}, \mathbb{P}(E^{(j)}_{[t,t+dt]} |\ S_t = \mathbf{s} - \nu_j) = \eta_j(\mathbf{s} - \nu_j, \theta) dt$$

Combining the Equation 2.3.3 and $\lim_{dt \to 0} \frac{\mathbb{P}(S_{t+dt} = \mathbf{s}) - \mathbb{P}(S_t = \mathbf{s})}{dt}$, we obtain the *Chemical Master Equation*:

$$\frac{d}{dt}\mathbb{P}(S_t = \mathbf{s}) = \sum_{j=1}^{M} (\eta_j(\mathbf{s} - \nu_j, \theta)\mathbb{P}(S_t = \mathbf{s} - \nu_j) - \eta_j(\mathbf{s}, \theta)\mathbb{P}(S_t = \mathbf{s})) \tag{2.5}$$

Random Time Change Representation

In the previous section, we focused on an infinitesimal time to derive an exact equation that describes the evolution in time of our system. Another option is to focus on the total amount of reactions that occur within time t. For $j \in \{1, \dots, M\}$, $N_j(S_0, t)$ is the random variable that gives the number of occurrences of the j-th reaction until time t. Then,

$$S_t = \sum_{j=1}^{M} N_j(S_0, t)\nu_j + S_0.$$

(Kurtz, 1980) proves:

$$S_t = \sum_{j=1}^{M} \mathcal{P}(\int_0^t \eta_j(S_v, \theta)dv)\nu_j + S_0. \tag{2.6}$$

where $\mathcal{P}(a)$ is a Poisson law with mean a. This is the *Random Time Change Representation* of the CTMC.

Tau-leap approximation and Chemical Langevin Equation

Equations 2.5 and 2.6 are in practice difficult to solve. An efficient way to obtain approximate equations that leads to faster simulation algorithms is to discretise the time interval by a step τ. Equation 2.6 becomes:

$$S_{t+\tau} = \sum_{j=1}^{M} \mathcal{P}(\int_t^{t+\tau} \eta_j(S_v, \theta)dv)\nu_j + S_t.$$

This equation is still exact. Two assumptions (Gillespie, 2007) for the interval time τ can lead to good approximations of our equations:

- For any time t and given the state $\mathbf{s}_t$, the propensity function does not vary a lot over $[t, t+\tau]$, i.e $\eta_j(S_{t'}, \theta) \approx \eta(\mathbf{s}_t, \theta)$ over $t' \in [t, t+\tau]$. (*Leap condition*).
- There is a significant amount of reactions that occur within $[t, t+\tau]$.

These assumptions seem contradictory. Indeed, they assume that τ is small enough so that the propensity function does not vary a lot, but it should be long enough to allow several reactions. In practice, these assumptions are valid if we have enough molecules of each species in the studied system.

The first assumption means that if we know in which state the system is at time t, then the propensity functions are constant over a time interval of length τ. We can then estimate by Riemann sum the integral involved in Equation 2.6 for any $N \in \mathbb{N}$:

$$\int_0^{N\cdot\tau} \eta_j(S_v,\theta)dv \approx \sum_{i=0}^{N-1} \eta_j(S_{i\tau},\theta)\tau$$

which gives the scheme

$$S_{t+\tau} = \sum_{j=1}^{M} \mathcal{P}(\eta_j(S_t,\theta)\tau)\nu_j + S_t. \tag{2.7}$$

Given the state $\mathbf{s}_t$, the expected number of reactions that will occur is $\eta_j(\mathbf{s}_t,\theta)\tau$.

The second assumption leads to an approximation of a Poisson distribution by a Normal distribution (according to the central limit theorem and the fact that a Poisson distribution is a sum of unit Poisson distributions).

$$\mathcal{P}(\eta_j(\mathbf{s}_t,\theta)\tau) \approx \mathcal{N}(\eta_j(\mathbf{s}_t,\theta)\tau, \eta_j(\mathbf{s}_t,\theta)\tau) = \eta_j(\mathbf{s}_t,\theta)\tau + \sqrt{\eta_j(\mathbf{s}_t,\theta)\tau}\mathcal{N}(0,1)$$

This gives the scheme:

$$S_{t+\tau} = \tau\sum_{j=1}^{M} \eta_j(\mathbf{s}_t,\theta)\nu_j + \sum_{j=1}^{M} \nu_j\sqrt{\eta_j(\mathbf{s}_t,\theta)\tau}\mathcal{N}(0,1) + S_t \tag{2.8}$$

which can be seen as a discretisation of the continuous stochastic differential equation:

$$dS_t = \sum_{j=1}^{M} \eta_j(S_t,\theta)\nu_j dt + \sum_{j=1}^{M} \nu_j\sqrt{\eta_j(\mathbf{s}_t,\theta)}dW_t^{(j)} \tag{2.9}$$

where $W_t^{(j)}$ are independent Brownian motions. The Equation 2.9 is called the *Chemical Langevin Equation* (Gillespie, 2000).

Reaction Rate Equation: a macroscopic deterministic approximation of a CRN

Initially, chemical kinetics were described by a set of ODE. The implicit assumption is that the system evolves in a chemical equilibrum. For example, in (Kermack and McKendrick, 1927) the system evolution of the SIR model is described by three ordinary differential equations (ODE). However, a macroscopic and deterministic equation can be derived from the CME 2.5 that corresponds to the classical ODE

description of chemical systems. We have for any $\mathbf{s}^{(i)} \in \mathbf{S} = \{(\mathbf{s}^{(i)})_{i\in\mathbb{N}}\}$

$$\frac{d}{dt}\mathbb{P}(S_t = \mathbf{s}^{(i)}) = \sum_{j=1}^{M}(\eta_j(\mathbf{s}^{(i)} - \nu_j, \theta)\mathbb{P}(S_t = \mathbf{s}^{(i)} - \nu_j) - \eta_j(\mathbf{s}^{(i)}, \theta)\mathbb{P}(S_t = \mathbf{s}^{(i)})). \quad (2.10)$$

As S_t is a discrete random variable, $\mathbb{E}[S_t] = \sum_i \mathbf{s}^{(i)}\mathbb{P}(S_t = \mathbf{s}^{(i)})$. Then by multiplying the previous equation by $\mathbf{s}^{(i)}$ and suming over $i \in \mathbb{N}$

$$\frac{d}{dt}\mathbb{E}[S_t = \mathbf{s}^{(i)}] = \sum_{j=1}^{M}\left(\sum_{i\in\mathbb{N}}\eta_j(\mathbf{s}^{(i)} - \nu_j, \theta)\mathbf{s}^{(i)}\mathbb{P}(S_t = \mathbf{s}^{(i)} - \nu_j) - \sum_{i\in\mathbb{N}}\eta_j(\mathbf{s}^{(i)}, \theta)\mathbf{s}^{(i)}\mathbb{P}(S_t = \mathbf{s}^{(i)})\right). \quad (2.11)$$

Then, as $\sum_{i\in\mathbb{N}}\eta_j(\mathbf{s}^{(i)}-\nu_j, \theta)\mathbf{s}^{(i)}\mathbb{P}(S_t = \mathbf{s}^{(i)}-\nu_j) = \sum_{i\in\mathbb{N}}\eta_j(\mathbf{s}^{(i)}, \theta)(\mathbf{s}^{(i)}+\nu_j)\mathbb{P}(S_t = \mathbf{s}^{(i)})$, it gives:

$$\frac{d}{dt}\mathbb{E}[S_t] = \sum_{j=1}^{M}\mathbb{E}[\eta_j(S_t, \theta)]\nu_j \quad (2.12)$$

which gives if we neglect the stochastic variations of our system ($t \to S_t$ is a deterministic function):

$$\frac{dS_t}{dt} = \sum_{j=1}^{M}\eta_j(S_t, \theta)\nu_j. \quad (2.13)$$

.

Equation 2.13 is called the *Reaction Rate Equation* (RRE). This equation is considered a good approximation of the system's dynamics if the population is very large (Gillespie, 2000).

Remark 2.3.3 (RRE with the Chemical Langevin Equation)
If we assume that the stochastic fluctuations of $(S_t)_t$ are negligible, i.e. it is a constant process, then by reconsidering the Chemical Langevin Equation 2.9

$$dS_t = \underbrace{\sum_{j=1}^{M}\eta_j(S_t, \theta)\nu_j\, dt}_{deterministic\ part} + \underbrace{\sum_{j=1}^{M}\nu_j\sqrt{\eta_j(\mathbf{s}_t, \theta)}dW_t^{(j)}}_{stochastic\ part}. \quad (2.14)$$

We obtain the Reaction Rate Equation 2.13 by neglecting the stochastic part.

2.3.4 Stochastic simulation of a CRN

We have derived useful exact or approximate equations for the description of the CRN dynamics. Thus, one can derive simulation algorithms of paths from the CRN based on these equations.

Stochastic simulation algorithm

Stochastic Simulation Algorithm (SSA) was first formulated in (Gillespie, 1977). It is an exact method that simulates statistically correct paths of a CRN.

In Section 2.1.1, we described the distribution of the sojourn time of a state $\mathbf{s}_t$, which is Exponential, and the distribution of transition from one state to another. These events are independent (Proposition 2.1.3). The probability density function of the couple $(S_{T_{\mathbf{s}_t}}, T_{\mathbf{s}_t})$ is:

$$p_{(S_{T_{\mathbf{s}_t}},T_{\mathbf{s}_t})}(\mathbf{s}_t+\nu_j,\tau|S_t=\mathbf{s}_t) = \underbrace{\frac{\eta_j(\mathbf{s}_t,\theta)}{\eta_{sum}(\mathbf{s}_t,\theta)}}_{\text{Categorical distribution}} \underbrace{\eta_{sum}(\mathbf{s}_t,\theta)e^{-\eta_{sum}(\mathbf{s}_t,\theta)\tau}}_{\text{Exponential distribution}}$$

where

$$\eta_{sum}(\mathbf{s}_t,\theta) = \sum_{j=1}^{M} \eta_j(\mathbf{s}_t,\theta) \text{ (total exit rate of } \mathbf{s}_t).$$

Equation 2.3.4 means that given the state $\mathbf{s}_t$ at a time t, the time before a reaction occurs and the next reaction that will occur are independent, so we can simulate them step by step, which provides the Stochastic Simulation Algorithm 1.

Algorithm 1 Exact Stochastic Simulation Algorithm

Require: A CRN, an end time T, an initial state $\mathbf{s}_0$.

Ensure: Two collections $(\mathbf{s}_i)_i$ and $(t_i)_i$ that represent the simulated path.

$t, t_0 \leftarrow 0$

$i \leftarrow 1$

while t < T **do**

$\tau \sim Exp(\eta_{sum}(\mathbf{s}_t, \theta))$

Take j in $\{1, \dots, M\}$ with probability $(\frac{\eta_j(\mathbf{s}_t,\theta)}{\eta_{sum}(\mathbf{s}_t,\theta)})_{j\in\{1,\dots,M\}}$

$\mathbf{s}_i \leftarrow \mathbf{s}_{i-1} + \nu_j$

$i \leftarrow i + 1$

$t, t_i \leftarrow t + \tau$

end while

The Modified Next Reaction Method is another exact stochastic simulation method developed by (Anderson, 2007) based on (Gibson and Bruck, 2000). This method needs drawing only one random number per simulated reaction instead of two in the SSA 1. The simulation is based on the representation 2.6. $T_j(t) = \int_0^t \eta_j(S_v, \theta)dv$ is called the internal time and depends on the propensity function. Algorithm 2 is based on the sequential computation of the internal time, see (Anderson, 2007) for more details.

Algorithm 2 Modified Next Reaction Method

Require: A CRN, an end time T, an initial state $\mathbf{s}_0$.

Ensure: Two collections $(\mathbf{s}_i)_i$ and $(t_i)_i$ that represent the simulated path.

$t \leftarrow 0$

For $j \in \{1, \dots, M\}, T_j \leftarrow 0$

$i \leftarrow 1$

while t < T **do**

For $j \in \{1, \dots, M\}, P_j \sim Exp(1)$

For $j \in \{1, \dots, M\}, \Delta t_j \leftarrow \frac{P_j - T_j}{\eta_j(\mathbf{s}_{i-1},\theta)}$

$j* \leftarrow \underset{\{1,\dots,M\}}{\arg\min} \Delta t_j$

$\mathbf{s}_i \leftarrow \mathbf{s}_{i-1} + \nu_{j*}$

$t \leftarrow t + \Delta t_{j*}$

$T_j \leftarrow T_j + \eta_j(\mathbf{s}_{i-1}, \theta)\Delta t_{j*}$

$s \sim Exp(1), P_{j*} \leftarrow P_{j*} + s$

$i \leftarrow i + 1$

end while

Tau-leap approximation

Exact simulation algorithms can be computationally expensive. The greater the end time T, the greater the number of occurred reactions. The idea of the explicit Tau-leaping simulation algorithm is to simulate a bunch of reactions with Poisson distributions at each time step. Indeed, with the tau-leap assumptions, the scheme in Equation 2.7 gives a sequential way to simulate the number of occurrences of each reaction with Poisson distributions during a time τ.

Algorithm 3 Explicit Tau-leaping Simulation Algorithm

Require: A CRN, a number of time steps N, an initial state $\mathbf{s}_0$.

Ensure: Two collections $(\mathbf{s}_i)_i$ and $(t_i)_i$ that represent the simulated path.

$i \leftarrow 1$

for i = 1:N **do**

$\forall j \in \{1, \dots, M\}, k_j \sim \mathcal{P}(\eta_j(\mathbf{s}_{(i-1)\tau}, \theta)\tau)$

$\mathbf{s}_{i\cdot\tau} \leftarrow \sum_{j=1}^{M} k_j \nu_j + \mathbf{s}_{(i-1)\tau}$

$i \leftarrow i + 1$

end for

More advanced tau-leaping methods with adaptive τ or implicit schemes can be found (Rathinam et al., 2003; Cao, Gillespie, and Petzold, 2005; Cao, Gillespie, and Petzold, 2006).

Example of simulations with the SIR model

To apply the concepts and algorithms we have described earlier in the chapter, we consider the classical SIR compartmental model (Kermack and McKendrick, 1927). The SIR model describes the spread of an infectious disease among a population of constant size. The state vector is $(X_S, X_I, X_R) \in \mathbb{N}^3$. Species S represents the susceptible individuals, I the infected and R the recovered ones. The system's dynamics is described by the Chemical Reaction Network 2.15. Reaction R_1 is the infection process: a susceptible person meets an infected person and gets infected. Reaction R_2 is the recovering process: the infected may become immune to the disease. The parameter vector of the model is $\theta = (k_i, k_r)$.

$$\begin{aligned} R_1 &: S + I \xrightarrow{k_i} 2I \\ R_2 &: I \xrightarrow{k_r} R \end{aligned} \tag{2.15}$$

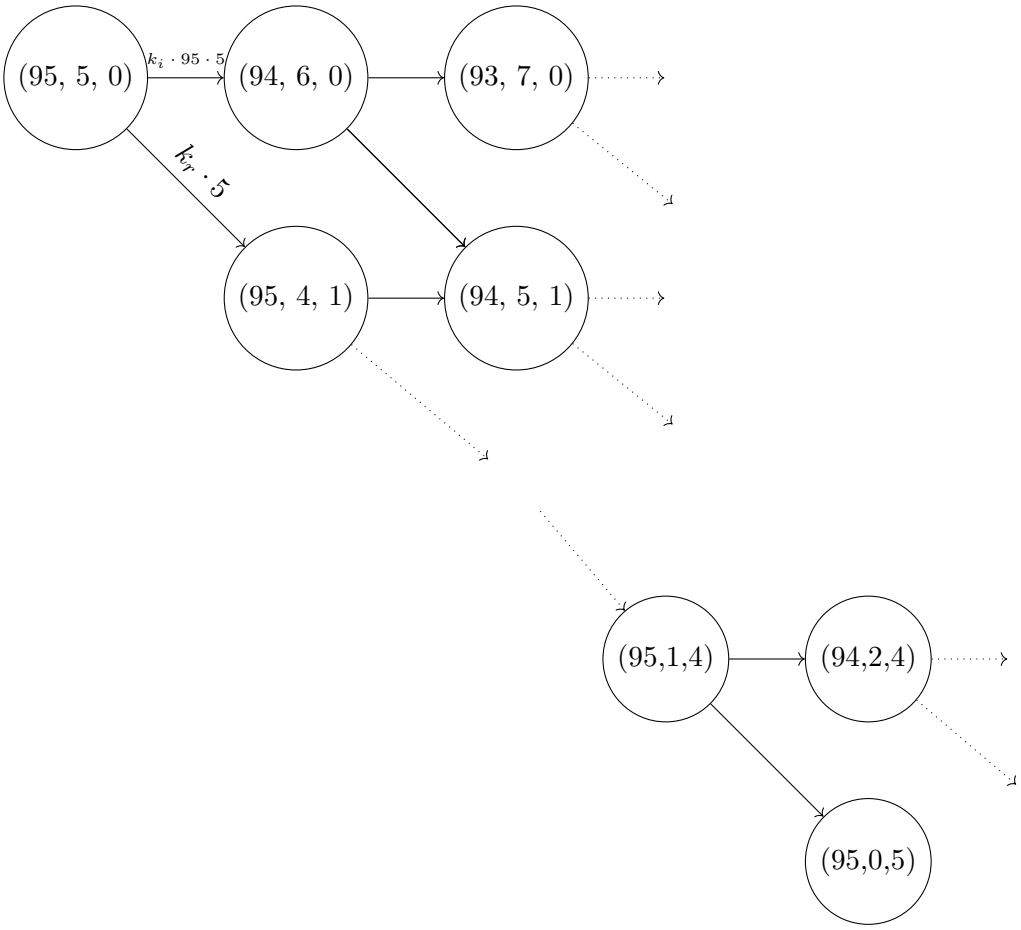


FIGURE 2.2: Graph of the SIR CTMC with the initial state $\mathbf{s}_0 = (95, 5, 0)$.

This chemical system is illustrated by a finite-state CTMC graph with an initial state $\mathbf{s}_0 = (95, 5, 0)$ in Figure 2.2.

We simulate different paths of the SIR system with $\theta = (0.12/N_{pop}, 0.05)$ with SSA and Tau-leaping simulations, and we plot the solution of the reaction rate equation for the infected people.

Figure 2.3 shows the different levels of approximation of the Reaction Rate Equation according to population sizes. On the left, the population is far smaller than on the right. We can see the simulated paths diverges more from the ODE solution than on the right plot. On the right plot, the ODE solution seems a much better approximation.

Figure 2.4 shows the different levels of approximations of the stochastic simulations according to the time step τ. On the right, τ is much higher, and the epidemic's peak is shifted to the right. Possibly a lot of reactions occurs between each time step, so the kinetic rate varies within a time step (violation of one of the tau-leap assumptions). Also, the tau leap paths diverge more from the SSA paths.

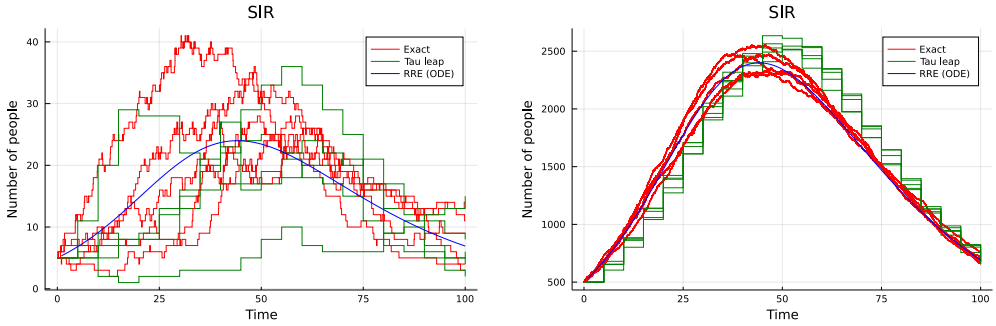


FIGURE 2.3: Effect of the population size on the simulations of the SIR model (only the number of infected people is plotted) with $\theta = (0.12/N_{pop}, 0.05)$ and $\tau = 5.0$. On the left plot, $\mathbf{s}_0 = (95, 5, 0)$ ($N_{pop} = 100$). On the right plot, $\mathbf{s}_0 = (9500, 500, 0)$ ($N_{pop} = 10000$). Red paths are simulated from the SSA. Green paths are simulated from the Tau-leaping algorithm. 5 paths for each algorithm.

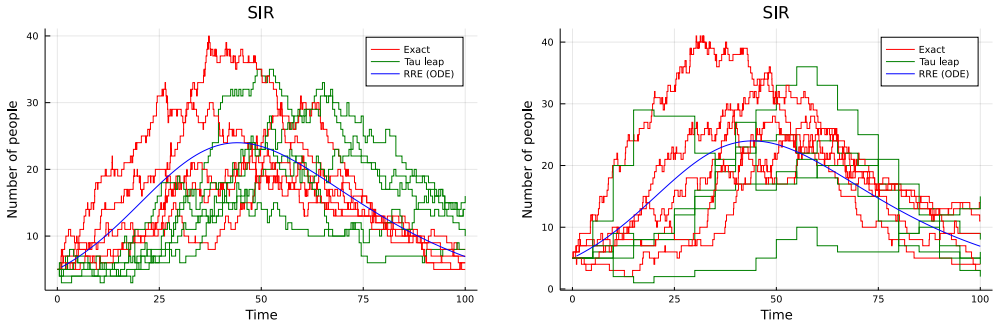


FIGURE 2.4: Effect of the tau-leap time step on the simulations of the SIR model (only the number of infected people is plotted) with $\theta = (0.12/N_{pop}, 0.05)$ and $\mathbf{s}_0 = (95, 5, 0)$ ($N_{pop} = 100$). In the left plot, $\tau = 0.5$. In the right plot, $\tau = 5.0$. Red paths are simulated from the SSA. Green paths are simulated from the Tau-leaping algorithm. 5 paths for each algorithm.

Script 1 (Simulation of the SIR model with SSA, Tau-leaping and ODE.)
This simulation is available in the git repository of this thesis at `/code/chap1/sim_sir.jl`.

2.4 Summary

In this chapter, we described:

- The fundamentals of Markov Chains in both discrete-time and continuous-time with different points of view.
- The construction of a probability measure of a CTMC over the set of trajectories. It will be used in the case of statistical inference and model checking of CTMCs.
- The Chemical Reaction Network formalism, their stochastic dynamics related to Continuous-Time Markov Chains and classical simulation algorithms of trajectories.

Chemical Reaction Networks are our main subject of study to address both statistical inference and verification tasks which are difficult in most cases. But Markov Chains are also the basis of advanced statistical methods such as MCMC the Bayesian method, which is the next chapter's subject.

Chapter 3

Statistical methods

The general point of parametric statistics is to formalise observations data within a probabilistic parametric model to treat, analyse, infer parameters and make predictions from them.

Statistical inference considers probabilistic models parametrised by a vector θ. It aims at estimating the parameters based on observations data.

Usually, two points of view are opposed in statistical inference. The first one, the *frequentist approach*, supposes that a true parameter θ_{true} generates the observations. Then, statisticians construct an estimator $\widehat{\theta}(X^{(1)}, \ldots, X^{(n)})$ of θ_{true}: it is a function of the random observations $X^{(1)}, \ldots, X^{(n)}$ that should converge to θ_{true} and have other convenient statistical properties.

The other one, called *Bayesian inference*, is the one we are the most focused on in this thesis. In this case, we do not suppose that a parameter θ_{true} generates the data. Instead, we define a *prior distribution*: it describes our first beliefs about the parameters. Then, this distribution is updated according to observations. The so-called *posterior distribution* represents our beliefs about the parameters after the integration of observations. We are focused on this type of methods for two main reasons:

- The class of models we are working on is complex, and likelihood computation is, in most cases, intractable. Some groups of methods in the Bayesian approach are likelihood-free. They make likelihood computation dispensable in exchange for less precise inference. Approximate Bayesian Computation is one of them. They have already proven efficiency in the context of Continuous-Time Markov Chains (Warne, Baker, and Simpson, 2019; Alharbi, 2018).
- We use Approximate Bayesian Computation methods to create a statistical verification method for time-bounded reachability problems.

In Bayesian methods, several well-known techniques such as accept-reject, Sequential Monte Carlo, MCMC or Importance Sampling are available to sample parameters from the posterior distribution. However, these methods are not always distinct: they are often connected to create more extensive algorithms to improve performance. Sequential Monte Carlo ABC is an illustrative example. It is thus essential to understand which technique refers to which algorithm to understand the more complex ones. In this Chapter, we describe the different bricks of Bayesian inference to detail a Bayesian inference method we were focused on in our work: Sequential Monte Carlo ABC.

This Chapter is organised as follows. Section 3.1 details the basic notions for Bayesian inference. Section 3.2 provides multiple Monte Carlo algorithms for sampling from a probability density function π computable up to a constant. Section 3.3 details a family of likelihood-free methods called Approximate Bayesian Computation. Section 3.4 describes kernel density estimation, a non-parametric method to obtain a continuous estimation $\widehat{\pi}$ of π after we have already managed to sample from the density π.

3.1 The Bayesian framework

In mathematical statistics, we consider a statistical model (Definition 3.1.1) of a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, i.e. a collection of probability distributions.

Definition 3.1.1
Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. A statistical model is a collection of probability distributions $(\mathbb{P}_\theta)_{\theta\in\Theta}$ indexed by a parameter vector θ.

Usually, statistical models are dominated by a measure that implies a probability density function p_θ for any $\theta \in \Theta$. Thus, the likelihood function is defined based on this density. It measures how probable observations $y^{(1)}, \ldots, y^{(N)}$ are drawn from a probability distribution $\mathbb{P}_\theta$.

Definition 3.1.2
Let $(\mathbb{P}_\theta)_{\theta\in\Theta}$ a parametric statistical model with corresponding densties $(p_\theta)_{\theta\in\Theta}$. Let

$y^{(1)}, \dots, y^{(N)}$ *idependant and identically distributed (i.i.d) observations. The likelihood function is defined as:*

$$l(.; y^{(1)}, \dots, y^{(N)}) : \Theta \to \mathbb{R}_{\geq 0}$$
$$\theta \to p_\theta(y^{(1)}, \dots, y^{(N)}) = \prod_{i=1}^{N} p_\theta(y^{(i)})$$

In the frequentist inference approach, the observations are supposed drawn from a distribution $\mathbb{P}_{\theta_{true}}$, i.e. we suppose that one parameter vector θ_{true} generates the data. An estimation of θ_{true} is obtained via the Maximum Likelihood Estimator (Definition 3.1.3). It consists of maximising the log-likelihood. When this problem is too complex, other methods are available such as the Expectation-Maximisation algorithm, which ensures a local minimum.

Definition 3.1.3 (Maximum Likelihood Estimator)
Considering a statistical model $(p_\theta)_{\theta \in \Theta}$ and i.i.d observations $y^{(1)}, \dots, y^{(N)}$, a maximum likelihood estimator is a parameter $\widehat{\theta} \in \Theta$ that verifies:

$$l(\widehat{\theta}; y^{(1)}, \dots, y^{(N)}) = \sup_{\theta \in \Theta} l(\theta; y^{(1)}, \dots, y^{(N)}).$$

In the following, we describe the *Bayesian inference* framework. This other statistical inference approach is the most focused on in this thesis. As the name mentions, it is based on the Bayes formula, that for two events A and B,

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}.$$

.

In the Bayesian approach, uncertainty about the parameters is modelled by a random variable ϑ. $\mathbb{P}_\vartheta$ is a probability distribution called *prior distribution.* This distribution represents the first beliefs about the parameters before integrating the observations.

In order to formalise the notion of "integration of observations", we need to define the distribution of a random variable conditionally to another, which is embodied by a transition kernel.

Definition 3.1.4 (Transition kernel)
Let $(\mathcal{X}, \mathcal{F}_\mathcal{X})$ and $(\mathcal{Y}, \mathcal{F}_\mathcal{Y})$ be two measurable spaces. The function $Q : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ is a transition kernel if:

1. $\forall x \in \mathcal{X}, Q(x,.)$ *is a probability measure over* $(\mathcal{Y}, \mathcal{F}_\mathcal{Y})$

2. $\forall A \in \mathcal{F}_\mathcal{Y}, Q(.,A)$ *is measurable.*

In the case of two random variables X and Y defined on $(\mathcal{X}, \mathcal{F}_\mathcal{X})$ and $(\mathcal{Y}, \mathcal{F}_\mathcal{Y})$, the transition kernel K that verifies $\forall A, B \in \mathcal{F}_\mathcal{X} \times \mathcal{F}_\mathcal{Y}, \mathbb{P}_{X,Y}(A \times B) = \int_\mathcal{X} Q(x,B) d\mathbb{P}_X$ is called the *conditional law* of Y given X and it is denoted $\mathbb{P}_{Y|\ X}$.

Remark 3.1.1 (Conditional law of a CTMC)
If we consider a collection of CTMC $(S^\theta)_{\theta \in \Theta}$, $(\theta, A) \to \mathbb{P}_{S^\theta}(A)$ *is a transition kernel and admits densities.*

We consider the random variable Y of observations and a prior distribution $\mathbb{P}_\vartheta$. We suppose conditional laws $\mathbb{P}_{Y|\vartheta}$ and $\mathbb{P}_{\vartheta|Y}$ admit densities. Bayesian inference is built upon the Bayes theorem:

$$\begin{aligned} p(\theta|y) &= Z^{-1} p(y|\theta) p(\theta) \\ &\propto p(y|\theta) p(\theta) \end{aligned}$$

where $Z = \int_{\theta' \in \Theta} p(y|\theta') p(\theta') d\theta'$ and $\propto$ means "proportional to". $p(.|y)$ is called the *posterior density* of the observation y. This distribution shows how the belief about the parameters is updated after observations of the studied phenomenon. Bayesian estimation deals with the *integration problem* (instead of the optimisation problem for the frequentist approach) because Z has an integral form. We want to estimate the posterior distribution by estimating the constant Z or even avoiding its computation when it is too complex.

3.2 Monte Carlo methods

Monte Carlo methods are a class of algorithmic methods based on simulations of the model. They first appeared after WWII (see (Nicholas Metropolis, 1987) for the emergence of the method). Given a probability $\mathbb{P}_*$ with density π, the idea is to sample $x^{(1)}, \ldots, x^{(n)}$ from π to estimate the expectation of a random variable with density π.

However, simulate samples from π when it is not fully known analytically is difficult. Nicholas Metropolis formulates the first Monte Carlo method based on Markov Chains theory (Metropolis et al., 1953; Hastings, 1970) when π is known up to a constant. Due to the maturity of methods and the fast computing power increase

of computers, such related Monte Carlo methods are considered standard techniques in the 2000s (Robert and Casella, 2004).

Until *the end of the Chapter*, the unknown target density to estimate is called π.

Definition 3.2.1 (Density of the empirical distribution)
Let X be a random variable with a density π on $\mathcal{X}$. We consider $X^{(1)}, \dots, X^{(N)}$ N i.i.d. (independent and identically distributed) observations drawn from π with a realisation $x^{(1)}, \dots, x^{(N)}$. The density function of the empirical distribution is defined as:

$$\widehat{\pi}(x) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x^{(i)}}(x)$$

By Glivenko-Cantelli theorem, $\frac{1}{N} \sum_{i=1}^{N} \delta_{x^{(i)}}(x) \xrightarrow{a.s.} \pi(x)$ which means $\widehat{\pi}$ is an empirical approximation of the targeted density π. This can lead to an approximation of the expectation for any function h:

$$\mathbb{E}_{\pi}[h(X)] = \int_{\mathcal{X}} h(x)\pi(x)dx \approx \int_{\mathcal{X}} h(x)\widehat{\pi}(x)dx = \frac{1}{N} \sum_{i=1}^{N} h(x_i)$$

The main issue of this estimation procedure is to know how to properly simulate from the density π to get enough samples $x^{(i)}$ in a reasonable time so that the density of the empirical distribution is a good enough approximation of the true probability density function π. Description of the simulation algorithms is the subject of the following sections. The knowledge about π is decreasing as we go along the chapter:

- In Section 3.2.1, π is fully known.
- In Sections 3.2.2, 3.2.3, 3.2.4, 3.2.5, π is known up to a constant, i.e. one can compute a function $\tilde{\pi}(x) \propto \pi(x)$.
- In Section 3.3, the target density π is a possibly intractable posterior distribution. It describes a likelihood-free method that samples approximatively the posterior distribution.

These methods are of much importance in Bayesian inference. Indeed, the classical application of these methods is when π is the posterior distribution $p(.|y)$, and $\tilde{\pi}$ is $p(y|.)p(.)$ because of the Bayes formula $p(\theta|y) \propto p(y|\theta)p(\theta)$.

3.2.1 Simulation of a density

The simulation of random variables is based on an available generator of a uniform random variable $\mathcal{U}(0,1)$. Then, each simulation is based on the generalised inverse of a cumulative distribution function (cdf).

Proposition 3.2.1 (Probability integral transform)
Let $U = \mathcal{U}(0,1)$ be a uniform random variable over $[0,1]$. Let X be a random variable with cumulative distribution function (cdf) F ($F(x) = \mathbb{P}(X \leq x)$). Then $F^{-1}(U)$ has the distribution F, where:

$$F^{-1}(u) = \inf\{x/F(x) \geq u\}$$

is the generalised inverse of F.

Any random variable is represented by a transformation of a uniform variable. If X has a cdf F, it suffices to simulate $u \sim \mathcal{U}(0,1)$ and take $x = F^{-1}(u)$. This procedure of simulation requires an explicit computation of F^{-1}.

3.2.2 Accept-reject algorithm

Sometimes the simulation of the target density π is too complex. The idea of the accept-reject algorithm is to use an auxiliary density $q(.)$ to get samples from π.

Let $\tilde{\pi}(x) \propto \pi(x)$ be an unnormalised density function proportional to the target density. If the ratio $\frac{\tilde{\pi}(x)}{q(x)}$ is bounded, the accept-reject algorithm is guaranteed to generate samples from π, i.e. when:

$$\exists M \leq 0, \forall x, \tilde{\pi}(x) < Mq(x) \tag{3.1}$$

Algorithm 4 samples from $q(.)$ and accepts the sample with a probability that depends on π. The inequality 3.1 ensures that the sample is drawn from π.

Algorithm 4 Accept-reject algorithm

Require: q proposal density

Ensure: $(x^{(i)})_{1 \leq i \leq N}$ N samples drawn from π

for $i = 1 : N$ **do**

repeat

$x \sim q(.)$

$u \sim \mathcal{U}(0, 1)$

until $u < \frac{\tilde{\pi}(x)}{Mq(x)}$

$x^{(i)} \leftarrow x$

end for

One can prove that the probability of acceptance is $\frac{\int \tilde{\pi}(x)dx}{M}$, which means M should be as small as possible.

3.2.3 Importance sampling

Another way of simulating the target distribution from an auxiliary density q is the important sampling. In this algorithm, no bound condition is required except that the support of q should include the support of π. The idea is to sample N i.i.d. samples $x^{(1)}, \ldots, x^{(N)}$ from the *importance distribution* q, and correct the contribution of each sample $x^{(i)}$ in the empirical distribution 3.2.1 by a weight $w(x^{(i)})$ called *importance weight*. The distribution associated with the Importance Sampling for $x^{(1)}, \ldots, x^{(N)}$ drawn from q is:

$$\widehat{\pi}_{IS}(x) = \frac{1}{N} \sum_{i=1}^{N} \underbrace{\frac{\pi(x^{(i)})}{q(x^{(i)})}}_{w(x^{(i)})} \delta_{x^{(i)}}(x)$$

.

This estimator is unbiased for integration estimation, indeed

$$\begin{aligned}
\mathbb{E}_{\pi}[h(X)] &= \int_{\mathcal{X}} h(x)\pi(x)dx \\
&= \int_{\mathcal{X}} h(x)\frac{p(x)}{q(x)}q(x)dx \\
&= \mathbb{E}_q[h\frac{\pi}{q}(X')] \\
&\approx \frac{1}{N}\sum_{i=1}^{N} h(x^{(i)})\frac{\pi(x^{(i)})}{q(x^{(i)})}
\end{aligned}$$

As detailed before, sometimes π is only known up to a constant, i.e. $\pi(x) \propto \tilde{\pi}(x)$.

To bypass this issue, one can compute the weight with $\tilde{\pi}$ and normalise it by dividing by the sum of all weights $\sum_{i=1}^{N} w(x^{(i)})$.

3.2.4 Sequential Monte Carlo methods

In complex cases, finding a proposal density q that will aim at the crucial regions of the target density π is difficult. Sequential Monte Carlo methods (Doucet, De Freitas, and Gordon, 2001; Del Moral, Doucet, and Jasra, 2006; Liu, 2008) were designed to overcome this issue. The idea is to sample from intermediate distributions $(\pi_m)_{m\in 0:M}$ sequentially with proposals $(q_m)_{m\in 0:M}$ so that $\pi_M = \pi$. These methods are helpful, especially in two cases:

- State-space models. If for one observation $x_{0:M}$, $M+1$ points are available, it is natural to set π_m as the density of the truncated observation $p(x_{0:m})$, $m \leq M$. As $x_{m+1} \sim p(.|x_m)$, finding a sequential scheme based on each observation point is more accessible. In this case, the density dimension increases. The target distribution π is the full density $p(x_{0:M})$.
- Densities with decreasing supports. π may have tiny unknown support compared to the space of observations. A strategy is to define a sequence of densities with decreasing support to find more efficiently the small support region of π. In this case, the densities dimensions may be fixed. This paradigm is used in the Sequential Monte Carlo version of the Approximate Bayesian Computation method (Del Moral, Doucet, and Jasra, 2012a), which we will discuss later. It is related to rare event analysis and subset simulation.

Sequential Importance Sampling

Sequential Importance Sampling is a specific method of Sequential Monte Carlo techniques to overcome issues related to high-dimensional problems related to Importance Sampling. In this part, we suppose each observation x is decomposable, i.e. $x = x_{0:M}$.

As mentioned before, we want to sample from intermediate densities $(\pi_m)_{m\in 0:M}$, i.e. from π_0 to $\pi_M = \pi$. At each step m, we sample from a proposal density q_m. A possible choice of proposal densities (Del Moral, Doucet, and Jasra, 2006) at each step is:

$$q_m(x_{0:m}) = \tilde{q}_m(x_m|\ x_{0:m-1}) q_{m-1}(x_{0:m-1})$$

The computation of the weight in Importance Sampling for the intermediate distributions can be rewritten as:

$$w_m(x_{0:m}) = \frac{\pi_m(x_{0:m})}{q_m(x_{0:m})} = \underbrace{\frac{\pi_{m-1}(x_{0:m-1})}{q_{m-1}(x_{0:m-1})}}_{w_{m-1}(x_{0:m-1})} \frac{\pi_m(x_{0:m})}{\pi_{m-1}(x_{0:m-1})\tilde{q}_m(x_m|\ x_{0:m-1})}$$

which gives a sequential scheme for the computation. This is the Sequential Importance Sampling method detailed in Algorithm 5.

This scheme is particularly interesting in the case of state-space models because the proposal densities can be chosen so that they verify the Markov property, i.e. $\tilde{q}_m(x_m|x_{0:m-1}) = \tilde{q}(x_m|x_{m-1})$ (Del Moral, Doucet, and Jasra, 2006; Doucet and Johansen, 2009).

Algorithm 5 Sequential Importance Sampling Algorithm

Require: $(q_m)_m$ proposal densities, $(\tilde{\pi}_m)_m$ unnormalised intermediate densities

Ensure: $(x_{0:M}^{(i)})_{1\leq i\leq N}$ N samples drawn from π

```
for i = 1 : N do
    x_0^(i) ~ q_0
    w_0^(i) ← π̃_0(x_0^(i)) / q_0(x_0^(i))
end for
Normalise (w^(i))_i
for m = 1 : M do
    for i = 1 : N do
        x_m^(i) ~ q̃_m(.|x_{0:m-1}^(i))
        w_m^(i) ← w_{m-1}^(i) π̃_m(x_{0:m}^(i)) / (π̃_{m-1}(x_{0:m-1}^(i)) q̃_m(x_m^(i)|x_{0:m-1}^(i)))
    end for
    Normalise (w_m^(i))_i
end for
```

Resampling step

As the number of steps M increases, the estimator's variance associated with (S)IS procedures can increase exponentially even in simple cases (Doucet and Johansen, 2009). Resampling is a method to overcome this issue.

During the procedure, some samples can have low weights, but our goal is to focus on high probability regions. These low weighted samples will perturb the whole estimation. The more the number of low weights increases, the more the number of valuable samples decreases. This phenomenon is known as *particle degeneracy*

(Gordon, Salmond, and Smith, 1993; Liu and Chen, 1998): after some steps, only a few samples will be valuable in our estimation.

One possible solution is to add a step where the low weighted samples are replaced by high weighted samples according to a distribution parametrised by the weights: this is the *resampling step*. A common way to run this task is multinomial resampling: the new indices of samples $(x_{0:m}^{(i)})_{i\in 1:N}$ are drawn from a multinomial distribution $\mathcal{M}$ with parameters $N, (w_m(x_{0:m}^{(i)}))$.

Algorithm 6 Multinomial resampling

Require: $(x_{0:m}^{(i)})_{1:N}$ N samples with associated weights $(w_m^{(i)})_{1:N}$
Ensure: Resampling of $(x_{0:m}^{(i)})_{1:N}$
  $j_{1:N} \sim \mathcal{M}(N, (w_m^{(i)})_{1:N})$
  for $i = 1 : N$ **do**
    $x_{0:m}^{(i)} \leftarrow x_{0:m}^{(j_i)}$
    $w_{0:m}^{(i)} \leftarrow \frac{1}{N}$
  end for

One could think this resampling step should be done at every step m to delete the useless samples directly. In pratice, a low weight sample at a step m can have a higher weight at a step $m+1$. There is a trade-off between the occurrences of resampling and stability: this is *adaptive resampling* (Del Moral, Doucet, and Jasra, 2012b).

Effective sample size (Kong and Liu, 1994) is a statistic to quantify the discrepancy (variability) between the weights. It is defined as:

$$ESS((w^{(i)})_{i\in 1:N}) = \frac{1}{\sum_{i=1}^{N}(w^{(i)})^2}.$$

ESS takes values between 1 (one weight is equal to one and the others 0, worst case) and N (perfectly balanced weights equal $\frac{1}{N}$). Thus, one can use this statistic to construct a condition that will decide when we should resample or not. A threshold N_T is set (typically $N_T = \frac{N}{2}$), and if the ESS is lower than the threshold, we resample with Algorithm 7.

Algorithm 7 Adaptive resampling step with Effective Sample Size

Require: $(x_{0:m}^{(i)})_{1\leq i\leq N}$ N samples with associated weights $(w_m^{(i)})_{1:N}$

Ensure: Resampling of $(x_{0:m}^{(i)})_{1\leq i\leq N}$

$N_{ESS} \leftarrow ESS((w_m^{(i)})_{1:N})$

if $N_{ESS} < N_T$ **then**

$j_{1:N} \sim \mathcal{M}(N, (w_m^{(i)})_{1:N})$

for $i = 1 : N$ **do**

$x_{0:m}^{(i)} \leftarrow x_{0:m}^{(j_i)}$

$w_{0:m}^{(i)} \leftarrow \frac{1}{N}$

end for

end if

This resampling step can be then integrated into generic Sequential Monte Carlo procedures. An example with Sequential Importance Sampling is described in Algorithm 8.

Algorithm 8 Sequential Importance Sampling with Resampling Algorithm

Require: $(q_m)_m$ proposal densities, $(\tilde{\pi}_m)_m$ unnormalised intermediate densities

Ensure: $(x_{0:M}^{(i)})_{1\leq i\leq N}$ N samples drawn from π

for $i = 1 : N$ **do**

$x_0^{(i)} \sim q_0$

$w_0^{(i)} \leftarrow \frac{\tilde{\pi}_0(x_0^{(i)})}{q_0(x_0^{(i)})}$

end for

Normalise $(w^{(i)})_i$

Resampling step for $(x_0^{(i)})_{1:N}, (w_0^{(i)})_{1:N}$ as in Algorithm 6 or 7

for $m = 1 : M$ **do**

for $i = 1 : N$ **do**

$x_m^{(i)} \sim \tilde{q}_m(.|x_{0:m-1}^{(i)})$

$w_m^{(i)} \leftarrow w_{m-1}^{(i)} \frac{\tilde{\pi}_m(x_{0:m}^{(i)})}{\tilde{\pi}_{m-1}(x_{0:m-1}^{(i)})\tilde{q}_m(x_m^{(i)}|x_{0:m-1}^{(i)})}$

end for

Normalise $(w^{(i)})_i$

Resampling step for $(x_{0:m}^{(i)})_{1:N}, (w_{0:m}^{(i)})_{1:N}$ as in Algorithm 6 or 7

end for

3.2.5 Markov Chain Monte Carlo

Markov Chain Monte Carlo methods are another type of Monte Carlo methods that sample from a target density π by knowing $\tilde{\pi} \propto \pi$. In Section 2.1.1 of Chapter 2, we quickly recalled basis of ergodicity theory quickly for discrete state-space Markov

Chains. The idea of Markov Chain Monte Carlo methods is to define a Markov Chain *whose stationary distribution has density* π.

However, MCMC methods aim at estimating distributions with continuous state space $\mathcal{X}$. The Markov property of a Markov Chain $(X_n)_{n\in\mathbb{N}}$ with a continuous measurable space $(\mathcal{X}, \mathcal{F}_\mathcal{X})$ is formulated as:

$$\mathbb{P}(X_{n+1} \in A_{n+1} | X_n = x_n, X_{n-1}, \dots, X_n) = \mathbb{P}(X_{n+1} \in A | \ X_n = x_n).$$

The event $\{X_n = x\}$ may have zero probability, so we define a continuous generalisation of n-step transition probabilities (Definition 2.1.3) with the transition kernel (Definition 3.1.4):

$$P^{(n)}(x, A) = \mathbb{P}(X_n \in A | X_0 = x).$$

In this context, a stationary distribution Π of a Markov Chain with a transition kernel P is defined as:

$$\Pi(A) = \int_\mathcal{X} P(x, A) d\Pi(x).$$

Similarly to Theorem 2.1.2 for discrete state-space Markov Chains, ergodic theorems for Markov Chain's convergence to its stationary distribution exist in the continuous state-space case. It is based on Harris Chains, which redefine more generally ergodicity of Markov Chains (Bladt and Nielsen, 2017, Chapter 7.3).

Construction of Markov Chains with a targeted stationary distribution is based on the balance condition.

Proposition 3.2.2 (Balance condition)
Let $(X_n)_{n\in\mathbb{N}}$ be a Markov Chain with a continuous state space $\mathcal{X}$ and a transition kernel P. We suppose that for all $x \in \mathcal{X}$ the probability measure on $(\mathcal{X}, \mathcal{F}_\mathcal{X})$ $P(x, .)$ has a density $p(x, y)$. Let π be a density. If π and p verify the detailed balance condition (or time reversibility conditon):

$$\forall x, y \in \mathcal{X}, \pi(x)p(x, y) = \pi(y)p(y, x) \tag{3.2}$$

then π is a density of a stationary measure of the Markov Chain.

The Metropolis-Hasting algorithm (Metropolis et al., 1953; Hastings, 1970) is the first formulation of an MCMC algorithm. The idea is to sample a chain from the

Markov Chain defined by:

$$p(x_n, x_{n+1}) = \alpha(x_n, x_{n+1}) q(x_n, x_{n+1}), \tag{3.3}$$

where

$$\alpha(x_n, x_{n+1}) = \min(1, \frac{\tilde{\pi}(x_{n+1}) q(x_{n+1}, x_n)}{\tilde{\pi}(x_n) q(x_n, x_{n+1})}) = \min(1, \frac{\pi(x_{n+1}) q(x_{n+1}, x_n)}{\pi(x_n) q(x_n, x_{n+1})})$$

and for $x_n \in \mathcal{X}$, $q(x_n, .)$ is a proposal density for the next chain's value.

The transition kernel 3.3 verifies the detailed balance condition 3.2, which ensures the Metropolis-Hasting Algorithm's convergence (Algorithm 9) to the stationary distribution π.

Algorithm 9 Metropolis-Hasting Algorithm

Require: A proposal density $q(.,.)$, computable $\tilde{\pi} \propto \pi$

Ensure: $(x_n)_{0 \leq n \leq N}$ drawn from a Markov Chain with limit stationary distribution π

```
Initialise a value x_0
for n = 0 : N − 1 do
    x* ∼ q(x_n, .)
    u ∼ U(0, 1)
    x_{n+1} ← x*_n
    if u ≤ α(x_n, x*) then
        x_{n+1} ← x*
    end if
end for
```

Another popular MCMC method is Gibbs sampling (Geman and Geman, 1984). This sampler is useful when the variable of the targeted distribution is decomposable into several blocks $\pi(x) = \pi(x_{[1]}, \ldots, x_{[k]})$.

Let $x_n = (x_{n[1]}, \ldots, x_{n[k]})$. The sampling of $x_{n+1} = (x_{n+1[1]}, \ldots, x_{n+1[k]})$ is divided into k steps, where the i-th step ($i \in \{1, \ldots, k\}$) is:

$$x_{n+1[i]} \sim \pi(.|x_{n+1[1]}, \ldots, x_{n+1[i-1]}, x_{n[i+1]}, \ldots, x_{n[k]})$$

Then each sampling step $x_{n[i]}$ is equivalent to a Metropolis-Hasting algorithm step of acceptance probability $\alpha = 1$ whose proposal distribution q is:

$$\begin{aligned} q(x, (x_{[1]}, \dots, x_{[i-1]}, y, x_{[i+1]}, \dots, x_{[k]})) &= \pi(y|x_{[-i]}) \\ &= \frac{\pi(x_{[1]}, \dots, x_{[i-1]}, y, x_{[i+1]}, \dots, x_{[k]})}{\pi(x_{[-i]})} \end{aligned}$$

where y has the same dimension of $x_{[i]}$ and

$$x_{[-i]} = x_{[1]}, \dots, x_{[i-1]}, x_{[i+1]}, \dots, x_{[k]}.$$

Thus, the sampling of x_{n+1} given x_n is equivalent to k Metropolis Hasting steps. One can prove that the target distribution is π (Robert and Casella, 2004), which gives the Algorithm 10.

Algorithm 10 Gibbs Sampling

Require: A target distribution π

Ensure: $(x_n)_{0:N}$ drawn from a Markov Chain with limit stationary distribution π

```
Initialise a value x_0
for n = 0 : N − 1 do
    x* ← x_n
    for i = 1 : k do
        y ∼ π(.|x*_[−i])
        x*_[i] ← y
    end for
    x_{n+1} ← x*
end for
```

More advanced MCMC methods based on Metropolis-Hasting, Gibbs and Sequential Monte Carlo methods such as particle filtering (Andrieu, Doucet, and Holenstein, 2010) were developed in the literature. We will not detail these methods here, but we refer the reader to (Brooks et al., 2011), (Gilks, Richardson, and Spiegelhalter, 1996) for a deeper exploration of MCMC methods.

3.3 Approximate Bayesian Computation: a likelihood-free method

So far, the Monte Carlo methods we presented aim at sampling from a distribution π, supposing it is computable up to a constant, i.e. one can compute $\tilde{\pi} \propto \pi$.

Approximate Bayesian Computation (ABC) is a Bayesian method for the approximate estimation of the posterior distribution when the likelihood is computationally intractable or too expensive. ABC is not the only likelihood-free method; other algorithms exist, such as pseudo-marginal MCMC (Andrieu and Roberts, 2009).

ABC algorithms have gained popularity over the last decade and are applied for parameter inference or model selection in many modelling fields, including systems biology (Toni et al., 2009; Ratmann et al., 2007; Koutroumpas et al., 2016; Lenive, Kirk, and Stumpf, 2016) or ecology (Beaumont, 2010; Fasiolo and Wood, 2015). The first fundamental work was developed in the field of population genetics (Pritchard et al., 1999), which was motivated by the high dimension and structure of the studied models. ABC methods have proved influential in many applications when classical Bayesian parameter inference methods are challenging to implement, including CTMCs (Warne, Baker, and Simpson, 2019; Alharbi, 2018).

In this section, we suppose that we observe $y_{exp} = (y^{(1)}, \dots, y^{(N)})$ N i.i.d. observations with $y^{(i)} \in \mathcal{Y}^{(i)}$ potentially multidimensional, and we consider and prior distribution with density $\theta \to p(\theta)$ and a likelihood $p(.|\theta)$. According to Bayes theorem,

$$p(\theta|y_{exp}) \propto p(y_{exp}|\theta)p(\theta).$$

In complex models, $p(y_{exp}|\theta)p(\theta)$ is not even computable. For example, in the case of CTMCs, we will see that the likelihood for time-discrete observations is intractable in most cases (Section 5.1.2).

3.3.1 ABC Rejection algorithm

The ABC method relies on the same principle as Rejection Algorithm 4 (Sisson, Fan, and Beaumont, 2018), except that it aims at drawing samples from an auxiliary distribution π_{ABC}. The idea consists in drawing parameters from a proposal density q, then simulate the model $y \sim p(.|\theta)$ and accept (θ, y) with a probability proportional to the discrepancy between y and y_{exp}: the closer y is to y_{exp}, the more probable it is to accept the sample.

The quantification of the discrepancy between two samples is based on three functions:

- The summary statistics function $\eta : \mathcal{Y} \to \mathrm{H}$. It computes statistics over the simulation result to reduce the dimension of the simulation while maximising the information contained in it.

- The distance function $\rho : \mathrm{H} \times \mathrm{H} \to \mathbb{R}_{\geq 0}$. It is a distance over the summary statistics space H.
- A kernel function K_ϵ with a scale parameter ϵ.

Algorithm 11 ABC Rejection Sampling Algorithm

Require: q proposal density, K_ϵ kernel function with a scale parameter ϵ, $M \geq K_\epsilon(0) \max_\theta \frac{\pi(\theta)}{q(\theta)}$

Ensure: $(\theta^{(i)}, y^{(i)})_{1 \leq i \leq n}$ drawn from the ABC posterior π_{ABC}

for $i = 1 : n$ **do**

repeat

$\theta \sim q$

$y \sim p(.|\theta)$

$u \sim \mathcal{U}(0, 1)$

until $u < \frac{K_\epsilon(\rho(\eta(y), \eta(y_{exp})))p(\theta)}{Mq(\theta)}$

$\theta^{(i)}, y^{(i)} \leftarrow \theta, y$

end for

By denoting $\tilde{q}(\theta, y) = q(\theta)p(y|\theta)$ and noticing that:

$$\frac{K_\epsilon(\rho(\eta(y), \eta(y_{exp})))p(\theta)}{Mq(\theta)} = \frac{K_\epsilon(\rho(\eta(y), \eta(y_{exp})))p(y|\theta)p(\theta)}{Mq(\theta)p(y|\theta)},$$

Algorithm 11 is equivalent to Algorithm 4 with the proposal density $\tilde{q}$. Thus, the samples of Algorithm 11 are drawn from the distribution:

$$\pi^\epsilon_{ABC}(\theta, y| \ y_{exp}) \propto K_\epsilon(\rho(\eta(y), \eta(y_{exp})))p(y|\theta)p(\theta).$$

It is not clear that this density is an approximation of the posterior distribution. Let us compute the marginal distribution of θ when η is the identity function.

$$\pi^\epsilon_{ABC}(\theta|y_{exp}) \propto \int_{\mathcal{Y}} \pi^\epsilon_{ABC}(\theta, y|y_{exp})dy$$

$$\begin{aligned}
\lim_{\epsilon \to 0} \pi^\epsilon_{ABC}(\theta|y_{exp}) &\propto \lim_{\epsilon \to 0} \int_{\mathcal{Y}} \pi^\epsilon_{ABC}(\theta, y|y_{exp})dy \\
&\propto \lim_{\epsilon \to 0} \int_{\mathcal{Y}} K_\epsilon(\rho(y, y_{exp}))p(y|\theta)p(\theta)dy \\
&\propto \int_{\mathcal{Y}} \delta_{y_{exp}}(y)p(y|\theta)p(\theta)dy \\
&\propto p(y_{exp}|\theta)p(\theta)
\end{aligned}$$

It shows that the smaller the tolerance level ϵ is, the closer we are from the posterior. It is then essential to understand the different levels of approximations of ABC:

$$\widehat{\pi^{\epsilon}_{ABC}}(.|\eta(y_{exp})) \underset{(1)}{\to} \pi^{\epsilon}_{ABC}(.|\eta(y_{exp})) \underset{(2)}{\to} p(.|\eta(y_{exp})) \underset{(3)}{\approx} p(.|y_{exp})$$

1. How the empirical posterior $\widehat{\pi^{\epsilon}_{ABC}}(.|\eta(y_{exp}))$ based on the samples $(\theta_i, y_i)_i$ is close to π^{ϵ}_{ABC}?
2. How to sample efficiently with small tolerance ϵ to be close to $p(.|\eta(y_{exp}))$?
3. How informative should be η for $p(.|\eta(y_{exp}))$ to be a good approximation of the true posterior $p(.|y_{exp})$?

Remark 3.3.1 (Proposal density and kernel function)
In most of our applications, we use the prior distribution as the proposal density, i.e. $q(\theta) = p(\theta)$. *Also, the kernel used is the indicator function* $K_\epsilon(u) = \frac{1}{2\epsilon}\mathbb{1}(u \leq \epsilon)$. *Then, the acceptance probability condition becomes* $\rho(\eta(y), \eta(y_{exp}))$. *These assumptions lead to a simpler ABC rejection algorithm given in Algorithm 12.*

Algorithm 12 Simple ABC rejection

Require: $p(.)$ prior, ρ, η, ϵ
Ensure: $(\theta^{(i)}, y^{(i)})_{1 \leq i \leq n}$ drawn from the ABC posterior π^{ϵ}_{ABC}
  for $i = 1 : n$ **do**
    repeat
      $\theta \sim p(.)$
      $y \sim p(.|\theta)$
    until $\rho(\eta(y), \eta(y_{exp})) \leq \epsilon$
    $\theta^{(i)}, y^{(i)} \leftarrow \theta, y$
  end for

Remark 3.3.2 (ABC as a Bayesian inference with auxiliary likelihood)
ABC can be seen as a standard Bayesian inference method with the likelihood

$$p_{ABC}(y_{exp}|\theta) = \int_{\mathcal{Y}} K_\epsilon(\rho(\eta(y), \eta(y_{exp})))p(y|\theta)dy.$$

An example based on Gaussian law is detailed in Appendix C. The approximate likelihood is analytically computed, which allows showing ABC in practice and test our algorithms.

In fine, ABC methods aim to draw samples from an auxiliary posterior distribution that approximates the true posterior distribution. Thus, one can use Monte

Carlo methods developed in 3.2.4 and 3.2.5 with the auxiliary ABC posterior π^{ϵ}_{ABC} as the target density: it is the goal of the following sections.

Markov Chain Monte Carlo ABC

A first improvement of the ABC rejection was proposed by (Marjoram et al., 2003). The idea is to create a sampler based on MCMC methods with stationary distribution π^{ϵ}_{ABC}.

Algorithm 13 Monte Carlo Markov Chain ABC

Require: q a proposal density, ϵ, ρ, η

Ensure: $(\theta_n)_{0 \leq n \leq N}$ drawn from a Markov Chain with limit stationary distribution π^{ϵ}_{ABC}.

Find a value θ_0 with Algorithm 12.

for $n = 0 : N - 1$ **do**

$\theta^* \sim q(.|\theta_n)$

$y^* \sim p(.|\theta*)$

$u \sim \mathcal{U}(0, 1)$

$\theta_{n+1} \leftarrow \theta_n$

if $u \leq \frac{p(\theta^*)q(\theta_n|\theta^*)}{p(\theta_n)q(\theta^*|\theta_n)}$ and $\rho(\eta(y^*), \eta(y_{exp})) \leq \epsilon$ **then**

$\theta_{n+1} \leftarrow \theta_n$

end if

end for

Sequential Monte Carlo ABC

If the tolerance ϵ is tiny, the ABC rejection algorithm can be computationally prohibitive. Thus, an alternative is to create a sampler based on Sequential Monte Carlo method 3.2.4. The idea is to consider an intermediate sequence of tolerances $(\epsilon_m)_{0:M}$ with $\epsilon_M = \epsilon$. Naturally, the intermediate distributions chosen for the Sequential Monte Carlo algorithm are:

$$\pi_m = \pi^{\epsilon_m}_{ABC}$$

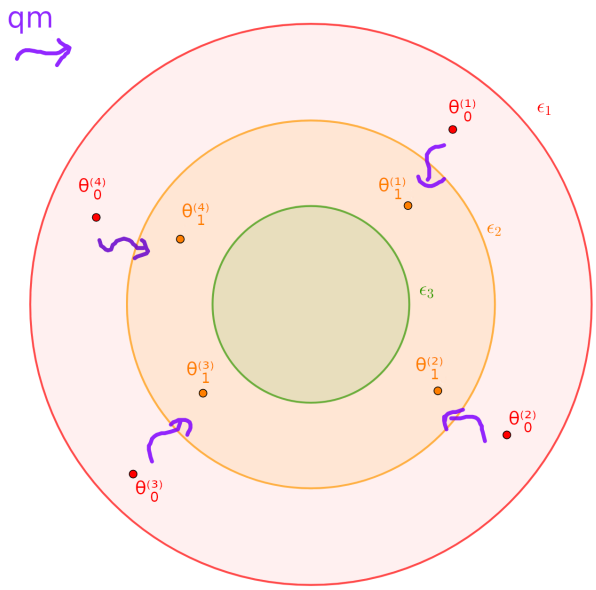


FIGURE 3.1: An illustration of the SMC ABC algorithm

In the following, we present the SMC sampler (Algorithm 14) of (Beaumont et al., 2009), which is based on (Sisson, Fan, and Tanaka, 2007) called Population Monte Carlo ABC (ABC-PMC). At first, we try to sample from $\pi_{ABC}^{\epsilon_0}$ with Algorithm 12. ϵ_0 should be big enough to have a moderate initialisation time (otherwise the SMC algorithm is useless). Sometimes, ϵ_0 equals $+\infty$, which is equivalent to sample from the prior. Then, at each step m, a particle $\theta_{m-1}^{(i)}$ is moved locally through a proposal density $\tilde{q}_m$ (for example, a Gaussian kernel), as depicted in Figure 3.1. If the resulting simulation y' of $\theta_m^{(i)}$ verifies the acceptance condition $\rho(\eta(y'), \eta(y_{exp}))$, $\theta_m^{(i)}$ is a sample from $\pi_{ABC}^{\epsilon_m}$: the parameter is selected. Also, a multinomial resampling procedure is taken into account (Algorithm 6).

Algorithm 14 Population Monte Carlo Algorithm ABC

Require: N: number of particles, y_{exp}, $(\epsilon_m)_{0:M}$, ρ, η, $(\tilde{q}_m)_{0:M}$

Ensure: $(w_M^{(i)}, \theta_M^{(i)})_{1\leq i\leq N}$ weighted samples drawn from $\pi_{ABC}^{\epsilon_M}$

Iteration $m = 0$: find $(\theta_0^{(i)})_{1\leq i\leq N}$ with Algorithm 12 and ϵ_0.

$w_0^{(i)} \leftarrow \frac{1}{N}$, $i \in \{1, \dots, N\}$

for $m = 1 : M$ **do**

  for $i = 1 : N$ **do**

    repeat

      Take θ' from $(\theta_{m-1}^{(j)})_{1\leq jN}$ with probabilities $(w_{m-1}^{(j)})_{1\leq j\leq N}$

      $\theta_m^{(i)} \sim \tilde{q}_m(.|\theta')$

      $y' \sim p(.|\theta_m^{(i)})$

    until $\rho(\eta(y'), \eta(y_{exp})) \leq \epsilon_m$

    $w_m^{(i)} \leftarrow \frac{p\left(\theta_m^{(i)}\right)}{\sum_{i'=1}^{N} w_{m-1}^{(i')} \tilde{q}_m(\theta_m^{(i)}|\theta_{m-1}^{(i')})}$

  end for

  Normalise $(w_m^{(i)})_{1\leq i\leq N}$

end for

(Del Moral, Doucet, and Jasra, 2012a) also proposes a generic ABC-SMC sampler based on (Del Moral, Doucet, and Jasra, 2006), with the possibility of several simulations per particle, an ESS resampling step 7, and an approximation of the weights based on the kernel used in the ABC posterior. However, in the case of a uniform kernel in the acceptance condition, many weights can equal zero after one step (because they do not satisfy the acceptance condition). For a deeper review of ABC samplers, we refer the reader to (Fan and Sisson, 2018; Marin et al., 2011).

3.3.2 Hyperparameters of ABC methods

Summary statistics

When the dimension of y_{exp} is high, to satisfy the acceptance condition is challenging: this issue is known as the *curse of dimensionality*. Thus, a summary statistics function η is introduced to compute statistics over the observations to reduce the dimension and keep the most complete amount of information contained in the observations. The perfect case is when the statistics are *sufficient* (Definition 3.3.1). It means that the reduction of the dimension by the summary statistics causes no lack of information. Unfortunately, the Pitman-Koopman-Darmois theorem states that sufficient statistics whose dimension is bounded while sample size increases only exist for distributions

of the exponential family. A practical analytical example with a sufficient statistic is detailed in Appendix C.

Definition 3.3.1 (Sufficient statistic)
Let Y and ϑ be random variables. η is a sufficient statistic if $\mathbb{P}_{Y|\ \eta(Y)=\eta(y),\vartheta=\theta}$ does not depend on θ for any y, θ.

Thus, selecting efficient statistics for the observations is challenging and depends on the type of model knowledge (e.g. for time series (Jasra, 2015)). It can be seen as a task of *information theory*: how can we retain maximally the information contained in the samples with a fixed dimension? Thus, several methods based on mutual information, projection techniques, subset selection or auxiliary likelihood were proposed. We refer the reader to (Blum et al., 2012; Prangle, 2015) for a complete review of these methods.

Distance function

The function ρ is a distance function over the summary statistics space H ($\mathrm{H} = \mathcal{Y}$ when $\eta = id$), i.e. it verifies symmetry, separation and triangle inequality. The most common choice is the Euclidean distance:

$$\begin{aligned} \rho : \mathrm{H} \times \mathrm{H} &\to \mathbb{R}_{\geq 0} \\ (\eta^{(1)}, \eta^{(2)}) &\to \sqrt{\sum_j (\eta_j^{(1)} - \eta_j^{(2)})^2} \end{aligned}$$

Recently, considering the simulations as empirical distributions and using distances related to distributions like Kullback-Leibler divergence, Wasserstein or Slice-Wasserstein distances were proposed (Bernton et al., 2019; Nadjahi et al., 2020). In the last two chapters, we will propose new distances to address problems of inference and verification of CTMCs.

Perturbation kernel in ABC-PMC Algorithm

In the SMC sampler of Algorithm 14, a perturbation kernel $\tilde{q}_m$ is involved in the local moves of the particles at each step. In most of our computations, we use the multivariate Gaussian kernel:

$$\tilde{q}_m(.|\theta') \sim \mathcal{N}(\theta', \widehat{\Sigma}_m)$$

where $\widehat{\Sigma}_m$ is the empirical covariance matrix of the parameters $(\theta_m^{(i)})_i$. Other kernels were proposed in (Filippi et al., 2011), for example, a multivariate Gaussian kernel with the M nearest neighbours. We tested this method. However, even if it can result in slightly fewer simulations, in practice we observed that the computation of a tree that computes the nearest neighbors at each step increases the algorithm's execution time.

Tolerance level

There are several issues about the tolerance level choice. First, in the SMC version of ABC, a fixed sequence of tolerances $(\epsilon_m)_m$ is involved, but it is difficult to know in advance which sequence will be computationally efficient. One can set the next ϵ during the execution by considering the α-quantile of the simulation distances from y_{exp}, which gives the adaptive tolerance schedule:

$$\epsilon_{m+1} = quantile(\alpha, (\rho(\eta(y^{(i)}), \eta(y_{exp})))_{1 \leq i \leq N}).$$

In this case, a new hyperparameter α is involved. The difficulty is to choose α so that we do not introduce too many useless steps with a big α (close to 1), but also it should not be too small; otherwise, each step m will be too computationally expensive. Another method developed in (Del Moral, Doucet, and Jasra, 2012a) is to find the new ϵ based on the ESS statistic 7. In their SMC sampler, the weights $w_m^{(i)}$ depend on ϵ_m. Then, the idea is, for a given reduction factor α (typically 0.5 or 0.75),to find the next ϵ by solving the problem:

$$ESS((w_m^{(i)}(\epsilon_m))_{1 \leq i \leq N}) = \alpha ESS((w_{m-1}^{(i)}(\epsilon_{m-1}))_{1 \leq i \leq N}) \text{ over } \epsilon_m \in]0, \epsilon_{m-1}].$$

Thus, we control the particle degeneracy (Section 3.2.4) of our samples with the tolerance evolution.

Also, which final ϵ should we use? We know the smaller the tolerance is, the closer we are to the posterior distribution. A typical practical technique is to fix the running time or the number of simulation and reduce the ϵ as much as we can. However, to our knowledge, no other automatic method exists. Work about asymptotics ABC exists, but it is more related to the concentration of the ABC posterior to a true parameter, which can be seen as a particular case of the posterior distribution when $p(.|y_{exp}) = \delta_{\theta_{true}}(.)$ (Frazier et al., 2016; Li and Fearnhead, 2017; Robert, 2019).

3.4 Kernel Density Estimation

On the previous sections, we have described algorithms that sample from a target density π, but they do not provide a continuous approximation of π. The goal of kernel density estimation is to get an estimate $\widehat{\pi}$ with the help of kernel functions. Indeed, if the empirical distribution 3.2.1 is taken as an approximation, the density $\widehat{\pi}(x)$ equals zero in most cases, except if x is one of the samples. The key idea of this non-parametric method is that each observation will give mass to the density, i.e. we will construct a density where points near the observations will have positive density. This mass is given with a kernel function defined below.

Definition 3.4.1 (Kernel function)
A function $K : \mathbb{R} \to \mathbb{R}_{\geq 0}$ is a kernel function if:

1. $\int_{\mathbb{R}} K(u)du = 1$
2. $\forall u \in \mathbb{R}, K(u) = K(-u)$

This section does not intend to present all the existing methods of kernel density estimation. For an exhaustive exploration, we refer the reader to the books (Silverman, 1986) and (Chacón and Duong, 2018).

3.4.1 Kernel density estimator

We give below the definition of a kernel density estimator (Silverman, 1986).

Definition 3.4.2 (Kernel density estimator)
Let $X^{(1)}, \ldots, X^{(N)}$ be N i.i.d samples from an unknown density π on $\mathcal{X}$. The kernel density estimator $\widehat{\pi}$ associated with a kernel function K on $\mathcal{X}$ is:

$$\forall x \in \mathcal{X}, \widehat{\pi}_h(x) = \frac{1}{N} \sum_{i=1}^{N} K_h(x, X^{(i)})$$

K_h is a rescaled function based on kernel K. The scale factor h is called the bandwidth parameter. h is either a scalar, a vector or a matrix depending on the dimension of $\mathcal{X}$ and the choice of K.

In this estimator, each observation gives mass probability on the estimator over the whole set $\mathcal{X}$. The further from the observations, the lower the probability. Figure 3.2 illustrates this idea with a Gaussian kernel K.

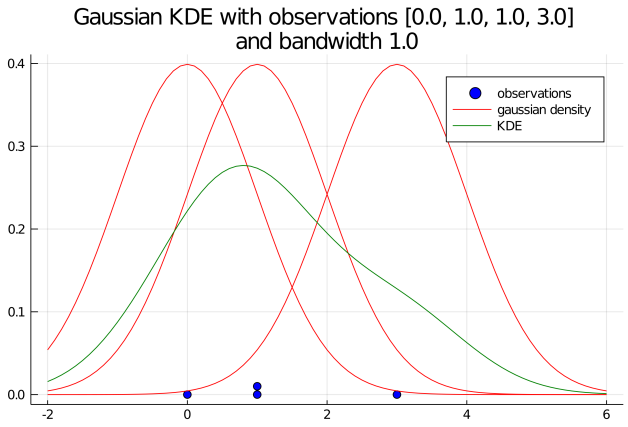


FIGURE 3.2: An illustration of Gaussian kernel density estimation.

This definition allows any form for K_h. In our work, we have focused on a specific type of kernel density estimator when the density is multidimensional. The *product kernel estimator* consists in considering the term $K_h(x, X^{(i)})$ as a product of univariate kernel functions for each dimension of $X^{(i)}$. It was first defined in (Bouezmarni and Rombouts, 2010).

Definition 3.4.3 (Product kernel multivariate density estimator)
Let $X^{(1)}, \dots, X^{(N)}$ N i.i.d samples from an unknown multidimensional density π on $\mathcal{X} \subset \mathbb{R}^d$. The s-th component of $X^{(i)}$ is denoted $X_s^{(i)}$. The product kernel density estimator $\widehat{\pi}$ associated with kernel functions $(K^s)_{1 \leq s \leq d}$ on $\mathcal{X}$ is:

$$\forall x = (x_1, \dots, x_d) \in \mathcal{X}, \widehat{\pi}(x) = \frac{1}{N} \sum_{i=1}^{N} \prod_{s=1}^{d} K_{h_s}^s(x_s, X_s^{(i)})$$

where K^s is a unidimensional kernel function, $K_{h_s}^s(x_s, X_s^{(i)}) = K^s(\frac{x_s - X_s^{(i)}}{h_s})$ and $h = (h_1, \dots, h_s) \in \mathbb{R}_{\geq 0}^d$ is the bandwidth vector.

This estimator is flexible and has convenient properties. (Bouezmarni and Rombouts, 2010) prove that it is free of boundary bias for specific kernels K_s, i.e. the bounds' bias is the same as for interior points of $\mathcal{X}$.

Several choices for the kernel function K are possible and depends on the nature of the estimation problem:

- When $\mathcal{X}$ is unbounded, one can preferably use Gaussian kernels 3.4.3,
- When $\mathcal{X}$ is bounded with positive probability on bounds, one can preferably use beta or gamma kernels 3.4.3.

Note that making a supposition on which category our problem belongs to is possible with an histogram visualisation.

3.4.2 Bandwidth selection and Least Squares Cross-Validation

Choosing h is critical to get a good estimator of the density. For the choice of the optimal bandwidth, we consider the minimisation of the mean integrated square error (Rosenblatt, 1956):

$$MISE(\widehat{\pi}_h) = \mathbb{E}_\pi[\int_{\mathbb{R}} (\widehat{\pi}_h(x) - \pi(x))^2 dx]$$

This statistic is helpful to prevent our estimator from the bias-variance trade-off. Analytical computations are often made with order-2 Taylor expansion of the MISE to find the optimal bandwidth that minimises it. For example, if one assumes π is a Gaussian density ($\pi \sim \mathcal{N}(\mu, \sigma)$), then the optimal bandwidth is $h_{opt} \approx 1.06\sigma n^{-1}$. However, in most cases, the analytical expression of the optimal bandwidth is based on the unknown density π, which makes the computation intractable.

Thus, a popular automatic estimation method of the optimal bandwidth is the minimisation of Least Squares Cross-Validation (LSCV). We want to estimate the part of the integrated square error that depends on the bandwidth parameter h.

$$R(\widehat{\pi}_h) = \int_{\mathcal{X}} \widehat{\pi}_h^2 - 2\int_{\mathcal{X}} \widehat{\pi}_h \pi$$

The first term is the squared L^2-norm of $\widehat{\pi}_h$. The second term is classically estimated by Leave-One-Out Cross-Validation (Silverman, 1986).

Definition 3.4.4 (Leave-one out estimator)
The Leave-One-Out estimator of $2\int \widehat{\pi}_h \pi$ *is:*

$$LOO(h) = \frac{2}{N} \sum_{i=1}^{N} \widehat{\pi}_h^{(-i)}(X_i)$$

where $\widehat{\pi}_h^{(-i)}(X_i) = \dfrac{1}{N-1} \displaystyle\sum_{j=1, j\neq i}^{N} K_h(X_j)$.

We can now define the LSCV estimator.

Definition 3.4.5 (Least Squares Cross-Validation)
The Least Squares Cross-Validation estimator is defined by:

$$LSCV(h) = \int_{\mathcal{X}} \widehat{\pi}_h^2 - LOO(h)$$

Thus, the difficulty in the computation of $LSCV(h)$ relies on the first term of the expression. In some cases, such as beta kernels 3.4.3, an analytical expression is not available, and the numerical estimation can be prohibitive depending on the number of observations and the complexity of the kernel. Other methods exist, such as likelihood cross-validation or the test-graph method (Silverman, 1986), but we do not address these techniques in our work.

The optimal bandwidth is chosen by solving:

$$\arg\min_h LSCV(h)$$

3.4.3 Kernel functions

As we said previously in the introduction, the choice of the kernel function depends on the nature of the problem, i.e. if $\mathcal{X}$ is bounded with positive probability on bounds or not.

Gaussian kernel

For the multivariate Gaussian kernel estimator, we do not use the product of univariate Gaussian kernels, but the multivariate Gaussian kernel. Let $X^{(1)}, \ldots, X^{(N)}$ N i.i.d samples from an unknown density π, $X^{(i)} = (X_1^{(i)}, \ldots, X_d^{(i)}) \in \mathcal{X}$.

$$\forall x \in \mathcal{X}, \widehat{\pi}_h(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\sqrt{(2\pi)^d \det(H)}} \exp(-\frac{1}{2}(x - X^{(i)})^\intercal H^{-1}(x - X^{(i)}))$$

where H is a $d \times d$ bandwidth matrix.

In this case, the LSCV estimator and its properties are well established (Duong and Hazelton, 2005) and tools such as the R package kernel smoothing (ks) (Duong et al., 2020) to perform the estimation.

Beta kernels

In our procedures, we will face density estimation with bounded support. In the following, we focus on the kernel density estimator using beta kernels. They proved their efficiency for estimating probability densities on bounded support when the density has positive probabilities on boundaries.

Thus, we have investigated methods for multivariate KDE with beta kernel and the associated analytical forms of LSCV for beta kernels, which are numerically estimated. This estimation has a non-negligible computation cost.

Univariate beta kernel estimator. In our work, we have to deal with densities that have to be estimated on bounded support, where the probability is possibly not zero on the bounds. To overcome this issue, (Chen, 1999) introduced beta kernel estimators.

Let $X^{(1)}, \dots, X^{(N)}$ be N i.i.d observations from an unknown density π over $\mathcal{X} = [a, b]$. We denote the scaled observations $\tilde{X}^{(1)}, \dots, \tilde{X}^{(N)}$, i.e. $\tilde{X}^{(i)} = \frac{X^{(i)}-a}{b-a}$. The univariate beta kernel estimator is defined as:

$$\forall x \in [a,b], \widehat{\pi}_{C1,h}(x) = \frac{1}{N(b-a)} \sum_{i=1}^{N} K^{\beta}_{\frac{x-a}{h(b-a)}+1, \frac{1-\frac{x-a}{b-a}}{h}+1}(\tilde{X}^{(i)})$$

The kernel function $K^{\beta}_{\alpha,\beta}(.)$ is the probability density function of a Beta law with parameters α, β, i.e. $K^{\beta}_{\alpha,\beta}(t) = \frac{t^{\alpha-1}(1-t)^{\beta-1}}{B(\alpha,\beta)}$ where B is the beta function.

In this case, one can express the ISE:

$$\begin{aligned}
\int_a^b \widehat{\pi}_{C1,h}(x)^2 dx &= \int_a^b \left(\frac{1}{N(b-a)}\right)^2 \left(\sum_{i=1}^{N} \frac{\tilde{X}^{(i)\,\frac{x-a}{h(b-a)}} (1-\tilde{X}^{(i)})^{\frac{1-\frac{x-a}{b-a}}{h}}}{B(\frac{x-a}{h(b-a)}+1, \frac{1-\frac{x-a}{b-a}}{h}+1)} \right)^2 dx \\
&= \left(\frac{1}{N(b-a)}\right)^2 \sum_{i=1}^{N}\sum_{j=1}^{N} \\
&\underbrace{\int_a^b \frac{\tilde{X}^{(i)\,\frac{x-a}{h(b-a)}} (1-\tilde{X}^{(i)})^{\frac{1-\frac{x-a}{b-a}}{h}} \tilde{X}^{(j)\,\frac{x-a}{h(b-a)}} (1-\tilde{X}^{(j)})^{\frac{1-\frac{x-a}{b-a}}{h}}}{B(\frac{x-a}{h(b-a)}+1, \frac{1-\frac{x-a}{b-a}}{h}+1)^2} dx}_{I_{(i,j)}}
\end{aligned}$$

With an adequate change of variables,

$$I_{(i,j)} = (b-a)\int_0^1 \frac{\tilde{X}^{(i)\frac{t}{h}}(1-\tilde{X}^{(i)})^{\frac{1-t}{h}}\tilde{X}^{(j)\frac{t}{h}}(1-\tilde{X}^{(j)})^{\frac{1-t}{h}}}{B(\frac{t}{h}+1,\frac{1-t}{h}+1)^2}dt.$$

Then,

$$\int_a^b \widehat{\pi}_{C1,h}(x)^2 dx = \frac{1}{N^2(b-a)}\left(\sum_{k=1}^N \int_0^1 \frac{\tilde{X}^{(k)\frac{2t}{h}}(1-\tilde{X}^{(k)})^{2\frac{1-t}{h}}}{B(\frac{t}{h}+1,1-\frac{t}{h}+1)^2}dt + 2\sum_{i=1}^N\sum_{j=i+1}^N \int_0^1 \frac{\tilde{X}^{(i)\frac{t}{h}}(1-\tilde{X}^{(i)})^{\frac{1-t}{h}}\tilde{X}^{(j)\frac{t}{h}}(1-\tilde{X}^{(j)})^{\frac{1-t}{h}}}{B(\frac{t}{h}+1,\frac{1-t}{h}+1)^2}dt\right)$$

In fine, we want to minimise:

$$LSCV(h) = \frac{1}{N^2(b-a)}\left(\sum_{k=1}^N \int_0^1 \frac{\tilde{X}^{(k)\frac{2t}{h}}(1-\tilde{X}^{(k)})^{2\frac{1-t}{h}}}{B(\frac{t}{h}+1,1-\frac{t}{h}+1)^2}dt + 2\sum_{i=1}^N\sum_{j=i+1}^N \int_0^1 \frac{\tilde{X}^{(i)\frac{t}{h}}(1-\tilde{X}^{(i)})^{\frac{1-t}{h}}\tilde{X}^{(j)\frac{t}{h}}(1-\tilde{X}^{(j)})^{\frac{1-t}{h}}}{B(\frac{t}{h}+1,\frac{1-t}{h}+1)^2}dt\right) - \frac{2}{N(b-a)}\sum_{i=1}^N \frac{1}{N-1}\sum_{j=1,j\neq i}^N K^{\beta}_{\frac{x-a}{h(b-a)}+1,\frac{1-\frac{x-a}{b-a}}{h}+1}(\tilde{X}^{(j)})$$

(Chen, 1999) proves that the optimal bandwidth is of order $O(N^{\frac{-2}{5}})$ as n grows to infinity.

Product kernel estimator and beta kernels. Let $X^{(1)},\ldots,X^{(N)}$ be N i.i.d samples from an unknown density π on the hyperrectangle $\mathcal{X} = [a_1,b_1]\times\ldots\times[a_d,b_d]$ with $X^{(i)} = (X_1^{(i)},\ldots,X_d^{(i)}) \in \mathbb{R}^d$. We denote the scaled observations $\tilde{X}^{(1)},\ldots,\tilde{X}^{(N)}$ with $\tilde{X}^{(i)}{}_s = \frac{X^{(i)}-a_s}{b_s-a_s}$. Then, one can use the product kernel estimator (Definition 3.4.3) for bounded kernel estimation on $\mathcal{X}$ with the beta kernel described in the previous section:

$$\widehat{\pi}_{C1,h}(x) = \frac{1}{N\cdot Vol(C)}\sum_{i=1}^N K^{\beta}_{\frac{x_s-a_s}{h_s(b_s-a_s)}+1,\frac{1-\frac{x_s-a_s}{b_s-a_s}}{h_s}+1}(\tilde{X}^{(i)}{}_s)$$

with $Vol(C) = \prod_{s=1}^d (b_s-a_s)$.

Thus, one can express the LCSV estimator as above.

$$LSCV(h) = \frac{1}{N^2 Vol(C)}\Big(\sum_{k=1}^{N}\int_0^1 \prod_{s=1}^{d} \frac{\tilde{X}^{(k)}{}_s^{\frac{2t}{h_s}}(1-\tilde{X}^{(k)}{}_s)^{2\frac{1-t}{h_s}}}{B(\frac{t}{h_s}+1, 1-\frac{t}{h_s}+1)^2}dt$$
$$+ 2\sum_{i=1}^{N}\sum_{j=i+1}^{N}\int_0^1 \prod_{s=1}^{d} \frac{\tilde{X}^{(i)}{}_s^{\frac{t}{h_s}}(1-\tilde{X}^{(i)}{}_s)^{\frac{1-t}{h_s}}\tilde{X}^{(j)}{}_s^{\frac{t}{h_s}}(1-\tilde{X}^{(j)}{}_s)^{\frac{1-t}{h_s}}}{B(\frac{t}{h_s}+1, \frac{1-t}{h_s}+1)^2}dt\Big)$$
$$- \frac{2}{N\cdot Vol(C)}\sum_{i=1}^{N}\frac{1}{N-1}\sum_{j=1, j\neq i}^{N} K^{\beta}_{\frac{x_s-a_s}{h_s(b_s-a_s)}+1, \frac{1-\frac{x_s-a_s}{b_s-a_s}}{h_s}+1}(\tilde{X}^{(j)}{}_s)$$

(Bouezmarni and Rombouts, 2010) proves that the optimal bandwidth is of order $O(N^{\frac{-2}{d+4}})$.

Remark 3.4.1 (Gamma kernels)
For estimation problems when $\mathcal{X}$ is partially bounded (e.g. $\mathcal{X} = [0, +\infty[$ in the univariate case), one can use gamma kernels ($K^{\gamma}_{k,\theta}(t) = \frac{t^{k-1}exp(\frac{-x}{\theta})}{\Gamma(k)\theta^k}$ where Γ is the gamma function) as proposed in (Chen, 2000; Bouezmarni and Rombouts, 2010). The methodology is the same as developed above.

Estimation of the bandwidth in practice. The estimation of the bandwidth parameter induces an optimisation problem:

$$\arg\min_h \ LSCV(h)$$

The bigger n, the more computationally expensive $LSCV(h)$ is. Thus, we consider the following gradient-free procedures for product kernel estimators:

- For the univariate beta kernel estimator, we consider Brent's method (Brent, 1971).
- For the multivariate beta kernel, we consider the simulated annealing algorithm (Goffe, Ferrier, and Rogers, 1994).
- If these methods are too computationally expensive, one can consider a grid search around the convergence rate of the bandwidth $O(N^{-\frac{2}{d+4}})$. This procedure is easily distributable.

Script 2 (Bounded Kernel Density Estimation in Julia language)
These procedures were implemented in Julia, accesible at the git repository `https://gitlab-research.centralesupelec.fr/2017bentrioum/boundedkde.jl` *with a*

focus on computation efficiency given that no implementation of product kernel estimators was available in Julia.

Remark 3.4.2 (π is supposed unknown)
In the whole section, we have described non-parametric methods to estimate π when the density is unknown. Suppose one has an intuition about the family of densities to which the unknown density π belongs. In that case, one can consider a parametrised family of probability density functions and fit the observations to reduce the complexity of the problem.

3.5 Summary

In this chapter, we described:

- The Bayesian framework and most common Monte Carlo methods that sample from an unknown density π when we can compute this density up to a constant.
- A family of likelihood-free methods called Approximate Bayesian Computation, which approximately samples from a posterior density, even if the likelihood is computationally intractable.
- Kernel density estimation methods for the estimation of an unknown density based on its samples, focusing on bounded support distributions.

Chapter 4

Verification of Continuous-Time Markov Chains

The world contains many complex systems such as communication networks, industries or transport that need to be safe, i.e. verify robust specifications. These systems may be modelled by deterministic or probabilistic models. Model checking (Baier and Katoen, 2008) groups several techniques for formal verification of models. This statement raises two questions:

- How a specification is formally described?
- Which class of models can be considered? How the formulation of model checking problems evolves along with the class of models?

In this Chapter, we focus on the verification of Continuous-Time Markov Chains. This is motivated by the growing interest in model checking in Systems Biology (Kwiatkowska, Norman, and Parker, 2008), which our subject study, Chemical Reaction Network models, belongs to. In particular, we focus on the statistical methods related to these verification tasks. Indeed, in the last two decades, statistical methods have emerged to balance the state-space explosion from which numerical methods of CTMC model checking suffer. These statistical methods are based on model simulations. The most classical one, called *statistical model checking*, is based on Monte Carlo simulations. Other statistical methods exist, and we propose a new one for parametric CTMCs in Chapter 6.

More precisely, the goals of the Chapter are:

- To describe the model checking formalism of a CTMC given a specification φ.
- To define *model checking of parametric CTMC*, which is model checking on a collection of parametrised CTMCs.

- To give an overview of statistical methods related to model checking.
- To detail the HASL formalism, which is well-suited for CTMC verification.

The Chapter is organised as follows. Section 4.1 describes temporal logics to formally define model specifications. Section 4.2 describes the verification tasks for a CTMC and the related statistical methods. Section 4.3 describes the verification tasks for a collection of CTMCs and the related statistical methods. Section 4.4 describes the HASL formalism based on Linear Hybrid Automata.

4.1 Temporal logic

To verify the specifications of our systems, we need a logical formalism to describe them. In this section, we describe the syntax and the satisfaction relation of two temporal logics: Metric-Interval Temporal Logic (MITL) (Alur, Feder, and Henzinger, 1991), which is based on the Linear Temporal Logic (LTL) and does not involve probabilistic operator; and CSL (Aziz, 2000; Baier et al., 2003), which is based on the Computation Tree Logic (CTL).

In the following, we consider a CTMC $\mathcal{M} = (\mathbf{S}, \alpha, Q)$ and its trajectory set $Path(\mathcal{M})$.

4.1.1 MITL

Definition 4.1.1 (Metric Interval Temporal Logic)
An MITL formula φ is a term of the following grammar:

$$\varphi \ ::= \ \mathsf{true} \mid \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathbf{U}^I \varphi_2$$

where true *stands for the true formula, $\mu \in AP$ is an atomic proposition, $\neg$ and $\wedge$ are the negation and conjunction operators of propositional logic, and $\mathbf{U}^I$ is the until temporal operator with $I \subseteq \mathbb{R}_{\geq 0}$ a non-singular interval.*

MITL formulae do not involve any *probabilistic operator*. The truth of an MITL formula for a path $\sigma \in Path(\mathcal{M})$ is expressed through a *satisfaction* relationship denoted $\models$ and defined below (see Section 2.2.1 of Chapter 2 for the notations of a CTMC path).

Definition 4.1.2 (Satisfaction relation $\models$ of MITL)
The satisfaction relation $\models$ of an MITL temporal formula is defined for a path $\sigma \in Path(\mathcal{M})$ as follows:

$$
\begin{array}{lcl}
\sigma \models \mathsf{true} & & \\
\sigma \models \mu & \Leftrightarrow & \sigma[0] \models \mu \\
\sigma \models \neg\varphi & \Leftrightarrow & \sigma \not\models \varphi \\
\sigma \models \varphi_1 \wedge \varphi_2 & \Leftrightarrow & \sigma \models \varphi_1 \text{ and } \sigma \models \varphi_2 \\
\sigma \models \varphi_1 \mathbf{U}^I \varphi_2 & \Leftrightarrow & \exists t \in I, \\
 & & \sigma[t\rangle \models \varphi_2 \text{ and } \forall t' < t, \sigma[t'\rangle \models \varphi_1
\end{array}
$$

For $\mathbf{s}$ a state of the CTMC $\mathcal{M}$, $\mathbf{s} \models \varphi$ reads: *state* $\mathbf{s}$ *satisfies* φ. Although MITL semantics is defined for paths of a CTMC the truth of an MITL formula is naturally extended to states of a CTMC by considering the set of paths originating from a given path (see the second row of Definition 4.1.2).

For example, as $\sigma[t\rangle \models \mu \Leftrightarrow \sigma@t \models \mu$, a time-bounded until formula $\varphi_1 \ \mathbf{U}^{[t_1,t_2]} \ \varphi_2$ is satisfied at time t for a path σ if and only if there exists $t' \in [t_1, t_2]$ such that $\sigma@(t+t') \models \varphi_2$ and $\forall t'' < t', \sigma@(t+t'') \models \varphi_1$, which is expressed as $\sigma@t \models \varphi_1 \ \mathbf{U}^{[t_1,t_2]} \ \varphi_2$. $\sigma \models \varphi$ means $\sigma@0 \models \varphi_1 \ \mathbf{U}^{[t_1,t_2]} \ \varphi_2$ as stated in Definition 4.1.2.

4.1.2 CSL

Continuous Stochastic Logic (CSL) (Aziz, 2000; Baier et al., 2003) is a logic derived from CTL and adapted to Continuous-Time Markov Chains.

Definition 4.1.3 (Continuous Signal Logic)
A CSL formula ϕ is described by the following grammar:

$$
\begin{array}{l}
\phi ::= \mathsf{true} \mid \mu \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid P_{\bowtie p}\varphi \mid S_{\bowtie p}\phi \\
\varphi ::= \phi \mathbf{U}^I \phi \mid \mathbf{X}^I \phi
\end{array}
$$

where $\bowtie \in <, \leq, \geq, >$ and $I \subseteq \mathbb{R}_{\geq 0}$ is a non-singular interval.

ϕ is a *state formula*, whereas φ is a *path formula*. For path formulae, CSL has an Until operator $\mathbf{U}$, and a next operator $\mathbf{X}$ that asserts if the next state of the path within the interval satisfies the property.

CSL includes two operators: the *probability operator* P and the *steady-state operator* S (also called long run operator). $P_{\bowtie p}\varphi$ means the probability that a path verifies φ is $\bowtie p$. $S_{\bowtie p}\phi$ means the steady-state probability that ϕ holds is $\bowtie p$.

The two types of formula lead to the definition of two satisfaction relations, one for state formulae and another for path formulae.

Definition 4.1.4 (Satisfaction relation $\models$ for a CSL state formula ϕ)
The satisfaction relation $\models$ over state formulae is defined as:

$$\begin{array}{lcl}
\mathbf{s} \models \mathsf{true} & & \\
\mathbf{s} \models \neg\phi & \Leftrightarrow & \mathbf{s} \not\models \phi \\
\mathbf{s} \models \phi_1 \wedge \phi_2 & \Leftrightarrow & \mathbf{s} \models \phi_1 \wedge \mathbf{s} \models \phi_2 \\
\mathbf{s} \models P_{\bowtie p}\varphi & \Leftrightarrow & Pr_{\mathcal{M}_0}(\{\sigma \in Path(\mathcal{M}_0), \sigma \models \varphi\}) \bowtie p \\
 & & \textit{with } \mathcal{M}_0 = (\mathbf{S}, \delta_{\mathbf{s}}, Q, AP, L) \\
\mathbf{s} \models S_{\bowtie p}\phi & \Leftrightarrow & \lim\limits_{t \to +\infty} Pr_{\mathcal{M}}(\{\sigma \in Path(\mathcal{M}), \sigma@t \models \phi\}) \bowtie p
\end{array}$$

$Pr_{\mathcal{M}_0}(\{\sigma \in Path(\mathcal{M}_0), \sigma \models \phi\})$ is the probability that $\sigma \models \phi$ when the CTMC $\mathcal{M}$ begins at state $\mathbf{s}$.

Remark 4.1.1
As detailed in the Remark 2.2.2 of Chapter 2, $Pr_{\mathcal{M}}$ is often denoted Pr_{α} in the literature. With this notation, $Pr_{\mathcal{M}_0}(\{\sigma \in Path(\mathcal{M}_0), \sigma \models \phi\}) = Pr_{\mathbf{s}}(\{\sigma \in Path(\mathbf{s}), \sigma \models \phi\})$.

Definition 4.1.5 (Satisfaction relation $\models$ for a CSL path formula φ)
The satisfaction relation $\models$ over path formulae is defined as:

$$\begin{array}{lcl}
\sigma \models \mathbf{X}^I\phi & \Leftrightarrow & \sigma[1] \models \phi \wedge \delta(\sigma, 0) \in I \\
\sigma \models \phi_1 \mathbf{U}^I \phi_2 & \Leftrightarrow & \exists t \in I, \sigma@t \models \phi_2 \wedge \forall t' < t, \sigma@t' \models \phi_1
\end{array}$$

4.1.3 Eventually and global operators

The Until operator is less convenient if one wants to express properties like "the variable X reaches x_1 between I" or "the variable X is contained in $[x_1, x_2]$ during I". Then, we consider two derivations of the Until operator: the *eventually* $\mathbf{F}^I\varphi = \mathsf{true}\ \mathbf{U}^I\varphi$, which stands for *"at some point within I φ is satisfied"*, and the *globally* $\mathbf{G}^I\varphi = \neg\mathbf{F}^I\neg\varphi$ which stands for *"φ is always satisfied within I"*.

4.2 Model checking of Continuous-Time Markov Chains

Now that we have defined how to describe specifications for CTMCs with temporal logic, we detail how we verify these specifications. Considering a specification ϕ, we

want to know if a CTMC $\mathcal{M}$ fulfils ϕ. It relies on the computation of the probability that a path formula φ is satisfied.

Definition 4.2.1 (Satisfaction probability of a path formula)
Let $\mathcal{M}$ be a CTMC and φ be an MITL formula. The probability that $\mathcal{M}$ satisfies φ defined by

$$Pr(\varphi; \mathcal{M}) = Pr_{\mathcal{M}}(\{\sigma \in Path(\mathcal{M}), \sigma \models \varphi\})$$

is called the satisfaction probability of φ.

Two kinds of problems related to model checking are distinguishable.

Definition 4.2.2 (Estimation problem in model checking for CTMC)
Given an MITL formula φ and a CTMC $\mathcal{M}$, the estimation problem in model checking for CTMC is defined as:

$$\text{Compute } Pr(\varphi; \mathcal{M})$$

Definition 4.2.3 (Threshold problem in model checking for CTMC)
Given a CSL formula $\phi = P_{\bowtie p}\varphi$ with φ a path formula, and a CTMC $\mathcal{M}$, the threshold problem in model checking for CTMC is defined as:

$$\text{Verify } \mathcal{M} \models \phi, \textit{ i.e. } Pr(\varphi; \mathcal{M}) \bowtie p.$$

Remark 4.2.1 (Qualitative/quantitative approach of Probabilistic Model Checking)
In the model checking literature, the threshold problem is sometimes referred to as the qualitative approach; whereas the estimation problem is sometimes referred to as the quantitative approach of model checking.

4.2.1 About numerical methods

Numerical methods are exact verification methods that exhaustively compute the probabilities contained in a CSL formula ϕ over the whole CTMC state space. It implies the evaluation of sub-formulae over the states of the CTMC to label the states with the entire formula ϕ. For example, if $\phi = \neg\phi_2$, the verification is done by labelling with ϕ the states that are not labelled with ϕ_2.

For example, in the case of a CSL formula $\phi = P_{\bowtie p}\varphi$ with φ a time-bounded Until formula, the verification of ϕ implies *transient analysis*, i.e. the computation of the transient probabilities described by Kolmogorov's Forward Equations 2.1.5.

As we have seen before, these methods can suffer from the *state-space explosion*, adding that in practice, the required memory to run these algorithms is high and the parallelism difficult. Several classical improvements exist for faster model checking as the aggregation of states (reduction of the state space) or uniformisation of CTMC (Baier et al., 2003), which we will not detail here.

Numerical methods for model checking of CTMC are available in the Prism model checker software (Kwiatkowska, Norman, and Parker, 2011).

4.2.2 Statistical model checking

Estimation problem - Confidence bounds

We consider the estimation problem in Definition 4.2.2. We want to estimate $p = Pr(\varphi; \mathcal{M})$. Simulating $\sigma \sim \mathcal{M}$ and verifying if $\sigma \models \varphi$ can be seen as a draw from a Bernoulli random variable $X \sim \mathcal{B}(p)$. Indeed, there is a probability p that $\sigma \models \varphi$. Statistical model checking aims at estimating p with statistical confidence: we want to know how many simulation runs we need to have a certain level of statistical confidence. This task is done by the computation of a confidence interval.

Definition 4.2.4
Let X be a random variable and p a parameter to estimate. A confidence interval of level $(1-\alpha)$ is a random interval [b(X), u(X)] with

$$\mathbb{P}(b(X) \leq p \leq u(X)) \geq 1 - \alpha.$$

For our Bernoulli variable $X \sim \mathcal{B}(p)$, $\mathbb{E}[X] = p$. A natural estimator is $\overline{X}_N = \frac{X^{(1)}+\ldots+X^{(N)}}{N}$ which converges to p.

The Hoeffding inequality gives for any $c \in \mathbb{R}_{\geq 0}$

$$\mathbb{P}(|\overline{X}_N - X| \geq c) \leq 2\exp(-2c^2N)$$
$$\mathbb{P}(|\overline{X}_N - X| \leq c) \geq 1 - 2\exp(-2c^2N)$$

By setting c so that $2\exp(-2c^2N) = \alpha$, we obtain the following confidence interval:

$$\mathbb{P}\left(\overline{X}_N - \sqrt{\frac{-\log(\alpha/2)}{2N}} \leq p \leq \overline{X}_N + \sqrt{\frac{-\log(\alpha/2)}{2N}}\right) \geq 1 - \alpha.$$

With this equality, one can set the number N of simulations according to the radius $r = \sqrt{\frac{-\log(\alpha/2)}{2N}}$ of the desired confidence interval. Dividing r by two leads to run four times more simulations. In (Jegourel, Sun, and Dong, 2019), they use Massart bounds to sharpen the bounds of the confidence interval.

When p is tiny (less than 10^{-4}), Statistical Model Checking is combined with a specific field of statistics called *rare event analysis*. It deals with methods such as subset simulation and important sampling mentioned in Chapter 3 (e.g. see (Barbot, 2014)). For a complete review of Statistical Model Checking methods, we refer the reader to (Legay, Delahaye, and Bensalem, 2010; Legay et al., 2019).

Threshold problem - Hypothesis testing

We consider a CSL formula $\phi = P_{\leq p}\varphi$ and the threshold problem of Definition 4.2.3. If numerical methods are too computationally expensive, a method to statistically verify this specification is hypothesis testing.

The principle of hypothesis testing is to formulate two disjoint hypotheses and test their validity based on observation data $X^{(1)}, \ldots, X^{(N)}$. The first hypothesis H_0, called the null hypothesis, is a default position and H_1 the alternative one. Then, we compute a statistic $T(X^{(1)}, \ldots, X^{(N)})$, for which we can derive the exact or asymptotic distribution under the null hypothesis. If the realisation of this statistic with our data is too unlikely compared to the distribution under the null hypothesis, the null hypothesis is rejected.

For statistical model checking, the most used statistic is Wald's Sequential Probability Ratio Test. We define the two hypotheses as:

- $H_0 : p \geq p_0$
- $H_1 : p \leq p_1$

Let $x^{(1)}, \ldots, x^{(N)}$ be a realisation of $X^{(1)}, \ldots, X^{(N)} \sim \mathcal{B}(p)$. Then, the test statistic is the quantity

$$t = \frac{p_{1,N}}{p_{0,N}} = \prod_{i=1}^{N} \frac{\mathbb{P}(X^{(i)} = x^{(i)} | p = p_1)}{\mathbb{P}(X^{(i)} = x^{(i)} | p = p_0)} = \frac{p_1^{d_N}(1-p_1)^{N-d_N}}{p_0^{d_N}(1-p_0)^{B-d_N}}$$

where $d_N = \sum_{i=1}^{N} x^{(i)}$. t is computable sequentially, which is convienent to obtain a decision without knowing the sufficient number of simulations in advance.

Let α be the probability of the type I error (rejection of the true null hypothesis) and β be the probability of the type II error (not rejecting while H_0 is false). Then, the decision process is:

1. If $t \leq \frac{\beta}{1-\alpha}$, accept H_0.
2. Else if $t \geq \frac{1-\beta}{\alpha}$, accept H_1.
3. Else, no conclusion.

4.3 Model checking of parametric Continuous-Time Markov Chains

The previous section's methods deal with an instance of a CTMC. In some cases, a vector θ naturally parametrises CTMCs. CTMCs induced by Chemical Reaction Networks (Section 2.3) are an illustrative example: they are parametrised by reaction rate constants. In this section, we present verification methods for a collection of CTMCs indexed by a parameter. These collections are called *parametric Continuous-Time Markov Chains*, for which we give a definition below.

Definition 4.3.1 (Parametric CTMC)
Let Θ be a non-empty set. A parametric CTMC, abbreviated pCTMC, $(\mathcal{M}_\theta)_{\theta\in\Theta}$ (or $(S^\theta)_{\theta\in\Theta}$) is a collection of CTMCs indexed by a parameter vector $\theta \in \Theta$.

The general goal of these methods is to analyse how the satisfaction probability of temporal properties evolve along with the parameters.

4.3.1 Estimation problem - Satisfaction function regression

We extend the verification task described in Definition 4.2.2. We no longer want to estimate a value $Pr(\varphi; \mathcal{M})$ for a specific value θ but rather a set of values $(Pr(\varphi; \mathcal{M}_\theta))_{\theta\in\Theta}$.

Definition 4.3.2 (Satisfaction probability function)
Let $(\mathcal{M}_\theta)_{\theta\in\Theta}$ be a parametric CTMC and φ an MITL formula. The function

$$\begin{aligned} f_\varphi :&\Theta \to [0,1] \\ &\theta \to Pr(\varphi; \mathcal{M}_\theta) \end{aligned}$$

is called the satisfaction probability function.

The satisfaction probability function (Definition 4.3.2) expresses how the satisfaction probability of φ varies with the parameters. Then, the estimation problem for parametric model checking is to *estimate the satisfaction probability function.*

If Θ is finite, then the task can be tackled by iterating numerical or statistical methods of classical model checking for each parameter. If Θ is infinite, numerical methods are impossible (it requires an infinite computation time). The problem becomes a task of function regression. An exact method means the analytical computation of transient probabilities is possible for every parameter $\theta \in \Theta$. In most cases, it is impossible (we give an example that is simple enough to have an analytical solution in Section 6.5.1). Such difficulties encourage statistical methods.

Remark 4.3.1 (Automaton-ABC method)
Chapter 6 details a new method for the estimation of the satisfaction probability function based on ABC and Linear Hybrid Automata.

Statistical methods

In (Ceška et al., 2014), they formulate the *max synthesis*, aiming to estimate the regions of Θ where the satisfaction probability function 4.3.2 is maximal, with confidence bounds. A significant work about the estimation of the satisfaction probability function for parametric CTMC 4.3.1 has been done in (Bortolussi, Milios, and Sanguinetti, 2016). The method is called *Smoothed Model Checking*. Smoothed MC algorithm estimates the satisfaction probability function in a Bayesian framework with the help of Gaussian Processes (GP).

Let us define the method's framework based on Bayesian statistics. We want to estimate the function f_φ based on random observations denoted $O^{(1)}, \ldots, O^{(n)}$ whose realisations are $o^{(1)}, \ldots, o^{(n)}$ (the meaning of these observations is detailed below). Smoothed model checking is a Bayesian learning regression method. We have to define

1. The *prior* f,
2. The *observation* model.

Smoothed MC estimates f_φ by estimating the *posterior* distribution:

$$p(f(\theta*)|\ O^{(1)} = o^{(1)}, \ldots, O^{(n)} = o^{(n)}), \theta* \in \Theta.$$

In function regression, defining a prior amounts to define a stochastic process (also called random function) $(f(\theta))_{\theta\in\Theta}$. It is analogous to the prior definition in statistical parameter inference when one defines a random variable over the parameters space. In the Smoothed MC method, the prior is set as a Gaussian process defined below.

Definition 4.3.3 (Gaussian process)
A Gaussian process is a stochastic process $(Y_x)_{x\in\mathcal{X}}$ whose finite dimensional distributions are Gaussian, i.e.

$$\forall n \in \mathbb{N}, \forall x_1, \dots, x_n \in \mathcal{X}, (Y_{x_1}, \dots, Y_{x_n}) \sim \mathcal{N}(\mu, \Sigma)$$

It is uniquely defined by its mean function $\mu(x) = \mathbb{E}[Y_x]$, and its covariance function $k(x, x') = Cov(Y_x, Y_{x'})$. We can write $Y \sim GP(m(.), k(.,.))$.

A Gaussian process is entirely described by its mean function and covariance function. The mean function is typically constant (either zero or the mean of the dataset). The chosen covariance function in Smoothed MC is the Gaussian Radial Basis function $k(x, x') = \sigma^2 e^{-\|x-x'\|^2/\lambda^2}$.

Second, we consider n observations $o^{(1)}, \dots, o^{(n)}$ related to parameters $\theta^{(1)}, \dots, \theta^{(n)}$. Each $o^{(i)}$ is the number of trajectories that verifies φ among the simulation of M trajectories with the CTMC $\mathcal{M}_{\theta^{(i)}}$ (as it can be done in statistical model checking, Section 4.2.2). Thus, $o^{(i)}$ is the result of M independent drawing from a Bernoulli law $\mathcal{B}(f_\varphi(\theta^{(i)}))$, it is a Binomial law $Bin(M, f_\varphi(\theta^{(i)}))$.

$$O^{(i)}|\ f(\theta^{(i)}) \sim Bin(M, f(\theta^{(i)}))$$

Thanks to Bayes rule,

$$p(f(\theta*), f(\theta^{(1)}), \dots, f(\theta^{(n)})|\ o^{(1)}, \dots, o^{(n)}) \propto \underbrace{p(f(\theta*), f(\theta^{(1)}), \dots, f(\theta^{(n)}))}_{\text{Gaussian process}} \cdot \prod_{i=1}^{n} \underbrace{p(o^{(i)}|\ f(\theta^{(i)}))}_{Bin(M, f(\theta^{(i)}))}$$

Then, the prediction is performed by the marginalisation of the density written below.

$$p(f(\theta*)|\ o^{(1)}, \dots, o^{(n)}) = \int_{\Theta^n} p(f(\theta*), f(\theta^{(1)}), \dots, f(\theta^{(n)})|\ o^{(1)}, \dots, o^{(n)}) df(\theta^{(1)}) \dots df(\theta^{(n)}) \tag{4.1}$$

which is analytically intractable. Smoothed MC uses the Expectation-Propagation algorithm (Minka, 2013) to compute an approximate posterior and control the error.

To have a good estimation over Θ, observations on the dataset are added iteratively (i.e. drawing a new θ^* and perform statistical model checking over $\mathcal{M}_{\theta^*}$) until a convergence criterion is met (about the estimated error).

Remark 4.3.2 (Smoothed MC and Gaussian Process regression)
As the matter of fact, Smoothed MC is not *Gaussian Process regression. In classical GP regression, we consider observations* $(x_i, y_i)_{1:N}$ *where* $y_i \sim Y_{x_i}$ *and* $Y_{x_{1:N}} = (Y_{x_1}, \dots, Y_{x_N})^t$ *as the random sample. Let* $x^* \in \mathcal{X}$*. We want to address the prediction task, i.e. estimate* $f(x^*)$*. This task is done by identifying the conditional law* $Y_{x^*}|Y_{x_{1:N}}$*. As* Y *is a Gaussian process, we know* $Y_{x^*}, Y_{x_1}, \dots, Y_{x_N}$ *is a Gaussian vector:*

$$\begin{pmatrix} Y_{x_{1:N}} \\ Y_{x^*} \end{pmatrix} \sim \mathcal{N}\left(0, \left[\begin{array}{c|c} (k(Y_{x_i}, Y_{x_j}))_{1:N} & \begin{matrix} k(Y_{x_1}, Y_{x^*}) \\ \vdots \\ k(Y_{x_N}, Y_{x^*}) \end{matrix} \\ \hline \begin{matrix} k(Y_{x^*}, Y_{x_1}) & \dots k(Y_{x^*}, Y_{x_N}) \end{matrix} & k(Y_{x^*}, Y_{x^*}) \end{array}\right]\right)$$

Thus, the conditional law $Y_{x^*}|Y_{x_{1:N}}$ *is easily computable with Gaussian identities, and the expectation has a closed form.*

4.3.2 Parameter synthesis - threshold problem

Parameter synthesis aims at partitioning the parameter space Θ with a CSL formula ϕ. We want to estimate two subsets of parameters Θ_ϕ and $\Theta_{\neg\phi}$, such that Θ_ϕ contains the parameters for which $\mathcal{M}_\theta$ satisfies the specifications, and conversely for $\Theta_{\neg\phi}$.

- $\Theta_\phi = \{\theta \in \Theta, \mathcal{M}_\theta \models \phi\}$
- $\Theta_{\neg\phi} = \{\theta \in \Theta, \mathcal{M}_\theta \not\models \phi\}$

which can be seen as a classification task. In practice, it is difficult to solve this problem exactly. In the case of an infinite parameter space Θ, it would require that transient analysis can be solved analytically. Thus, the *parameter synthesis* (also called *threshold synthesis*) is formulated as follows.

Let ϕ be a CSL formula let $\epsilon > 0$ be a tolerance error. We want to find a partition (T, F, U) of Θ s.t.

- $T \subseteq \Theta_\phi$
- $F \subseteq \Theta_{\neg\phi}$
- $U \subset \Theta$ with $Volume(U)/Volume(\Theta) \leq \epsilon$

Several numerical methods have been published to solve this problem. (Han, Katoen, and Mereacre, 2008) address an approximate solution by discretising Θ for time-bounded reachability probabilities. In (Ceška et al., 2014), they solve the problem by computing the bounds of transient probabilities of (Brim, Ceška, and Dražan, 2013).

Statistical formulation

If numerical methods are too computationally expensive, we can formulate a statistical version of threshold synthesis.

Let $\phi = P_{\geq p}\varphi$ with φ a path formula. Considering an estimator $\widehat{f_\varphi}$ (a random function) of the true satisfaction probability function f_φ, a tolerance ϵ, a confidence threshold α, we want to find a partition $(\hat{T}, \hat{F}, \hat{U})$ of Θ s.t.

- $\hat{T} = \{\theta \in \Theta, \mathbb{P}(\widehat{f_\varphi}(\theta) \geq p) \geq 1 - \alpha\}$
- $\hat{F} = \{\theta \in \Theta, \mathbb{P}(\widehat{f_\varphi}(\theta) < p) \geq 1 - \alpha\}$
- $\hat{U} \subset \Theta$ with $Volume(U)/Volume(\Theta) \leq \epsilon$

To our knowledge, a statistical approach of parameter synthesis has only been tackled by (Bortolussi and Silvetti, 2018), which uses the Smoothed MC estimator of the function to compute statistical bounds. In (Molyneux and Abate, 2020), they use Support-Vector Machines to approximate the partition of Θ according to the CSL formula. Indeed, parameter synthesis is a classification task. Thus any supervised learning method is worth considering.

4.4 Hybrid Automata Stochastic Logic

Hybrid Automata Stochastic Logic (HASL) is a logic (Ballarini et al., 2011) designed for Discrete-Event Stochastic Processes (DESP), including CTMCs. It is a quantitative logic that allows operators to compute real-valued indicators like expectation or maximum. This logic is more expressive than CSL for non-nested formulae (Ballarini

et al., 2011, Section 3.3). This quantitative logic inherently adopts the statistical point of view of verification, and is based on model simulations, which makes it well-suited for statistical model checking of continuous-time models with discrete events.

An HASL formula is described by a couple $(Z, \mathcal{A})$, where $\mathcal{A}$ is a linear hybrid automaton $\mathcal{A}$, and Z is an HASL expression.

4.4.1 Stochastic Petri Net

The implementation of HASL (see Section 4.4.4 further) is based on Generalised Stochastic Petri Nets (GSPN) formalism. Chemical Reaction Network models can be described by Stochastic Petri Nets (SPN), which is a special case of GSPNs.

Definition 4.4.1 (Stochastic Petri Net)
A Stochastic Petri Net is a tuple $SPN = (P, T, F, M_0, \Lambda)$ *where*

- $P = \{p_1, \ldots, p_d\}$ *is a set of places.*
- $T = \{t_1, \ldots, t_M\}$ *is a set of transitions.*
- $W : (P \times T) \cup (T \times P) \to \mathbb{N}$ *is a function that assigns a multiplicity to each possible arc.*
- $m_0 \in \mathbb{N}^d$ *is the initial marking, which represents the initial distribution of tokens over the places.*
- $\Lambda = \{\lambda_1(.), \ldots, \lambda_M(.)\}$ *is a vector of firing rates that may depend on the marking of the SPN.*

An SPN is a bipartite graph. A place p can only be connected to a transition t and vice versa, which is represented by $W(p,t) > 0$ (or $W(t,p) > 0$). A marking m is a vector of dimension d. It assigns an integer to each place. Each integer represents the number of *tokens* contained in a place. A firing rate $\lambda(m)$ is a transition rate of some transition t_i, i.e. the firing time of t_i is distributed as $Exp(\lambda(m))$.

An SPN easily describes a Chemical Reaction Network (Section 2.3, Chapter 2). Indeed, the transitions are the chemical reactions, i.e. $T = \{R_1, \ldots, R_M\}$. Each place represents a species, the number of tokens contained in a place is the population of species, and a marking m is an element $\mathbf{s} \in \mathbf{S}$.

For a marking m, the stochastic dynamics of an SPN are composed of two steps:

- *Enabling* of transitions. A transition t_i is enabled if all its input places p (which means $W(p, t_i) > 0$) have a number of tokens greater than $W(p, t_i)$.
- *Firing* of a transition. A enabled transition is fired if it has the minimum sojourn time within the enabled transitions. The sojourn times are exponentially distributed with mean $\frac{1}{\lambda_i(m)}$. This is analogous to the SSA (Algorithm 1) over the enabled transitions.

Let us illustrate this with the SIR Chemical Reaction Network whose equations are recalled below:

$$R_1 : S + I \xrightarrow{k_i} 2I$$
$$R_2 : I \xrightarrow{k_r} R$$

Figure 4.1 illustrates the SIR CRN with the SPN formalism.

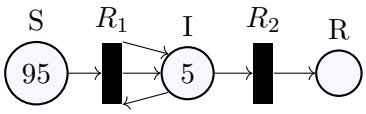


FIGURE 4.1: SIR Stochastic Petri Net with initial marking $m_0 = (95, 5, 0)$

This SPN has two transitions R_1 and R_2. The place S has 95 tokens, I has 5 tokens, whereas no tokens are in R. S and I have at least one token, which enables R_1. Also, I has at least one token, which enables R_2. If R_1 occurs, one token is moved from the place S ($W(S, R_1) = 1$), whereas the place I receives a token. ($W(R_1, I) - W(I, R_1) = 1$). The marking is updated to $m_1 = (94, 6, 0)$.

Remark 4.4.1 (Generalised Stochastic Petri Nets)
GSPNs are SPNs that also include immediate transitions. The analogous stochastic processes are Markov Renewal Processes. We refer the interested readers to (Bause and Kritzinger, 2013) for a deeper exploration of Petri Net models and their timed extensions.

4.4.2 Linear Hybrid Automata

Definition

Linear Hybrid Automata (LHA) are a class of automata that extends Deterministic Timed Automata (DTA). They are equipped with clocks and can deal with continuous variables as well as discrete changes.

Definition 4.4.2 (Linear Hybrid Automaton)
A Linear Hybrid Automaton (LHA) is a tuple $\mathcal{A} = (E, L, \Lambda, Init, Final, X, Flow, \rightarrow)$ *where*

- E *is a finite alphabet of events.*
- $L = \{l_0, \ldots, l_q\}$ *is a finite set of locations.*
- $\Lambda : L \rightarrow Prop$ *is a location labelling function.*
- $Init \subset L$ *is the set of initial locations.*
- $Final \subset L$ *is the set of final locations.*
- $X = (x_1, \ldots, x_n)$ *is a n-tuple of data variables.*
- $Flow : L \rightarrow Ind^n$ *is a function which associates each location with n indicator functions (one per data variable).* $Flow_i$ *is the i-th projection.*
- *The transition relation (the set of edges)* $\rightarrow$*, which is a subset of* $L \times \big((2^E \times \mathsf{Const}) \uplus (\{\sharp\} \times \mathsf{IConst})\big) \times Up \times L$ *where*
  - $\sharp$ *means the transition is asynchronous, i.e. the transition can be fired even if it is not synchronised with an event of the stochastic model,*
  - Const *is a set of constraints,*
  - IConst *is a set of left-closed linear constraints.*

A transition $(l, (E, \gamma), U, l') \in \rightarrow$ is written $l \xrightarrow{E,\gamma,U} l'$.

This definition is coupled with additional properties.

- **Initial determinism**: $\forall l, l' \in Init$ with $l \neq l'$, $\Lambda(l) \wedge \Lambda(l') \Leftrightarrow \mathsf{false}$.
- **Determinism on events**: $\forall E, E' \subseteq E$ with $E \cap E' \neq \emptyset, \forall l_0, l, l' \in L$, if $l_0 \xrightarrow{E,\gamma,U} l$ and $l_0 \xrightarrow{E',\gamma',U'} l'$ with $(l, E, \gamma, U) \neq (l', E', \gamma', U')$, then either $\Lambda(l) \wedge \Lambda(l') \Leftrightarrow \mathsf{false}$ or $\gamma \wedge \gamma' \Leftrightarrow \mathsf{false}$. In other terms, if we are in the location l_0 and an event $E_0 \in E \cap E'$ occurs, then at most one transition can be fired.
- **Determinism on asynchronous transitions** $\sharp$: $\forall l_0, l, l' \in L$, if $l_0 \xrightarrow{\sharp,\gamma,U} l$ and $l_0 \xrightarrow{\sharp,\gamma',U'} l'$ with $(l, \gamma, U) \neq (l', \gamma', U')$, then either $\Lambda(l) \wedge \Lambda(l') \Leftrightarrow \mathsf{false}$ or $\gamma \wedge \gamma' \Leftrightarrow \mathsf{false}$. In other terms, if we are in the location l_0, at most one asynchronous transition can be fired.
- **No $\sharp$-labelled loops**: For all $l_0 \xrightarrow{E_0,\gamma_0,U_0} l_1 \xrightarrow{E_1,\gamma_1,U_1} \ldots \xrightarrow{E_{p-1},\gamma_{p-1},U_{p-1}}$, there exists $i \leq p$ so that $E_i \neq \sharp$. It means the automaton cannot loop infinitely with asynchronous transitions.

Remark 4.4.2 (Events for Chemical Reaction Network)
In the case of Chemical Reaction Network, the set of events E is the set of reactions $\{R_1, \ldots, R_M\}$.

Synchronised simulation

An LHA $\mathcal{A}$ is synchronised with the simulations of a GSPN model $\mathcal{M}$. Each path is infinitely simulated when either an accepting state of $\mathcal{A}$ is reached, or the synchronisation can no longer proceeds (hence the path is rejected). The transitions of $\mathcal{A}$ synchronises with the transitions of the trajectory σ being sampled. An LHA admits two kinds of transitions.

- *Synchronising* transitions (associated with a subset of E, with $ALL = E$), which may be traversed when an event (in E) is observed.
- *Asynchronous* transitions (denoted by $\sharp$), which are traversed autonomously. Asynchronous transitions have priority over synchronised ones.

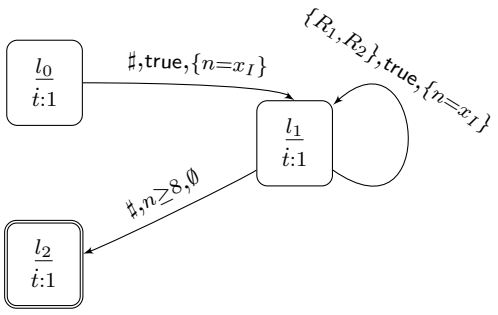


FIGURE 4.2: An example of LHA $\mathcal{A}_{count}$ for SIR model. It accepts a trajectory if the number of infected people reaches eight.

Let us illustrate a synchronised simulation with a simple example. We consider the SIR model $\mathcal{M}$ in Figure 4.1, and the LHA $\mathcal{A}_{count}$ depicted in Figure 4.2 (l_0 is the initial state, l_2 the final state). The LHA accepts the trajectory if the population reaches eight infected people.

The associated synchronised model is a couple $(\mathcal{M}, \mathcal{A})$. A state of the synchronised model is $(m, l, X, time)$ where m is a marking of $\mathcal{M}$, $l \in L$ a location, $X = (n, t)$ are the automaton values, and $time$ is the time.

Let us describe the synchronised simulation of the path $\sigma = (95, 5, 0) \xrightarrow{0.2} (94, 6, 0) \xrightarrow{0.1} (93, 7, 0) \xrightarrow{0.08} (93, 6, 1) \xrightarrow{0.02} (92, 7, 1) \xrightarrow{0.12} (91, 8, 1)$.

- First, the synchronised state is initialised: $(m_0, l_0, (0.0, 0.0), 0.0)$.
- From l_0, there exists only one edge $l_0 \to l_1$. This edge is asynchronous, so it does not need a reaction occurrence to be fired. The transition predicate is true, so $l_0 \to l_1$ is fired, and the variable n is updated. The state becomes $(m_0, l_1, (5.0, 0.0), 0.0)$.
- One asynchronous edge arises from l_1, but the transition predicate $\gamma = n \geq 8$ is not satisfied, the edge is not fired. The other edge is synchronous, so we have to wait that a reaction occurs.
- Time goes on until a duration of 0.2. In the location l_1, $\dot{t} = 1$ gives that t has a flow of 1 in l_1. More precisely, $Flow(l_1) = (0.0, 1.0)$ which means the derivative of t with respect to *time* is 1, whereas the derivative of n is zero. Thus $t = 0.2$.
- Then, R_1 occurs. The marking evolves to $m_1 = (94, 6, 0)$. One edge $l_1 \xrightarrow{\{R_1,R_2\},\mathsf{true},\{n=x_I\}} l_1$ is synchronous. As $R_1 \in \{R_1, R_2\}$ and $\gamma = \mathsf{true}$ is verified, this edge is fired. The synchronised state becomes $((94, 6, 0), l_1, (6.0, 0.2), 0.2)$.
- The transition predicate $\gamma = n \geq 8$ is still not verified, so $l_1 \to l_2$ is still not fired.
- The last two steps are repeated until we arrive at the time $t = 0.2 + 0.1 + 0.08 + 0.02 + 0.12 = 0.52$. Then R_1 occurs, and the synchronised state is updated: $((91, 8, 1), l_1, (8.0, 0.52), 0.52)$. This time, the transition predicate $\gamma = n \geq 8$ is fulfiled, so $l_1 \to l_2$ is fired. As l_2 is a final state, the simulation ends with state $((91, 8, 1), l_2, (8.0, 0.52), 0.52)$. The path is accepted by the LHA.

Note that with the path $\sigma = (95, 5, 0) \xrightarrow{0.12} (95, 4, 1) \xrightarrow{0.13} (95, 3, 2) \xrightarrow{0.11} (95, 2, 3) \xrightarrow{0.021} (95, 1, 4) \xrightarrow{0.38} (95, 0, 5)$, the LHA would have rejected σ.

4.4.3 HASL Expressions

An HASL formula is described by a couple $(Z, \mathcal{A})$, where $\mathcal{A}$ is an LHA and Z is an expression built on top of the grammar in Definition 4.4.3. It leads to the evaluation of an arithmetic expression based on synchronised simulations. Such an arithmetic expression is formally described by the HASL expression Z.

Definition 4.4.3 (HASL Expression)
An HASL expression Z is defined by the following grammar

$$\begin{aligned} Z &::= c \mid PROB \mid AVG(Y) \mid Z + Z \mid Z \cdot Z \\ Y &::= c \mid Y + Y \mid Y \cdot Y \mid Y/Y \mid \\ &\quad Last(y) \mid Min(y) \mid Max(y) \mid Int(y) \mid Mean(y) \\ y &::= c \mid x \mid y + y \mid y \cdot y \mid y/y \end{aligned}$$

In this definition, y is an arithmetic expression built on top of the automaton variables, computed at each step. Y is called an HASL trajectory expression. It represents a value computed at the end of an accepted trajectory. Z is an HASL expression, evaluated over a set of simulated trajectories. For Z expressions, $PROB$ is the probability that a path is accepted by the LHA, $AVG(Y)$ is the first moment (expectation) of the HASL trajectory expression Y, which allows the computation of any $k-th$ order moments.

For an expression Y, several quantities are available:

- $Last(y)$ is the value of y in the last state
- $Min(y), Max(y), Mean(y)$ are the minimum, maximum and average value of y along the trajectory
- $Int(y)$ is the integral over time of y along the trajectory

Definition 4.4.4 (Notation for synchronised simulations)
Let Y an HASL trajectory expression, $\mathcal{A}$ an LHA and $\mathcal{M}$ a CTMC. The notation

$$x \sim (Y, \mathcal{A}) \times \mathcal{M}$$

describes the evaluation of the trajectory expression Y after a simulation of $\mathcal{M}$ synchronised with $\mathcal{A}$.

Considering the synchronised simulation example of Section 4.4.2, $x \sim (Last(n), \mathcal{A}_{count}) \times \mathcal{M}$ means that x stores the last value of n, which is 8.

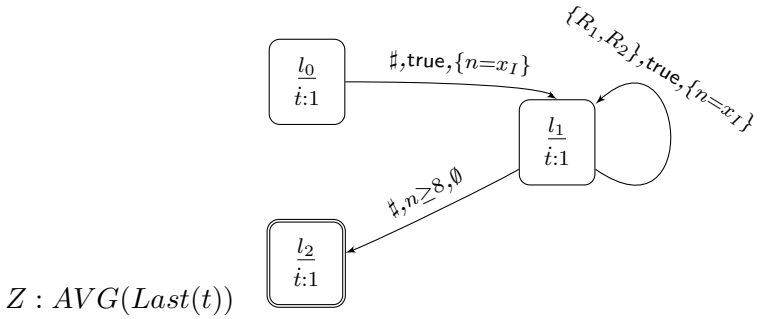
Remark 4.4.3
The notation of Definition 4.4.4 is mainly used in the automaton-ABC procedure detailed in the next chapter.

4.4.4 Cosmos Statistical Model Checker

Cosmos (Ballarini et al., 2015) is a statistical model checker for HASL specifications developed in C++. It takes as inputs a Generalised Stochastic Petri Net (GSPN) and an HASL formula (i.e. an LHA and an HASL expression).

This tool implements the simulation of any Discrete-Event Stochastic Processes described by a GSPN. This formalism allows efficient path simulation of the model. Each simulation is synchronised with the LHA, and evaluation of the HASL expression is done during the simulation. For more details, we refer the reader to (Djafri, 2012, Chapter 6) and (Barbot, 2014, Chapter 6).

Script 3 (An example of a Cosmos run)
An example of a Cosmos script is available in `code/chap3/example_cosmos_sir.sh` *with the following HASL formula that computes the average time the SIR model reaches eight infected people.*



4.5 Summary

In this chapter, we described:

- Temporal logics to formally describe specifications over a CTMC.
- Model checking of CTMCs with a particular focus on two problems (estimation and threshold problem) and the related statistical methods.
- Model checking on collections of CTMCs called *model checking of parametric Continuous-Time Markov Chain* and the related statistical methods.
- The HASL formalism designed for discrete event stochastic processes such as CTMCs.

At this point of the thesis, we have defined the main concepts to present our contributions in statistical inference and verification of Continuous-Time Markov Chains, based on ABC methods with Linear Hybrid Automata.

Chapter 5

Automaton-ABC for the statistical inference of CTMCs

The difficulty of statistical inference for complex models relies on the lack of a closed form of the likelihood. Continuous-Time Markov Chains form a class of stochastic models that suffers from this difficulty: in most cases, the likelihood of observations is intractable.

Thus, ABC likelihood-free methods (Section 3.3) seem adequate for the Bayesian estimation of such models. In this chapter, our goal is to define the statistical framework of CTMCs and apply ABC methods using the synchronised simulation included in the HASL formalism (Section 4.4). We address several tasks:

- In Section 5.1, we describe the observation models of CTMCs. We consider two types of observations: when trajectories are observed continuously, and when trajectories are observed discretely in time.
- In Section 5.2, we detail a new algorithm called automaton-ABC, which combines classical ABC methods with synchronised simulations (defined in Section 4.4.2).
- Section 5.3 and Section 5.4 are applications of the automaton-ABC algorithm for two different tasks: take into account oscillatory trends of models and Bayesian inference.

5.1 Observation model and likelihood

Let us consider observations y_{exp}. A statistical model induced by a parametric CTMC (Definition 4.3.1) is a dominated model (Billingsley, 1961) (Proposition 5.1.1).

Proposition 5.1.1 (Statistical model of a parametric CTMC)
Let $(S^\theta)_{\theta\in\Theta}$ a parametric CTMC. Then $(\mathbb{P}_{S^\theta})_{\theta\in\Theta}$ is a statistical model dominated by a measure $\overline{\mu}$.

Corollary 5.1.1 (Density of a CTMC)
Let S a CTMC. $\mathbb{P}_S$ admits a density p_S:

$$p_S : Path(\mathcal{M}) \to \mathbb{R}_{\geq 0}$$
$$\sigma = \left(\mathbf{s}_0 \xrightarrow{t_0} \dots \xrightarrow{t_{k-1}} \mathbf{s}_k\right) \to P(\mathbf{s}_0)\prod_{i=1}^{k} Q(\mathbf{s}_{i-1},\mathbf{s}_i)e^{-E(\mathbf{s}_{i-1})t_{i-1}}$$

Such properties are also helpful in the Bayesian framework (Proposition 5.1.2).

Proposition 5.1.2 (Conditional density of a parametric CTMC)
Let $(S^\theta)_{\theta\in\Theta}$ be a parametric CTMC, $\mathbb{P}_\vartheta$ a prior distribution.

- *$(\theta, A) \to \mathbb{P}_{S^\theta}(A)$ is a transition kernel.*
- *$\mathbb{P}_{S|\vartheta=\theta}$ admits a continuous density $\sigma \to p(\sigma|\theta) = p_{S^\theta}(\sigma)$.*

Before any mention of learning algorithm from data, we need to formulate the nature of observations y_{exp} mathematically in the case of Continuous-Time Markov Chains. We consider two types of observations.

- *Event-discrete observations.* We observe a path $\sigma \in Path(\mathcal{M})$ by observing the transitions of the CTMC.
- *Time-discrete observations.* At fixed times $t_1, \dots, t_N$, we observe the state of the CTMC. It corresponds to the classical case of observation of a state-space dynamical model.

5.1.1 Event-discrete observations

Let $S^\theta = (S_t{}^\theta)_{t\in\mathbb{R}_{\geq 0}}$ be a CTMC defined by a transition rate matrix Q_θ, transition probability matrix P_θ, and exit rate E_θ. We consider the paths $\sigma^{(1)}, \dots, \sigma^{(N)}$ as observations.

Definition 5.1.1 (Likelihood function of a parametric CTMC)
Let $\sigma^{(1)}, \ldots, \sigma^{(N)}$ observation paths. The likelihood function is defined as:

$$\begin{aligned} l(\theta; \sigma^{(1)}, \ldots, \sigma^{(N)}) &= \prod_{k=1}^{N} p(\sigma^{(i)}|\theta) \\ &= \prod_{i=1}^{N} P(\sigma^{(i)}{}_{[0]}) \prod_{j=1}^{k^{(i)}} Q_\theta(\sigma^{(i)}{}_{[j-1]}, \sigma^{(i)}{}_{[j]}) e^{-E_\theta(\sigma^{(i)}[j-1]\delta(\sigma^{(i)}, j-1))} \end{aligned}$$

where $k^{(i)}$ is the number of states of $\sigma^{(i)}$.

These observations correspond to the case where we fully observe the transitions of the CTMC model. This case seems unrealistic for most biological phenomena, but it is more reasonable for queueing network models or fault tolerance models because computers can log any occurred event. An analytical form of the Maximum Likelihood Estimator (Definition 3.1.3) can be derived (Bladt and Nielsen, 2017, Chapter 12).

Sometimes, we can only observe a subset of the state space. Thus, we introduce an observation function γ defined over the set of trajectories by:

$$\begin{aligned} \gamma : Path(\mathcal{M}) &\to \mathcal{E} \\ \sigma \to \gamma(\sigma) = g(\mathbf{s}_0) &\xrightarrow{t_0} \ldots \xrightarrow{t_k} g(\mathbf{s}_k) \end{aligned}$$

where g is the projection of an element $\mathbf{s} \in \mathbf{S}$ over the set of observed states. Thus, $\gamma(\sigma)$ is a trajectory with a reduced state space.

A frequent case is when only a few variables of the system are observed. In the Chemical Reaction Network framework, it means only a few species are observed. In this case, we define $I_g = \{i_1, \ldots, i_{d_{obs}}\} \subseteq \{1, \ldots, d\}$ as the set of observed species and the observation function g is:

$$\begin{aligned} g : \mathbf{S} &\to \mathbb{N}^{d_{obs}} \\ \mathbf{s} &\to (\mathbf{s}[i_1], \ldots, \mathbf{s}[i_{d_{obs}}]) \end{aligned}$$

For example, with the SIR model 2.15, we usually only observe the number of infected people, so the observation function over the state space g is defined as

$$\begin{aligned} g : \quad & \mathbb{N}^3 \to \mathbb{N} \\ (X_S, X_I, \quad & X_R) \to X_I. \end{aligned}$$

5.1.2 Time-discrete observations: state-space model

In most applications of continuous phenomena, observations are considered discrete in time. In this case, the standard framework used for the observation model is the state-space model (Cappé, Moulines, and Rydén, 2009).

Let $S = (S_t)_{t\in\mathbb{R}_{\geq 0}}$ a CTMC. Let $t_1, \dots, t_K$ a collection of observation times. We suppose $Z = (S_k)_{1\leq k\leq K}$ is unobservable, and we consider $Y = (Y_k)_{1\leq k\leq K}$ the corresponding observable states. The state-space model is the Hidden Markov Model (Y, Z) defined by the scheme, for $k \in \{1, \dots, K\}$:

$$\begin{aligned} Y_k &\sim g(.|S_{t_k}, \theta_y) \\ Z_k = S_{t_k} &\sim p(.|S_v, \theta) \text{ with } t_0 \leq v \leq t_k \end{aligned}$$

.

$(Y_k)_{k\in\ \mathbb{N}}$ are the observations of some hidden/latent continuous trajectory. g is the observation function, it models the link between the observed state and the latent state. In this case, the i.i.d random observations are named $(Y^{(1)}, \dots, Y^{(N)})$, which a realisation is $(y^{(1)}_{1:K}, \dots, y^{(N)}_{1:K})$.

Due to the Markov property of CTMCs, the discrete scheme is rewritten as:

$$\begin{aligned} Y_k &\sim g(.|S_{t_k}, \theta_y) \\ S_{t_k} &\sim p(.|S_{t_{k-1}}, \theta) \end{aligned}$$

.

Let us compute the likelihood for one observation $y_{1:K} \in \mathbf{S}^K$:

$$\begin{aligned} l(\theta; y_{1:K}) &= p_Y(y_{1:K}) \\ &= \int_{z_{0:K}\in\mathbf{S}^{K+1}} p_{Y,Z}(y_{1:K}) dz_{0:K} \\ &= \int_{z_{0:K}\in\mathbf{S}^{K+1}} p_{Y|Z}(y_{1:K}|\ z_{0:K}) p_Z(z_{0:K}) dz_{0:K} \\ &= \int_{z_{0:K}\in\mathbf{S}^{K+1}} p(z_0) \prod_{k=1}^{N} g(y_k|z_k) p(z_k|z_{k-1}) dz_{0:K} \end{aligned}$$

For i.i.d observations $y^{(1)}_{1:K}, \dots, y^{(N)}_{1:K}$, the likelihood is:

$$l(\theta;\ y^{(1)}_{1:K}, \dots, y^{(N)}_{1:K}) = \prod_{i=1}^{N} \int_{z_{0:K}\in\mathbf{S}^{K+1}} p(z_0) \prod_{k=1}^{N} g(y^{(i)}_k|z_k) p(z_k|z_{k-1}) dz_{0:K}$$

When the true state is observed at each step, $Y_k = S_{t_k}$ and the likelihood becomes

$$l(\theta;\ y_{1:K}^{(1)}, \dots, y_{1:K}^{(N)}) = \prod_{i=1}^{N} p(y_0) \prod_{k=1}^{N} p(y_k^{(i)} | y_{k-1}^{(i)}).$$

Even in this simple case, the likelihood is difficult to compute. Indeed, the transition function of a CTMC $p(y_k|y_{k-1})$, i.e. the probability of being in a state at time t_k given the state at time t_{k-1}, is challenging to compute. The observations times are not coincident with jumps/transitions of the system, so we only know

$$p(y_k|y_{k-1}) = \mathbb{P}(S_{t_k} = y_k | S_{t_{k-1}} = y_{k-1}) = \delta_{y_{k-1}}(y_k) + Q(y_{k-1}, y_k)(t_k - t_{k-1}) + o(t_k - t_{k-1}).$$

Note that given observations y_k, t_k and a CTMC, methods that reconstruct the most probable trajectory σ exist, such as Viterbi algorithm or more evolved methods (Perkins, 2017).

Statistical inference of Continuous-Time Markov Chains based on time-discrete observations has been much studied in the literature (Craciun et al., 2013; Schnoerr, Sanguinetti, and Grima, 2017; Loskot, Atitey, and Mihaylova, 2019). Consideration of time-discrete observations for CTMCs has been considered in (Bladt and Sørensen, 2005). They proved the existence and uniqueness of the maximum likelihood estimator under certain conditions, and the possibility of applying Expectation-Maximisation (EM) and MCMC procedures. More evolved EM algorithms suited to CTMCs exist. For example, stochastic approximation EM algorithm that involves an ABC procedure is detailed in (Picchini and Samson, 2018). Some of these statistical methods have been implemented in R (Pfeuffer, 2017).

Particularly, Bayesian inference for CTMCs has been considered (Gómez-Corral et al., 2015). (Warne, Baker, and Simpson, 2019) reviews the main Bayesian techniques for CTMCs with a particular focus on ABC related methods. They are illustrated by numerical examples.

In the thesis (Alharbi, 2018), Bayesian inference is considered on CTMCs, focusing on the two main likelihood-free methods of the Bayesian statistical literature: ABC and Particle Marginal Metropolis-Hasting algorithm (Andrieu, Doucet, and Holenstein, 2010). Note that to compare the likelihood-free posteriors, in (Alharbi, 2018, Chapter 6), they performed exact posterior distribution inference for the repressilator model (a model considered in the last two Sections of the Chapter) using the Sequential Importance Sampling algorithm. It lasted 7137 hours after a state-space reduction. The author had difficulties managing the run of the Particle Marginal

Metropolis-Hasting algorithm, because assessing a trustful convergence of a Markov Chain is a challenging task. Driving the convergence of ABC-SMC (Algorithm 14) was easier because the Sequential Monte Carlo scheme adapts the tolerance ϵ during the run, and the α adaptive tolerance parameter (Section 3.3.2) easily adapts the convergence speed.

More recently, ABC methods have been coupled with Multi-Level Monte Carlo for CTMC inference. The main idea of Multi-Level Monte Carlo is to estimate (the expectation of) a probability distribution with approximate samples. These samples have different levels of accuracy $l \in \{1, \dots, L\}$. The Multi-Level Monte Carlo estimator reduces the variance compared to the classical Monte Carlo estimator. In (Jasra et al., 2019), the approximation is related to an ABC tolerance $\epsilon_l, l \in \{1, \dots, L\}$, i.e. the approximate samples are drawn from $\pi_{ABC}^{\epsilon_l}, \{1, \dots, L\}$. In (Lester, 2018), the approximation is related to the accuracy of the CTMC simulation. Instead of exact CTMC simulation (Stochastic Simulation Algorithm, see Section 2.3.4), simulation is performed by the Tauleap Algorithm 3 with a time step $\tau_l, l \in \{1, \dots, L\}$.

ABC methods have proven efficiency in the context of CTMCs. In this chapter, we will consider ABC procedures with a verification point of view for Bayesian inference.

5.1.3 Approximate Bayesian Computation for event-discrete observations

As we saw in Section 3.3 and above in this chapter, Approximate Bayesian Computation methods are suitable for Bayesian inference when the likelihood is complex. Hence, these methods are helpful for Continuous-Time Markov Chains. In this section, we detail several configurations of ABC methods in order to show how they are usable, and apply them to several models.

Distance over paths of CTMC

Here, we consider event-discrete observations (Section 5.1.1) $y_{exp} = (\upsilon^{(1)}, \dots, \upsilon^{(n)})$ where $\upsilon^{(i)} = g(\mathbf{s}_{0,(i)}) \xrightarrow{t_{0,(i)}} \dots \xrightarrow{t_{k^{(i)}-1,(i)}} g(\mathbf{s}_{k^{(i)},(i)}), I_g \subseteq \{1, \dots, d\}$ is the set of observed species, and $g \in \{0,1\}^d$ the observation function. By denoting $g(\mathbf{s}_{j,(i)}) = y_j^{(i)}$, one can represent the i-th observation with $(y_j^{(i)}, t_j^{(i)})_{j \in 0:k^{(i)}}$. The observation paths are defined over $[0, T_{end}]$, $T_{end} \in \mathbb{R}_{\geq 0}$, which means $\sum_j t_{j,(i)} = T_{end}$.

To execute the ABC algorithm, we have to define a distance function between observations and simulations. A classical choice is the Euclidean distance:

$$\begin{aligned} d_{||.||_2} : \mathbb{R}^k \times \mathbb{R}^k &\to \mathbb{R}_{\geq 0} \\ (y^{(1)}, y^{(2)}) &\to \sqrt{\sum_j (y_j^{(1)} - y_j^{(2)})^2} \end{aligned} \tag{5.1}$$

However, the dimension of our observations is not fixed. To use the Euclidean distance, we have to define a discrete timeline $t_1, \ldots, t_N$ over $[0, T_{end}]$ and collect the values of the paths $\gamma(\sigma^{(i)})$ over this timeline. This operation is equivalent to consider the scheme of time-discrete observations (Section 5.1.2), which lose information.

As the observations are step functions on a bounded interval $[0, T_{end}]$, one can use L^1 and L^2 distances for 1-dimensional trajectories:

$$d_{L^p}(\sigma_1, \sigma_2) = \left(\int_{[0,T_{end}]} |\sigma^1@t - \sigma^2@t|^p dt\right)^{\frac{1}{p}}, p \in \{1, 2\} \tag{5.2}$$

which guarantees the full use of the information contained in the observations.

As our observations are step functions, the integral is easy to compute. Figure 5.1 illustrates the computation for two one-dimensional paths σ_1 and σ_2. The distance between the paths is the sum of the green areas. Hence, the computational cost of the distance is proportional to the number of events for both trajectories.

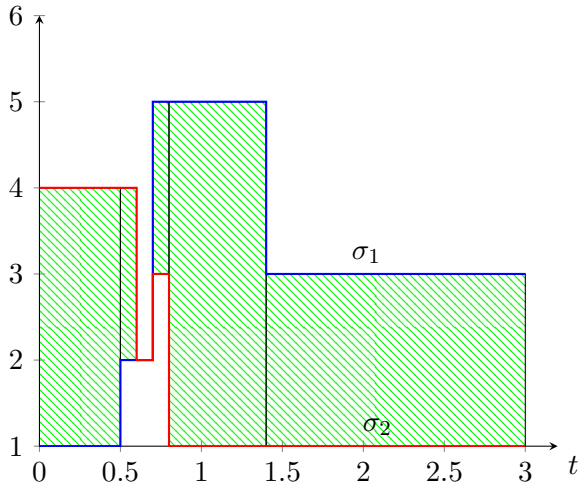


FIGURE 5.1: Plot of two 1-dimensional path. The L^1 distance of these paths is the sum of the areas hatched in green.

Examples of ABC-SMC inference on parametric CTMC with different observation schemes and distances

In this section, we present a few examples of Bayesian inference with ABC-SMC (Algorithm 14). We compare the effect of three observation configurations based on the same simulated dataset. We see the evolution of the resulting ABC posteriors around the true parameter estimated with a fixed number of simulations.

We consider a simulated dataset of five observations $y_{exp} = (v^{(1)}, \dots, v^{(5)})$ and the distance:

$$\begin{aligned} \rho : \mathcal{E}^5 \times \mathcal{E}^5 &\to \mathbb{R}_{\geq 0} \\ (y, y_{exp}) &\to \min_{1 \leq i,j \leq 5} d(y_{[i]}, \underbrace{y_{exp[j]}}_{v^{(j)}}) \end{aligned} \tag{5.3}$$

where d can either be the integral L^2 distance (5.2) or the Euclidean distance (5.1). For each model, two sets of observation times are used in the case of Euclidean distance. Observations are collected according to these observation times.

Resulting posterior distributions should be comparable, so we fix a maximal number of simulations $N_{end} \in \{10^6, 10^7\}$. N_{end} is the simulation budget. When the number of simulations reaches N_{end}, we let the current step of the ABC-SMC algorithm finish and end the algorithm's run. In practice, it results in a similar number of simulations. In this application, the adaptive tolerance schedule is set with $\alpha = 0.75$ (Section 3.3.2).

Enzymatic reaction model We consider the enzymatic reaction (ER) system. A *substrate* species S is converted into a *product* P through the mediation of an *enzyme* E. The processes are modelled by the CRN (5.4), parametrised by the parameter vector $\theta = (k_1, k_2, k_3)$. k_1, k_2, k_3 are the kinetic rate constants of the reactions R_1, R_2, R_3. The initial state is $(E_0, S_0, ES_0, P_0) = (100, 100, 0, 0)$.

$$\begin{aligned} R_1 &: E + S \xrightarrow{k_1} ES \\ R_2 &: ES \xrightarrow{k_2} E + S \\ R_3 &: ES \xrightarrow{k_3} E + P \end{aligned} \tag{5.4}$$

We simulate the dataset (five observations) with $\theta_{true} = (1, 1, 5)$ and $T_{end} = 0.075$. Parameter inference focuses on the parameter k_3 (the two others are supposed known) with the prior distribution $k_3 \sim \mathcal{U}(0, 100)$.

Three observation settings are considered with an end time of $T_{end} = 0.075$:

- 5 event-discrete observations $y_{exp} = (v^{(1)}, \dots, v^{(5)})$, with L^2 distance.
- 5 time-discrete observations with 10 points $y^{(1)}, \dots, y^{(5)}$ with $y^{(i)} = (y_1^{(i)}, \dots, y_{10}^{(i)})$ collected from $(v^{(1)}, \dots, v^{(5)})$, with Euclidean distance.
- 5 time-discrete observations with 75 points $y^{(1)}, \dots, y^{(5)}$ with $y^{(i)} = (y_1^{(i)}, \dots, y_{75}^{(i)})$ collected from $(v^{(1)}, \dots, v^{(5)})$, with Euclidean distance.

One of the trajectories from the simulated dataset is shown in Figure 5.2 with the two time-discrete observations settings.

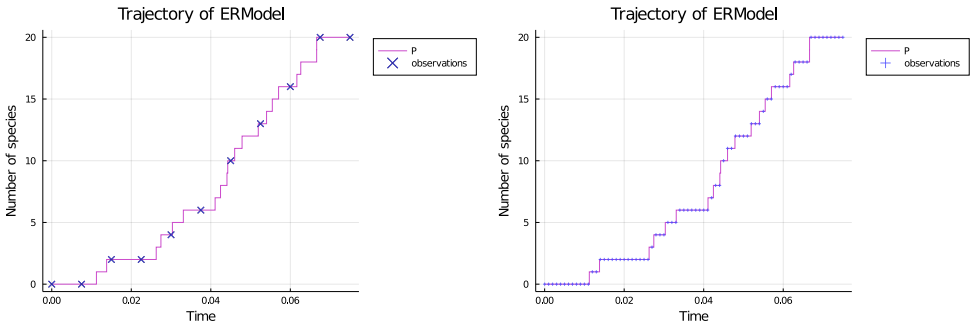


FIGURE 5.2: A trajectory from the simulated dataset. Left picture: 10-points observation. Right picture: 75-points observation.

Figure 5.3 shows the resulting ABC-SMC posteriors. Table 5.1 reports statistics about the posteriors. These results indicate that with the same simulation budget, the resulting ABC posteriors are pretty similar in estimation accuracy. The estimated mean (Table 5.1, first row) with the 10-points observations is slightly better, even if it contains less information than the two other settings.

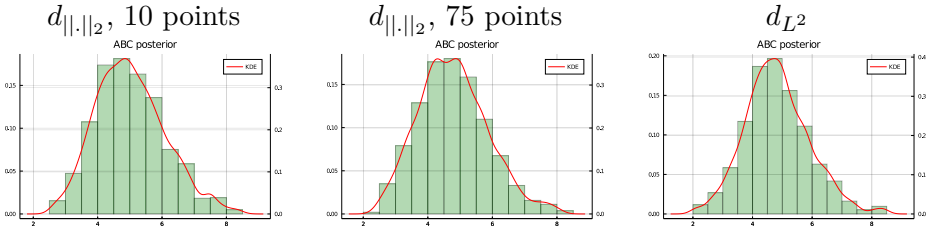


FIGURE 5.3: ABC-SMC posteriors with the ER model. $N_{end} = 10^6$. Left: time-discrete observations with 10 points, center: time-discrete observations with 75 points, right: the 5 trajectories.

	$d_{\|\|.\|\|_2}$, 10 points	$d_{\|\|.\|\|_2}$, 75 points	d_{L^2}
Mean	5.02	4.78	4.82
Std	1.08	1.07	1.08

TABLE 5.1: Statistics of the ABC posterior for the three observations settings for the ER model with $N = 10^6$ particles. True value is $k_3 = 5.0$.

SIR model We consider the SIR model already described in 2.15 with initial state $\mathbf{s}_0 = (95, 5, 0)$. We simulate the dataset with $\theta_{true} = (1.2E{-}3, 5.0E{-}2)$ and $T_{end} = 150.0$. Both parameters are infered with prior distributions $k_i \sim \mathcal{U}(5E{-}5, 3E{-}3)$ and $k_r \sim \mathcal{U}(5E{-}3, 0.2)$.

The three observations settings are:

- 5 event-discrete observations $y_{exp} = (v^{(1)}, \dots, v^{(5)})$, with L^2 distance.
- 5 time-discrete observations with 10 points $y^{(1)}, \dots, y^{(5)}$ with $y^{(i)} = (y_1^{(i)}, \dots, y_{10}^{(i)})$ collected from $(v^{(1)}, \dots, v^{(5)})$, with Euclidean distance.
- 5 time-discrete observations with 150 points $y^{(1)}, \dots, y^{(5)}$ with $y^{(i)} = (y_1^{(i)}, \dots, y_{150}^{(i)})$ collected from $(v^{(1)}, \dots, v^{(5)})$, with Euclidean distance.

One of the trajectories from the simulated dataset is shown in Figure 5.4 with the two time-discrete observations settings.

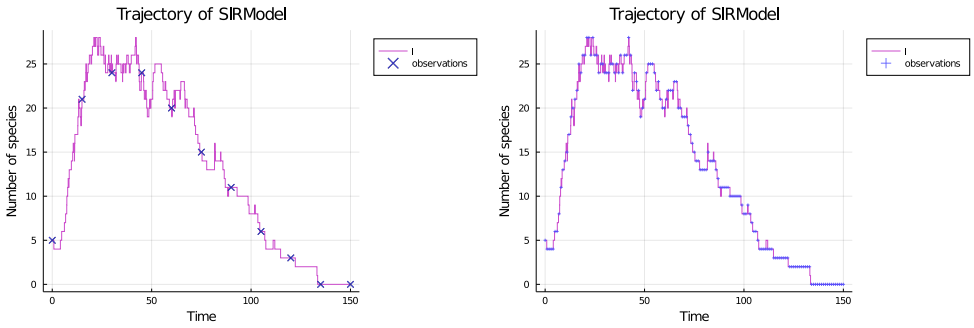


FIGURE 5.4: A trajectory from the simulated dataset. Left picture: 10-points observation. Right picture: 150-points observation.

Figure 5.5 shows the resulting ABC-SMC posteriors. Tables 5.2 and 5.3 report statistics about the posteriors. For the two simulation budgets $N_{end} = 10^6$ and $N_{end} = 10^7$, no significant improvement in the parameter inference has been noticed for the 10-points observations than the two more informative configurations.

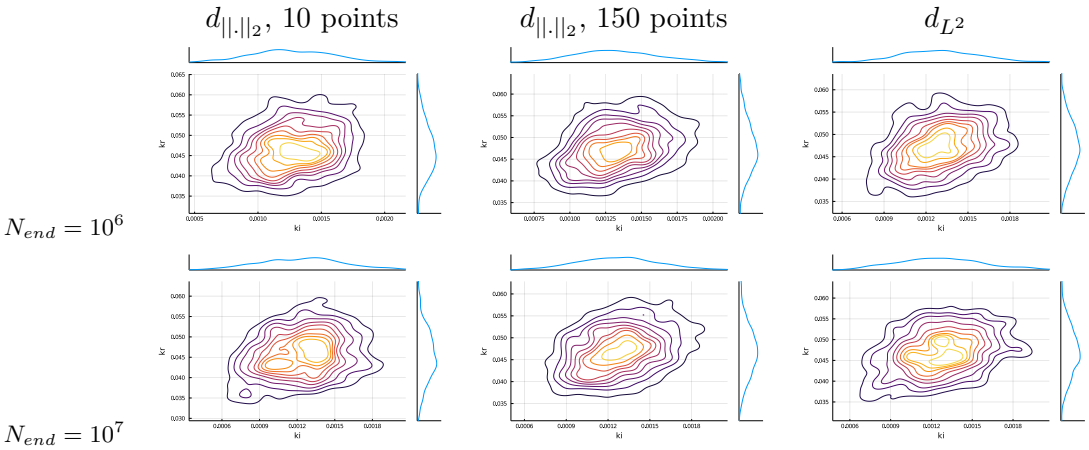


FIGURE 5.5: ABC-SMC posterios with the SIR model. First row: $N_{end} = 10^6$, second row: $N_{end} = 10^7$. Left: time-discrete observations with 10 points, center: time-discrete observations with 75 points, right: the 5 trajectories.

	$d_{\|\|.\|\|_2}$, 10 points	$d_{\|\|.\|\|_2}$, 150 points	d_{L^2}
Mean k_i	$1.27E{-}3$	$1.32E{-}3$	$1.31E{-}3$
Std k_i	$3.31E{-}4$	$3.14E{-}4$	$3.14E{-}4$
Mean k_r	$4.70E{-}2$	$4.75E{-}2$	$4.72E{-}2$
Std k_r	$7.54E{-}3$	$6.35E{-}3$	$6.35E{-}3$

TABLE 5.2: Statistics of the ABC posterior for the three observations configurations for the SIR model with $N = 10^6$ particles. The true parameter is $(k_i, k_r) = (1.2E{-}3, 5E{-}2)$.

	$d_{\|\|.\|\|_2}$, 10 points	$d_{\|\|.\|\|_2}$, 150 points	d_{L^2}
Mean k_i	$1.25E{-}3$	$1.27E{-}3$	$1.27E{-}3$
Std k_i	$3.4E{-}4$	$2.90E{-}4$	$3.25E{-}4$
Mean k_r	$4.57E{-}2$	$4.75E{-}2$	$4.73E{-}2$
Std k_r	$6.96E{-}3$	$6.54E{-}3$	$6.6E{-}3$

TABLE 5.3: Statistics of the ABC posterior for the three observations configurations for the SIR model with $N = 10^7$ particles. The true parameter is $(k_i, k_r) = (1.2E{-}3, 5E{-}2)$.

Discussion The most time-consuming experiment (SIR model with $N_{end} = 10^7$, 150-points observations) has not exceeded 35 minutes with eight jobs. These two experiments showed that adding information in the observations does not constantly improve the inference in likelihood-free ABC methods. Indeed, the little information contained in the addition of more observations is counterbalanced by the *curse of dimensionality*. Adding more points leads to the increase of the observation space

dimension. Thus, an important task is to compute informative statistics (Sisson, Fan, and Beaumont, 2019, Chapter 5) over the observations. This task can be considered more important than collecting more observations in some cases. An original approach with parametric CTMC oscillators is presented in Section 5.3.

Script 4 (Examples of ABC inference on parametric CTMC)
The scripts of these ABC inference examples can be found in the git repository of the thesis at `code/chap4/inference`.

5.2 Automaton-ABC: ABC procedures with synchronised simulation

In Section 4.4.2 of Chapter 4, we have defined synchronised simulations of CTMCs with a Linear Hybrid Automaton.

Evaluating an HASL trajectory expression Y with the synchronised simulation of a CTMC can be seen as an efficient method to compute complex summary statistics based on properties. Indeed, the automaton variables are updated within the simulation, which can be seen as statistics of the simulated trajectory. In this context, HASL formalism offers two main advantages:

- It offers a well-suited and visual tool for the definition of properties over CTMC trajectories.
- A synchronised simulation does not need the storage of trajectory values which saves memory allocation time.

In the following, we formulate an adaptation of ABC procedures (Section 3.3) with HASL formalism called *automaton-ABC*. As classical ABC for a parametric CTMC $(\mathcal{M}_\theta)_{\theta\in\Theta}$, automaton-ABC requires a prior π and a tolerance ϵ. In this context, we do not consider observations y_{exp}. Instead, we consider an HASL trajectory expression Y and an LHA $\mathcal{A}$. The main difference with classical ABC is that the simulation of the CTMC is replaced by the synchronised simulation $(Y, \mathcal{A}) \times \mathcal{M}$ (Definition 4.4.4). Algorithm 15 details the automaton-ABC analogous to the simple ABC Algorithm 12.

Algorithm 15 General automaton-ABC Algorithm

Require: $(\mathcal{M}_\theta)_{\theta\in\Theta}$ parametric CTMC, π prior,
N: number of particles, ϵ: tolerance level,
Y : HASL trajectory expression, $\mathcal{A}$: LHA
Ensure: $(\theta^{(i)})_{1\leq i\leq N}$ drawn from π^{ϵ}_{ABC}
for $i = 1 : N$ **do**
repeat
$\theta' \sim \pi$
$d' \sim (Y, \mathcal{A}) \times \mathcal{M}_{\theta'}$
until $d' \leq \epsilon$
$\theta^{(i)} \leftarrow \theta'$
end for

Algorithm 16 is the Sequential Monte Carlo version of Algorithm 15, with the adaptive tolerance schedule of Section 3.3.2.

Algorithm 16 General automaton-ABC Sequential Monte Carlo Algorithm

Require: $(\mathcal{M}_\theta)_{\theta\in\Theta}$ parametric CTMC, π prior,
N: number of particles, ϵ: tolerance level, α: quantile level,
Y HASL trajectory expression, $\mathcal{A}$ LHA, K: kernel proposal density
Ensure: $(\omega^{(i)}, \theta^{(i)})_{1\leq i\leq N}$ weighted samples drawn from π^{ϵ}_{ABC}
$\theta_0^{(i)} \sim \pi, i \in 1, \ldots, N$
$d_i \sim (Y, \mathcal{A}) \times \mathcal{M}_{\theta_0^{(i)}}, 0 \in 1, \ldots, N$
$\epsilon_{current} \leftarrow quantile(\alpha, (d_i)_{1\leq i\leq N})$
$(\omega_0^{(i)})_{1\leq i\leq N} \leftarrow \frac{1}{N}$
$m \leftarrow 1$
while $\epsilon_{current} > \epsilon$ **do**
for $i = 1 : N$ **do**
repeat
Take θ' from $(\theta_{m-1}^{(j)})_{1\leq j\leq N}$ with probabilities $(\omega_{m-1}^{(j)})_{1\leq j\leq N}$
$\theta_m^{(i)} \sim K(.|\theta')$
$d_i \sim (Y, \mathcal{A}) \times \mathcal{M}_{\theta_m^{(i)}}$
until $d_i \leq \epsilon_{current}$
$\omega_m^{(i)} \leftarrow \frac{\pi\left(\theta_m^{(i)}\right)}{\sum_{i'=1}^{N} \omega_{m-1}^{(i')} K(\theta_m^{(i)}|\theta_{m-1}^{(i')})}$
end for
Normalise $(\omega_m^{(i)})_{1\leq i\leq N}$
$\epsilon_{current} \leftarrow quantile(\alpha, (d_i)_{1\leq i\leq N})$
$m \leftarrow m + 1$
end while
return $(\omega_m^{(i)}, \theta_m^{(i)})_{1\leq i\leq N}$

The fact that y_{exp} is not mentioned does not mean that these procedures cannot be related to statistical inference. Indeed, the constants of the automaton $\mathcal{A}$ may

contain the observations, which will be the case in Section 5.4. In the rest of the chapter, as well as the next chapter, automaton-ABC procedures are used to tackle several problems of statistical inference and statistical verification.

5.3 Oscillatory trends of genetic networks

In computational biology, quantitative analysis of the biological system's dynamics is challenging even for simple models (Baldan et al., 2010).

This section focuses on a specific behaviour of dynamical systems: oscillatory trends of gene regulatory networks. Genetics oscillators have grown much interest in the domain of synthetic biology over the past two decades (Purcell et al., 2010; O'Brien, Van Itallie, and Bennett, 2012; Li and Yang, 2018). They allow modelling many rhythmed physiological phenomena such as circadian clock (Leloup and Goldbeter, 2003), heartbeat or cell cycle.

Characterisation of oscillations in model checking CTMC has been considered in the literature (Ballarini, Mardare, and Mura, 2009) (Spieler, 2014, Chapter 7). In (Ballarini and Duflot, 2015), they use the expressiveness of HASL for the characterisation of noisy oscillations. In this work, we define a Linear Hybrid Automaton $\mathcal{A}_{per}$ based on (Ballarini and Duflot, 2015) that measures a CTMC trajectory time period and computes a distance to oscillations. We explore the parameter space of parametric CTMCs that show oscillatory behaviours with an indicated time period, based on the LHA $\mathcal{A}_{per}$. The main idea is to use this automaton that measures the mean time period of a trajectory within automaton-ABC (Section 5.2) to find the subset of the parameter space that produces trajectories with an indicated time period.

5.3.1 Period automaton $\mathcal{A}_{per}$

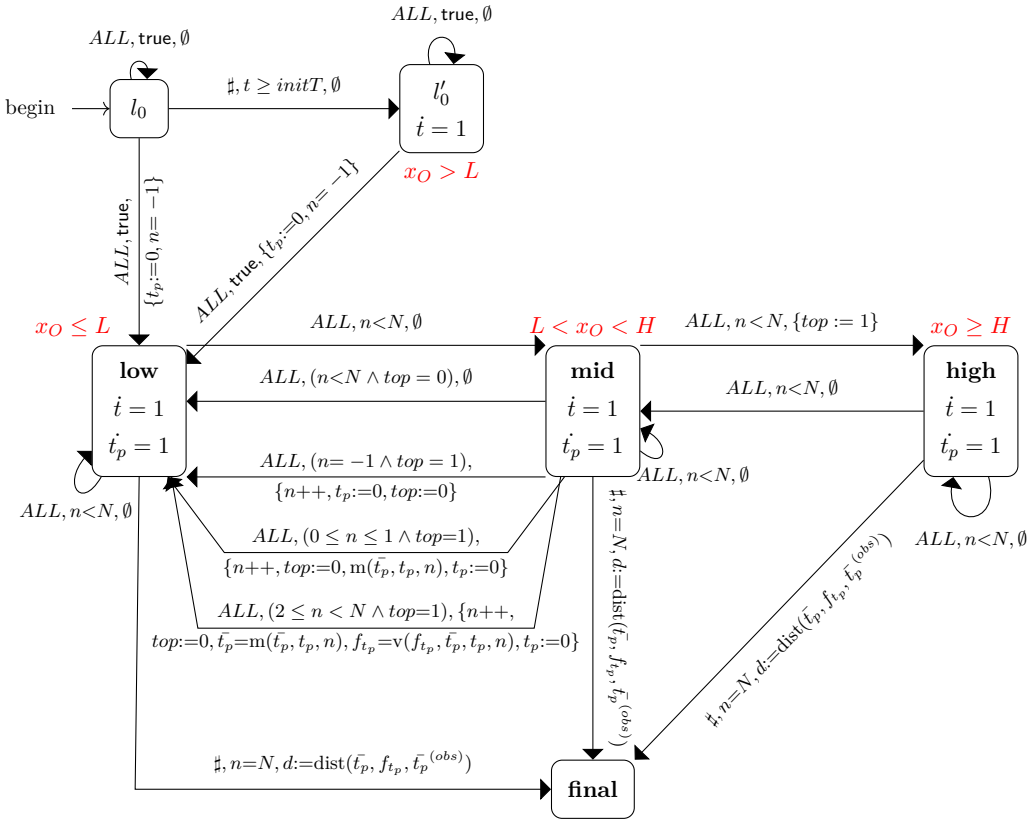


FIGURE 5.6: Period automaton $\mathcal{A}per$.

We consider the LHA $\mathcal{A}_{per}$ in Figure 5.6 to measure oscillatory trends of trajectories. This automaton is designed to compute two measures for the trajectory of a species O: the period duration mean $\bar{t_p}$ and the period variance f_{t_p}.

It is mainly formed of three locations: low, mid and high, plus other locations related to the initialisation and the end of the simulation. These three locations outline three levels of the species population O. Being in the location low means that the species population is below a user-defined level L, i.e. $x_O \leq L$, whereas entering mid (resp. high) means $L < x_O < H$ (resp. $x_O \geq H$). L and H are hyperparameters of the automaton and have to be defined during the creation of the automaton. Moreover:

- n is the number of detected periods.
- *top* is a boolean that indicates if the automaton has entered high location.
- $\bar{t_p}$ is the period mean, and f_{t_p} is the period variance.

- t_p is the current computed time period. It equals zero when $\bar{tp}$ and f_{t_p} are updated after the detection of a period, i.e. when $mid \to low$ occurs with $top \Leftrightarrow \mathsf{true}$.
- L, H and N_{per} are hyperparameters of the automaton. L, H defines the partition of the population species. N_{per} is the number of periods to be detected. After the detection of N_{per} periods, the automaton ends.
- d is the distance from a period mean reference called $\bar{t_p}^{(obs)}$.

A period duration is computed each time a trajectory contains a sub-trajectory that corresponds to a $\mathcal{A}per$ trace of the form $low \cdot (low|mid)^* \cdot high \cdot (high|mid)^* \cdot mid \cdot low$:

- We start to count a period when the automaton enters low.
- The trajectory oscillates between low and mid.
- The automaton enters high at least once.
- The period duration finishes when $mid \to low$ is observed with $top \Leftrightarrow \mathsf{true}$.

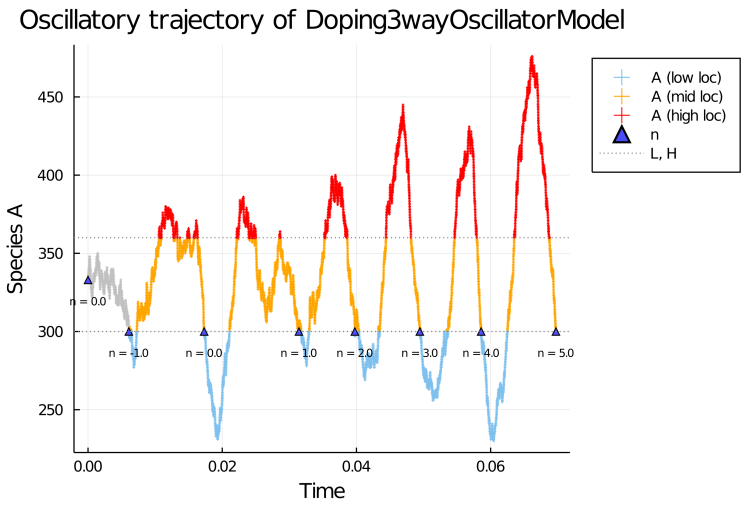


FIGURE 5.7: Example of a oscillatory trajectory simulation synchronised with $\mathcal{A}per$.

Figure 5.7 illustrates the behaviour of $\mathcal{A}per$ for a trajectory simulated by the doping 3-way oscillator (see section below) with $L = 300, H = 360, N_{per} = 5$. Let us describe this trajectory step by step:

- The grey part corresponds to the initial locations l_0 and l'_0. The automaton is in one of these locations until it reaches low, i.e. $x_O \leq L$.

- When the automaton reaches *low*, $n = -1$. The automaton waits to detect a period before computing the measures.
- The trajectory continues until a period is detected, n is updated to 0.
- The trajectory continues until another period is detected. n is updated to one, and the period mean is updated to t_p.
- For the next period, the period mean and variance period are updated according to Equations 5.5.
- The trajectory continues until n reaches $N_{per} = 5$.

$$\begin{aligned} \mathrm{m}(\bar{t_p}, t_p, n) &= \frac{(n-1)\bar{t_p} + t_p}{n} \\ \mathrm{v}(f_{t_p}, \bar{t_p}, t_p, n) &= \frac{n-2}{n-1} \cdot f_{t_p} + \frac{(\bar{t_p} - t_p)^2}{n} \end{aligned} \tag{5.5}$$

At the end of the simulation (i.e N_{per} periods are detected), the variable d is updated. The automaton computes a distance from a duration period mean.

$$\mathrm{dist}(\bar{t_p}, f_{t_p}, \bar{t_p}^{(obs)}) = \min\left(\frac{\bar{t_p} - \bar{t_p}^{(obs)}}{\bar{t_p}^{(obs)}}, \frac{\sqrt{f_{t_p}}}{\bar{t_p}^{(obs)}}\right)$$

The variation coefficient helps discard trajectories with, for example, periods of duration $0.5\bar{t_p}^{(obs)}$ and $2\bar{t_p}^{(obs)}$ but a resulting period mean of $\bar{t_p}^{(obs)}$. Thus, if $d \leq 0.1$, the relative error from the period mean reference is less than 10%, as well as the variation coefficient.

5.3.2 Applications of the automaton-ABC algorithm with $\mathcal{A}per$

In the following, we use the automaton-ABC algorithm with the period automaton $\mathcal{A}per$.

Doping 3-way oscillator

Model. First, we consider a toy model for oscillatons called *doping 3-way oscillator*, from which a probabilistic model checking description has been detailed in (Ballarini, Mardare, and Mura, 2009). It is described by the following six reactions with mass-action law:

$$A + B \xrightarrow{r_A} 2B \qquad B + C \xrightarrow{r_B} 2C \qquad C + A \xrightarrow{r_C} 2A \tag{5.6}$$

$$D_A + C \xrightarrow{r_C} A \qquad D_B + A \xrightarrow{r_A} D_B + B \qquad D_C + B \xrightarrow{r_B} D_C + C \tag{5.7}$$

The number of species is constant in time. Equations 5.6 creates oscillations of species by creating positive loop feedback. If the first reaction occurs a few times, the number of species B increases, which makes the second reaction (that induces consumption of B) more probable, and so on. Equations 5.7, called doping reactions, permit sustained oscillations by avoiding that one species vanishes.

Results. We apply the automaton-ABC algorithm with the synchronised simulation $(Last(\bar{t_p}), \mathcal{A}per)$. We want to find the subset of parameters for which oscillations have a period duration mean of 0.02.

We made two experiments. The first is one-dimensional, the setting is:

- $N_{per} = 4$ periods, $L = 300$ and $H = 360$
- $\bar{t_p}^{(obs)} = 0.01$
- $r_A \sim \mathcal{U}(0, 10)$ and other parameters are fixed ($r_B = r_C = 1.0$).
- $\mathbf{s}_0 = (A_0, B_0, C_0, (D_A)_0, (D_B)_0, (D_C)_0) = (333, 333, 333, 10, 10, 10)$
- $N = 1000$ particles
- Tolerance of 20% ($\epsilon = 0.2$)

Figure 5.8 shows the resulting automaton-ABC posterior. The sequential run lasted 140 seconds and performed 19846 simulations. First, we can observe that the support is included in $[0.0, 4.0]$, which is shrinker than the support of the prior: we have reduced the parameter space to a subset where it is probable to obtain trajectories with a period mean of 0.01 (relatively to a tolerance of 20%). Second, the posterior has only one mode, which is quite intuitive. As we have fixed r_B and r_C, one could expect that the period duration mean is directly linked to the kinetics of reaction 1, which is only parametrised by r_A.

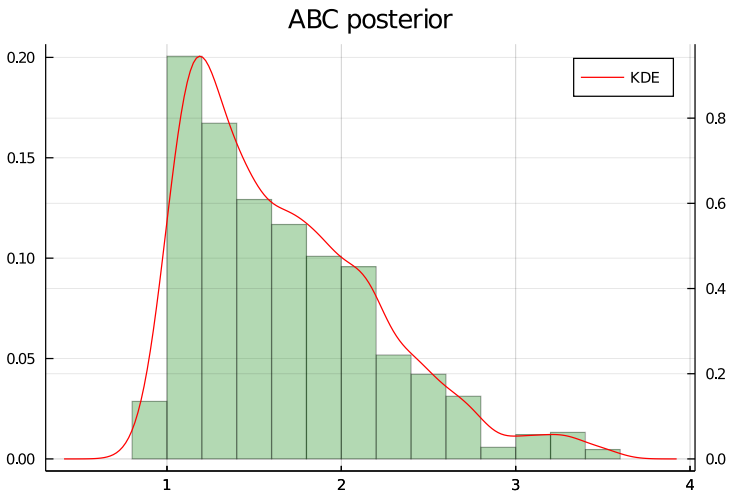


FIGURE 5.8: Automaton-ABC posterior of r_A with $\mathcal{A}per$ automaton for the 1D experiment of doping 3-way oscillator.

The second experiment considers variations over the whole parameter space. The setting is:

- $N_{per} = 4$ periods, $L = 300$ and $H = 360$
- $\bar{t_p}^{(obs)} = 0.01$
- $r_A, r_B, r_C \sim \mathcal{U}(0, 10)$
- $\mathbf{s}_0 = (A_0, B_0, C_0, (D_A)_0, (D_B)_0, (D_C)_0) = (333, 333, 333, 10, 10, 10)$
- $N = 1000$ particles
- Tolerance of 20% ($\epsilon = 0.2$)

Figure 5.9 shows the correlation plot of the resulting automaton-ABC posterior. The parallel run (120 jobs) performed 533003 simulations and lasted 268 seconds. We can see that the different parameters have structured correlations: the three histograms have parabolic shapes. Also, one can notice that for each histogram, the area near the point $(1, 1)$ is a high probability area. This is consistent with the above one-dimensional experiment.

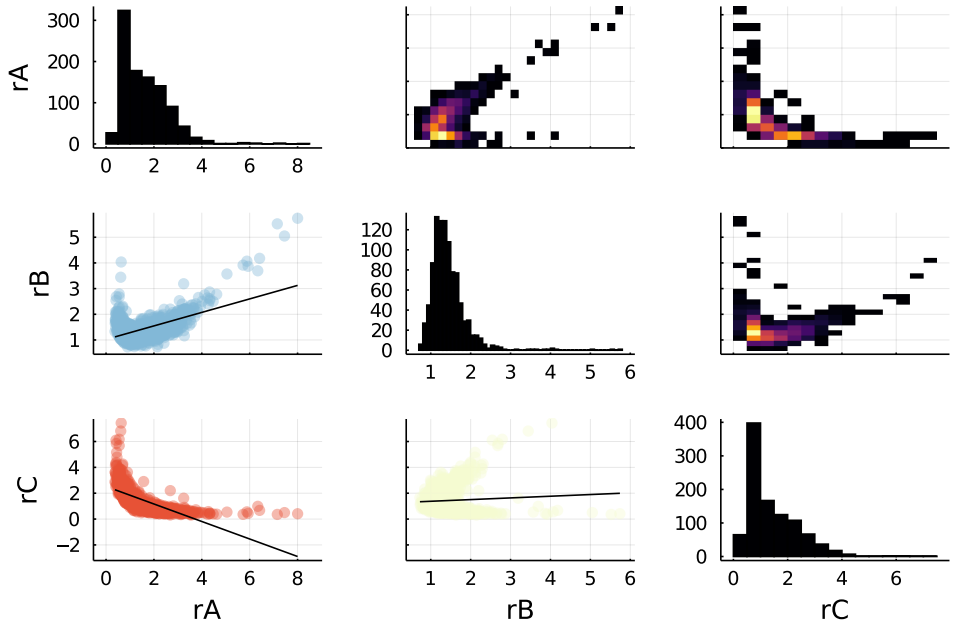


FIGURE 5.9: Correlation plot of automaton-ABC posterior with $\mathcal{A}per$ automaton for the 3D experiment of doping 3-way oscillator.

Repressilator model

Model. Repressilator models are synthetic genetic networks that describe a cycle of three genes with a negative feedback loop (Purcell et al., 2010). In this part, we consider one of the first repressilator model developed by (Elowitz, Leibler, and Leibier, 2000). This model includes three genes *lacI, tetR* and *cI* (called $G_i, i \in \{1, 2, 3\}$ in the equations) and their corresponding proteins *LacI, TetR* and *CI* (called $P_i, i \in \{1, 2, 3\}$). $mRNA_i$ is the messenger RNA corresponding to the gene G_i.

This synthetic genetic network was developed to reproduce oscillatory behaviours within a cell. It is based on a three-gene circuit illustrated in Figure 5.10: each protein represses the transcription of the successor gene. Protein P_1 represses the transcription of the gene G_2 (i.e. represses the production of $mRNA_2$), whereas P_2 represses the transcription of the gene G_3, and so on. This mechanism is commonly called a *negative feedback loop*; it leads to oscillatory behaviours.

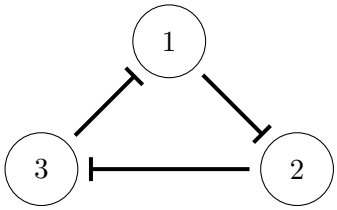


FIGURE 5.10: Repressilator topology. Each protein represses the transcription of the successor.

Equations 5.8, 5.9 and 5.10 below constitute a possible reduced description of the processes involved in the repressilator dynamics.

$$tr1 : G_1 \xrightarrow{r_1} G_1 + mRNA_1 \qquad tr2 : G_2 \xrightarrow{r_2} G_2 + mRNA_2 \qquad tr3 : G_3 \xrightarrow{r_3} G_3 + mRNA_3 \tag{5.8}$$

$$trl1 : mRNA_1 \xrightarrow{\beta} mRNA_1 + P_1 \quad trl2 : mRNA_2 \xrightarrow{\beta} mRNA_2 + P_2 \quad trl3 : mRNA_3 \xrightarrow{\beta} mRNA_3 + P_3 \tag{5.9}$$

$$degr1 : mRNA_1 \xrightarrow{1} \emptyset \qquad degr2 : mRNA_2 \xrightarrow{1} \emptyset \qquad degr3 : mRNA_3 \xrightarrow{1} \emptyset$$
$$degr4 : P_1 \xrightarrow{1} \emptyset \qquad degr5 : P_2 \xrightarrow{1} \emptyset \qquad degr6 : P_3 \xrightarrow{1} \emptyset \tag{5.10}$$

The kinetic rates are:

- $r_1 = \frac{\alpha}{1+[P_3]^n} + \alpha_0$, $r_2 = \frac{\alpha}{1+[P_1]^n} + \alpha_0$ and $r_3 = \frac{\alpha}{1+[P_2]^n} + \alpha_0$ for the transcription reactions,
- mass-action law for the translation reactions (i.e. $\beta \cdot [mRNA_i]$),
- 1 for the degradation reactions (independent of the concentrations of P_i).

These reaction channels do not follow mass-action law. Indeed they use a reparametrisation of the model detailed in (Elowitz, Leibler, and Leibier, 2000, Box 1), which is convenient to study the dynamics of the related Reaction Rate Equations of the CRN (continuous ODE system):

- β is the ratio of the protein decay rate over the mRNA decay rate. Equations 5.9 are thus parametrised by β, whereas degradation Equations 5.10 are not parametrised.
- n is the Hill coefficient: it measures the interaction affinity between species, i.e. how much the ligands will bind to create macromolecules.
- α is the parameter related to transcription growth.
- α_0 is the parameter related to a minimum level of transcription growth. Indeed $r_i \underset{[P_{i+1}]\to+\infty}{\to} \alpha_0$, which means α_0 models a saturation of the repressors' effect.

With Equations 5.8 5.9 5.10, we neglect several aspects of molecular interactions:

- Binding and unbinding of RNA polymerase with promoters, which is a necessary phenomenon for any transcription, is implicit in Equations 5.8.

- The observation of one protein species is initially measured in (Elowitz, Leibler, and Leibier, 2000) by a green fluorescent protein (*GFP*), which has the role of a reporter for the *TetR* protein. Our model supposes that we directly observe the protein (or the link between *GFP* and *TetR* is known and invertible).

Simulations. To better understand the system's dynamics, we present a few simulations. Default parameter values are $(\alpha, \beta, n, \alpha_0) = (200.0, 2.0, 2.0, 0.0)$.

Figure 5.11 and Figure 5.12 show simulations with different parameter values of α and β. It illustrates trajectories with different amplitudes and period duration, but oscillations are pretty stable.

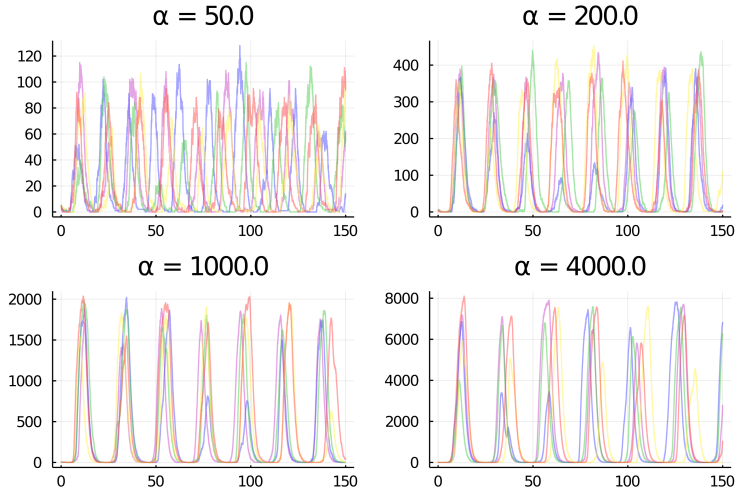


FIGURE 5.11: 5 simulated trajectories of the species P_1. α varies in $\{50, 200, 1000, 4000\}$.

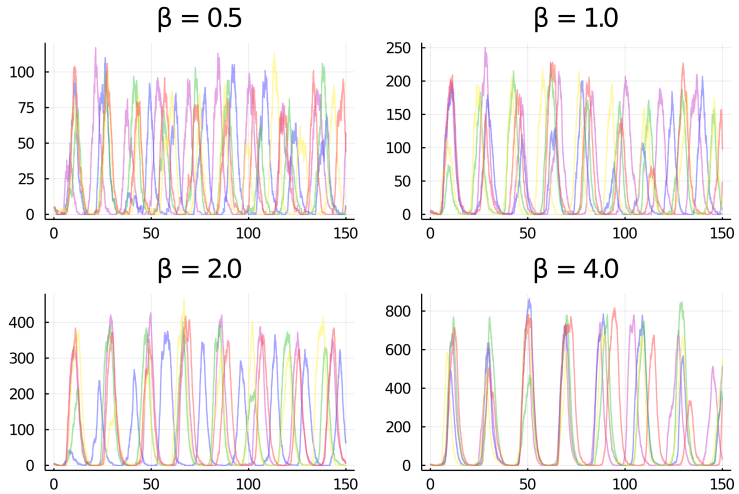


FIGURE 5.12: 5 simulated trajectories of the species P_1. β varies in $\{0.5, 1.0, 2.0, 4.0\}$.

Figure 5.13 shows simulations with different parameter values of n. This parameter has more influence on the stability of oscillations since simulations with $n = 0.5$ and $n = 1.0$ are very noisy.

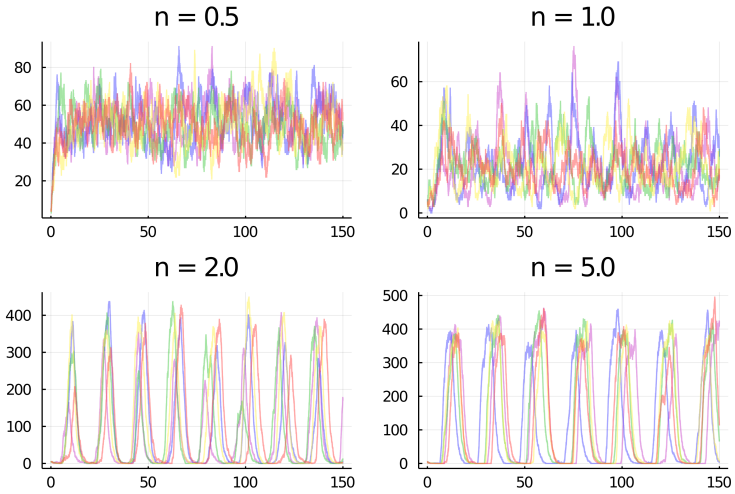


FIGURE 5.13: 5 simulated trajectories of the species P_1. n varies in $\{0.5, 1.0, 2.0, 5.0\}$.

Last, Figure 5.14 shows simulations with different parameter values of α_0. Oscillations with $\alpha_0 = 0.0$ or $\alpha_0 = 0.01$ are quite stable. With $\alpha_0 = 0.1$, stability gets worst whereas simulations with $\alpha_0 = 1.0$ oscillations are much noisier. This remark is consistent with the stability diagram from (Elowitz, Leibler, and Leibier, 2000, Figure 1a.). Considering the continuous ODE model, they deduced that the system is steady-state unstable if $\frac{\alpha}{\alpha_0} \geq 10^{-3}$, which is the case for $\alpha_0 = 1.0$ (as $\alpha = 200.0$).

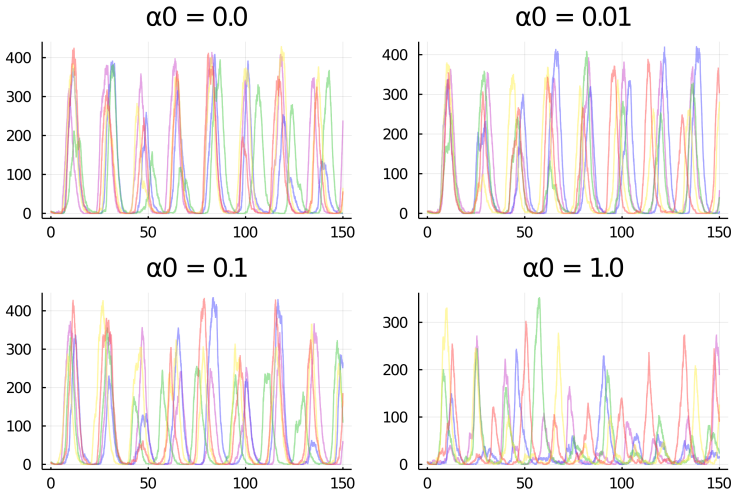


FIGURE 5.14: 5 simulated trajectories of the species P_1. α_0 varies in $\{0.0, 0.01, 0.1, 1.0\}$.

Results. We apply the automaton-ABC algorithm with the synchronised simulation $(Last(d), \mathcal{A}per)$. We want to find the subset of parameters for which oscillations have a period duration mean of 20.0.

We did two experiments. The first is three-dimensional, the setting is:

- $N_{per} = 4$ periods, $L = 50$ and $H = 200$
- $\bar{t_p}^{(obs)} = 20.0$
- $\alpha \sim \mathcal{U}(50, 5000)$, $\beta \sim \mathcal{U}(0.1, 5.0)$, $n \sim \mathcal{U}(0.5, 5.0)$ $(\alpha_0 = 0.0)$.
- $\mathbf{s}_0 = ((mRNA_1)_0, (mRNA_2)_0, (mRNA_3)_0, (P_1)_0, (P_2)_0, (P_3)_0) = (0, 0, 0, 5, 0, 15)$
- $N = 1000$ particles
- Tolerance of 10% $(\epsilon = 0.1)$

Figure 5.15 shows the correlation plot of the resulting automaton-ABC posterior. The parallel run (250 jobs) performed 15333 simulations and lasted 2040 seconds. The parameter n has the shrinkest marginal posterior, consistent with the simulation study detailed above: varying n induces more instability than α and β.

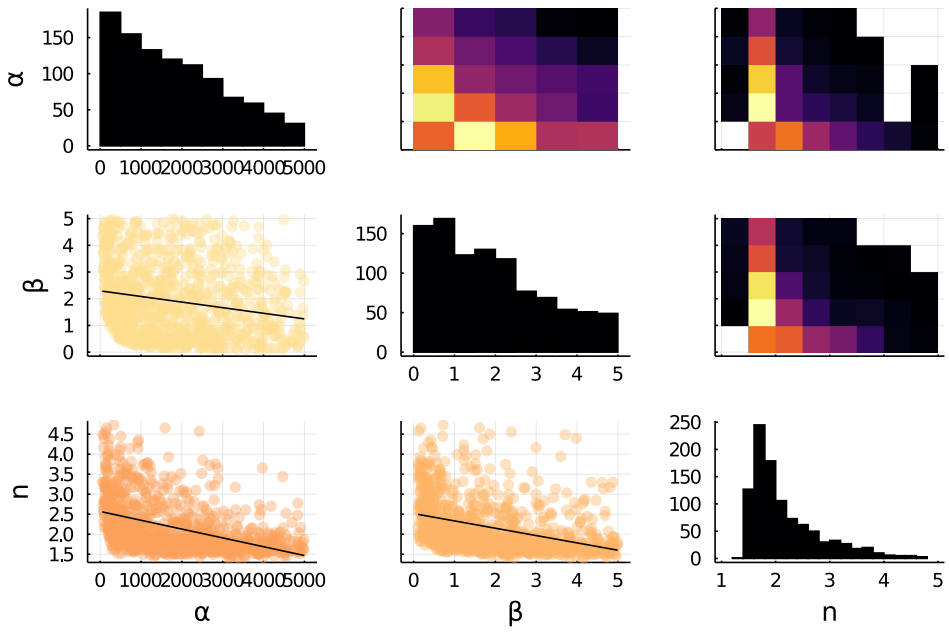


FIGURE 5.15: Correlation plot of automaton-ABC posterior with $\mathcal{A}per$ posterior for the 3D experiment of repressilator model.

The second experiment is four-dimensional. The setting is:

- $N_{per} = 4$ periods, $L = 50$ and $H = 200$

- $\bar{t_p}^{(obs)} = 20.0$
- $\alpha \sim \mathcal{U}(50, 5000)$, $\beta \sim \mathcal{U}(0.1, 5.0)$, $n \sim \mathcal{U}(0.5, 5.0)$, $\alpha_0 \sim \mathcal{U}(0.0, 5.0)$.
- $\mathbf{s}_0 = ((mRNA_1)_0, (mRNA_2)_0, (mRNA_3)_0, (P_1)_0, (P_2)_0, (P_3)_0) = (0, 0, 0, 5, 0, 15)$
- $N = 1000$ particles
- Tolerance of 10% ($\epsilon = 0.1$)

Figure 5.16 shows the correlation plot of the resulting automaton-ABC posterior. The parallel run (250 jobs) performed 22291 simulations and lasted 491 seconds. One can see adding a degree of freedom on α_0 has changed the correlation between α and β as well as α and n, whereas the correlation between β and n seems to have the same shape.

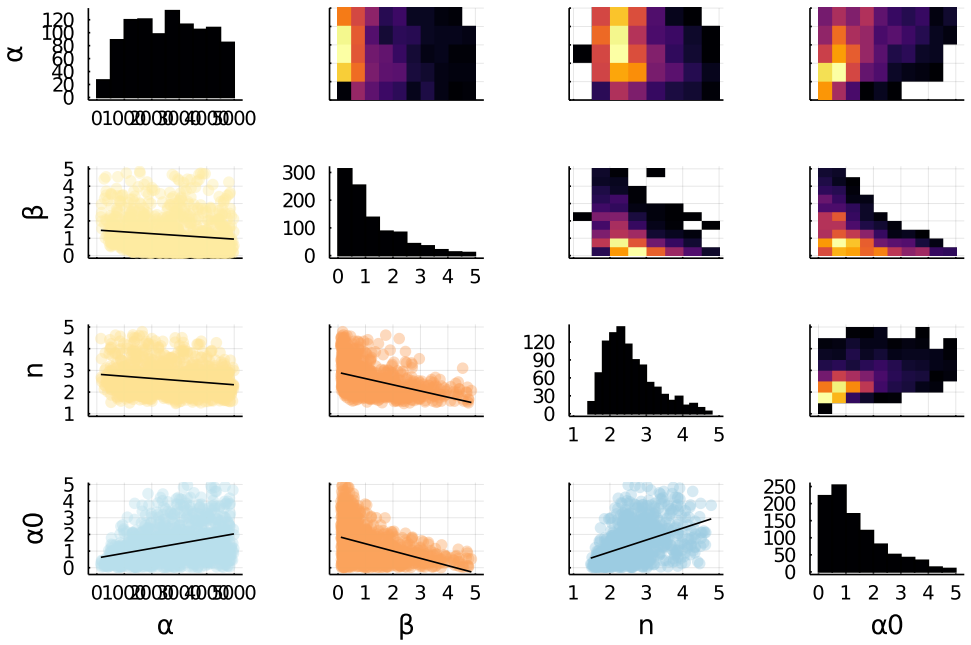


FIGURE 5.16: Correlation plot of automaton-ABC posterior with *Aper* posterior for the 4D experiment of repressilator model.

Script 5 (Oscillatory trends of parametric CTMC)

The scripts about oscillations of the doping 3-way oscillator and repressilator can be found in the git repository of the thesis at `code/chap4/oscillators`.

5.4 Accelerating the ABC procedure with HASL formalism

This section presents an automaton that aims to speed up the general ABC procedure used in Bayesian statistical inference.

In complex models, one simulation can be quite computationally expensive depending on the time window. The repressilator model is an illustrative example (cf Section 5.3.2). As ABC methods are based on a lot of model simulations, the convergence of ABC methods is directly linked to the cost of simulations.

ABC procedures simulate the model until the distance between a simulation and the observations are lower than a tolerance ϵ. However, this tolerance can be reached before the simulation ends.

Let $(t_i, y_i)_{1 \leq i \leq K}$ be time-discrete observations of a one-dimensional trajectory and let σ be a path of a CTMC model. The Euclidean distance between the observations and the simulation is:

$$\sqrt{\sum_{i=1}^{K} (y_i - \sigma@t_i)^2}$$

If the simulation σ differs a lot from the observations, it may exist $i^* < K$ with $\sqrt{\sum_{i=1}^{i^*} (y_i - \sigma@t_i)^2} > \epsilon$. One can deduce that, within the ABC procedure, the simulation could have been stopped at time t_{i^*} instead of simulating until t_K.

This section presents a Linear Hybrid Automaton called $\mathcal{A}_{ABC,\epsilon}$ that computes the Euclidean distance between a trajectory and observations. The synchronised simulation with $\mathcal{A}_{ABC,\epsilon}$ calculates the Euclidean distance to the observations on the fly of the simulation. It terminates if the distance exceeds the tolerance ϵ or time reaches t_K. The main goal is to save computation time in ABC procedures by cutting off useless simulation steps when the simulation already exceeds the tolerance ϵ before t_K.

5.4.1 Automaton $\mathcal{A}_{ABC,\epsilon}$

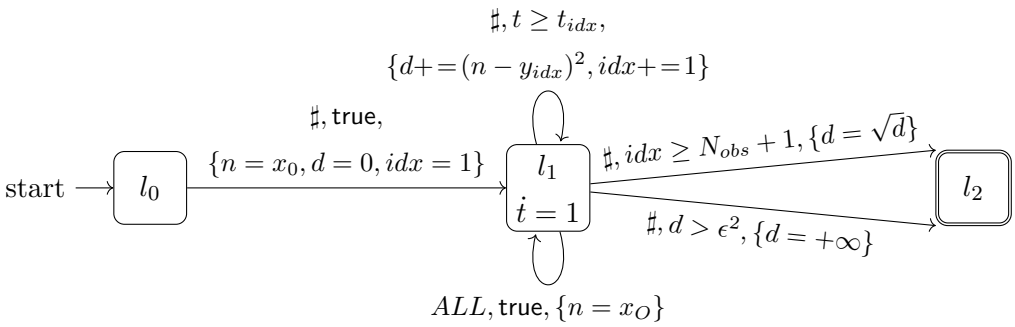


FIGURE 5.17: Automaton $\mathcal{A}_{ABC,\epsilon}$.

Figure 5.17 shows the $\mathcal{A}_{ABC,\epsilon}$ Linear Hybrid Automaton. The automaton variables are:

- n is the population of the observed species O,
- $(t_i, y_i)_{1 \leq i \leq K}$ are the one-dimensional time-discrete observations,
- $idx \in \{1, \ldots, K\}$ is the index of the next observation (y_{idx}) to add,
- ϵ is the tolerance level of the computed distance.

For any synchronised simulation, first, the distance d is initialised to 0, and idx is initialised to 1 because the next observation to add is y_1. It corresponds to the firing of the transition $l_0 \to l_1$.

While in l_1, for any occurred reaction, n is updated (synchronous bottom self loop). If time t reaches t_{idx}, the distance d is updated by adding $(n - y_{idx})^2$ and idx is incremented by one (asynchronous top self edge loop). Note that while in l_1, d is the square of the Euclidean distance.

The simulation finishes in l_2 in two ways. Either the distance has exceeded the tolerance ϵ ($d > \epsilon^2$) and bottom edge $l_1 \to l_2$ fires, or all the observations have been counted and $l_1 \to l_2$ top edge fires.

5.4.2 Applications

To illustrate the potential gain of computational time for automaton-ABC with $\mathcal{A}_{ABC,\epsilon}$, we consider a challenging task of inference with the repressilator model inspired from data exposed in (Elowitz, Leibler, and Leibier, 2000).

We considerer a simulated dataset of a 40-points observation of the population of the species P_1 from $t_1 = 0.0$ to $t_{40} = 400.0$ with parameters $(\alpha, \alpha_0, \beta, n) = (200.0, 0.0, 5.0, 4.0)$, which is plotted in Figure 5.18.

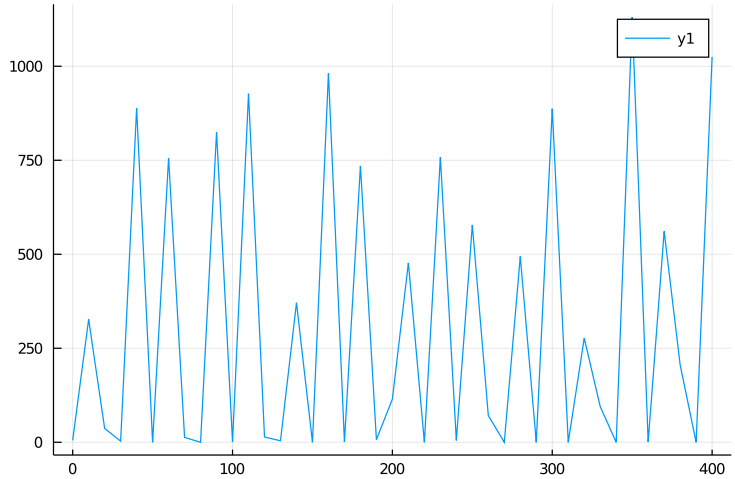


FIGURE 5.18: 40-points observation of the species P_1 from repressilator model over $[0.0, 400.0]$ with a time step of 10.0.

In the following, we run two parallel ABC-like procedures with 80 jobs on HPC resources from the "Mésocentre":

- Classical ABC-SMC: at each iteration, we simulate until t_K and then computes the Euclidean distance from the observations.
- Automaton-ABC-SMC with $(Last(d), \mathcal{A}_{ABC,\epsilon})$.

For both ABC and automaton-ABC runs, the setting is:

- $N = 500$ particles,
- $\beta \sim \mathcal{U}(0.5, 10.0), n \sim \mathcal{U}(0.5, 10.0)$ ($\alpha = 200.0$ and $\alpha_0 = 0.0$ are fixed),
- Tolerance of 50% relative error $\epsilon = 0.5||y_{exp}||$.

Figure 5.19 shows the ABC posteriors for both ABC procedures. These two posteriors are very similar, which means that the two procedures behaved in the same way, as expected.

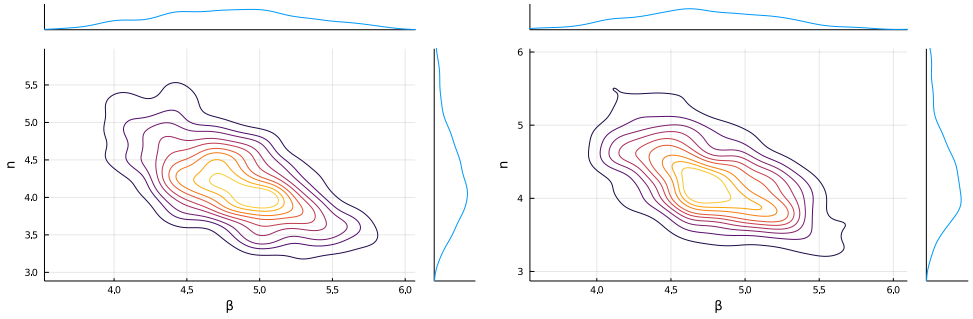


FIGURE 5.19: Contour plot of ABC posteriors with marginals. Left: classical ABC posterior. Right: automaton-ABC posterior with $\mathcal{A}_{ABC,\epsilon}$.

Table 5.4 shows the execution time of both runs. The automaton-ABC run was at least three times faster than the classical ABC procedure.

	Number of jobs	Number of simulations	Execution time (sec)
Automaton-ABC	80	2300706	16294
Classical ABC	80	2554773	54049

TABLE 5.4: Execution times of statistical inference with classical ABC compared to automaton-ABC with $\mathcal{A}_{ABC,\epsilon}$

With a longer timeline of observations or a more complex model, the saved computational time could be even greater.

Script 6 (Example CTMC)
This experiment can be found in the git repository of the thesis at `code/chap4/reject_abc_automaton`.

5.5 Summary

In this Chapter:

- We have detailed the statistical framework of CTMCs, the different types of observations, and distances between observations. We have discussed the available techniques for statistical inference of CTMCs.
- We have detailed automaton-ABC, a new ABC procedure based on synchronised simulations of a CTMC with a Linear Hybrid Automaton.

- We have applied the automaton-ABC procedure on two main tasks. First, we describe oscillatory trends of models such as repressilator with the help of $\mathcal{A}_{per}$. Second, we have speeded up classical ABC inference with the help of $\mathcal{A}_{ABC,\epsilon}$.

The next chapter tackles the estimation problem in model checking parametric CTMC (Section 4.3.1) with the help of automaton-ABC.

Chapter 6

Automaton-ABC for the statistical parametric verification of CTMCs

Approximate Bayesian Computation (ABC) methods (Section 3.3) allow approximating the posterior distribution of a model without evaluating the likelihood function when the computation cost is too high or even impossible. These methods rely on model simulations and a discrepancy between simulations and observations. Simply speaking, only parameters for which simulated summary statistics are close to observed ones, relative to a distance, are preserved while the others are dismissed. These parameters are sampled from the ABC posterior, which approximates the true posterior distribution.

This chapter proposes a new method based on the ABC algorithm for the estimation problem of parametric CTMC in verification, described in Section 4.3.1. We want to estimate the satisfaction probability function given a *time-bounded reachability* specification φ related to a parameter θ. The initial idea of this work relies on transposing this concept originally developed with observations data to temporal logic: only parameters for which simulations are close to fulfilling a given formula are kept while the others are discarded.

Thus, the last statement supposes that one can measure the *discrepancy* between a simulation and a specification φ. We formally define a *satisfiability distance* of a trajectory of a CTMC from a logical property φ and use Linear Hybrid Automata to compute this distance. This distance is integrated within the ABC procedure to estimate the subset of the parameter space in which the logical property can be satisfied thanks to the obtained ABC posterior. We show that the sequential version of ABC (Algorithm 18) is well suited to using the distance from a property,

leading to an efficient exploration of the parameter space. Our method also allows the estimation of the satisfaction function of a formula, with a remarkable result linking this satisfaction function to the ABC posterior (Theorem 6.4.1). We demonstrate the effectiveness of our method by its application to several models of Chemical Reaction Networks (CRN).

This Chapter is organised as follows. In Section 6.2, the notion of satisfiability distance for time-bounded reachability problems is introduced. We see in Section 6.3 how simulations synchronised with Linear Hybrid Automata compute the satisfiability distances. The automaton-ABC procedure with LHA related to satisfiability distances is detailed in Section 6.4: the objective is to find the parameter subspace for which the probability of reaching the target region is positive. Its effectiveness is demonstrated through several experiments in Section 6.5 and Section 6.6, while some conclusive remarks and perspectives are discussed in Section 6.7.

6.1 Problem setting: time-bounded reachability

In this Chapter, we tackle the estimation problem detailed in Section 4.3.1. We consider a parametric CTMC $(\mathcal{M}_\theta)_{\theta\in\Theta}$ (Section 4.3.1) with a state space of dimension d. This work is focused on CTMCs whose states represent the number of individuals of several species, which is also called Markov Population Processes (see Remark 2.3.1). φ is a specification described by an MITL formula (Definition 4.1.1). Our goal is to estimate the satisfaction probability function f_φ (Definition 4.3.2):

$$\begin{aligned} f_\varphi : \Theta &\to [0,1] \\ \theta &\to Pr(\varphi; \mathcal{M}_\theta) \end{aligned}$$

where $Pr(\varphi; \mathcal{M}_\theta)$ (Definition 4.2.1) is the probability that a trajectory of $\mathcal{M}_\theta$ verifies φ.

In particular, we consider specifications φ related to time-bounded reachability. The term *reachability problem* identifies the class of problems that checks if a given model *reaches* (i.e. enters) at some point during its execution a specific region of its state space. For models that produce time series, *time-bounded reachability* aims to establish whether the state space's target region is entered within a time-interval $[t_1, t_2] \subset \mathbb{R}_{\geq 0}$.

Based on MITL formulae, we distinguish three kinds of time-bounded reachability problems:

- *Eventual reachability* problems ($\mathbf{F}^I\mu$) are concerned with model trajectories that fulfil μ at least once within a given time interval I.
- *Global reachability* problems ($\mathbf{G}^I\mu$) are concerned with trajectories that consistently fulfil μ within a time interval I.
- *Conditional reachability* problems ($\mu_1\mathbf{U}^I\mu_2$) are concerned with trajectories that fulfil μ_2 at least once within a time interval I and fulfil μ_1 beforehand.

Each kind of problem induces a satisfiability region that characterises the trajectories verifying the corresponding formula.

Definition 6.1.1 (CTMC region)
Let $\mathcal{M}$ a d-dimensional CTMC. A region R of the CTMC $\mathcal{M}$ is a collection of subsets of $\mathbf{S} \subseteq \mathbb{N}^d$.

Definition 6.1.2 (CTMC time-bounded region)
Let $\mathcal{M}$ a d-dimensional CTMC. A time-bounded region $TR = R \times T$ is the cartesian product of a region R of $\mathcal{M}$ with a time set $T \subset \mathbb{R}_{\geq 0}$.

A region is *elementary* if it is characterised by a single subset of $\mathbf{S}$. We denote $[[p, q]]$ the set of integers $\{p, \ldots, q-1, q\}$. For example, let a bi-dimensional CTMC S with a state space $\mathbf{S} = \mathbb{N}^2$. $R_1 = [[1, 2]] \times \mathbb{N}$ is an elementary region ($S_{[1]}$ in $[[1, 2]]$ while $S_{[2]}$ is unbounded), $R_2 = ([[0, 3]] \cup [[5, 8]]) \times [[5, \infty[[$ is a non-elementary region ($S_{[1]}$ is either in $[[0, 3]]$ or $[[5, 8]]$, $S_{[2]}$ is larger than 5), whereas $TR_1 = ([[1, 2]] \times \mathbb{N}) \times [0.2, 1.41]$ is an elementary time-bounded region (similar to R_1, but with the supplemental condition that the time is in $[0.2, 1.41]$).

6.2 Satisfiability distances

The adaptation of the parameter space exploration by ABC algorithms to reachability problems requires the definition of a *satisfiability distance* $d(\sigma, \varphi)$. This distance represents how far a trajectory $\sigma \in Path(\mathcal{M})$ is from satisfying an MITL formula φ. Note that we call it a distance to keep up with the ABC vocabulary: it is not mathematically strictly a distance function, but rather a discrepancy. Indeed, (σ, φ) is not a pair of elements of the same set.

Definition 6.2.1 (Satisfiability distance)
Let φ an MITL formula. f is a satisfiability distance if $\forall \sigma, f(\sigma, \varphi) \geq 0$ and $f(\sigma, \varphi) = 0 \Leftrightarrow \sigma \models \varphi$

MITL propositional formulae $\mu \in AP$ induce CTMC regions called *satisfiability region.*

Definition 6.2.2 (Satisfiability region of μ)
A satisfiability region $\mathbf{S}_\mu$ *of a propositional formula* μ *is a CTMC region (Definition 6.1.1) where* μ *is fulfilled, i.e.* $\mathbf{S}_\mu = \{\mathbf{s} \in \mathbf{S}, \mid \mathbf{s} \models \mu\}$.

Definition 6.2.3 (Time-bounded satisfiability region of μ)
A time-bounded satisfiability region ${\mathbf{S}_\mu}^{[t_1,t_2]}$ *is a time-bounded region (Definition 6.1.2)* $\mathbf{S}_\mu \times [t_1, t_2]$, *with* $\mathbf{S}_\mu$ *a satisfiability region.*

For example, for a bi-dimensional CTMC, the formula $\mu_1 = x_1 \geq 1 \wedge x_1 \leq 2$ induces the satisfiability region $R_1 = [[1, 2]] \times \mathbb{N}$ while $\mu_2 = [(x_1 \leq 3) \vee (x_1 \geq 5 \wedge x_1 \leq 8)] \wedge x_2 > 4$ induces the satisfiability region $R_2 = ([[0, 3]] \times [[5, +\infty[) \cup ([[5, 8]] \times [[5, +\infty[)$. By a slight abuse of vocabulary, we identify μ with its satisfiability region, and we say that two formulae μ_1, μ_2 are *disjoint* if their corresponding regions are.

Let μ be a propositional formula. We introduce the notion of *satisfiability distance* of a trajectory from a temporal formula build on top of μ over $[t_1, t_2]$. These distances are related to the time-bounded reachability problems described in Section 6.1. The following definitions hold for any trajectory σ, whose last jump occurred at time $t_{last} \leq t_2$, where $t_{last} = \min\{t \in \mathbb{R}_{\geq 0}, \forall t' \in [t, t_2], \sigma@t' = \sigma@t\}$. Indeed, jumps after t_2 do not affect a reachability problem over $[t_1, t_2]$. In the following, $d_e(\mathbf{s}, \mathbf{S}_\mu) = \min_{\mathbf{s}' \in \mathbf{S}_\mu} \sqrt{\sum_{i=1}^{d} (\mathbf{s}[i] - \mathbf{s}'[i])^2}$ is the least Euclidean distance between $\mathbf{s}$ and $\mathbf{S}_\mu$, whereas $d_e((t, \mathbf{s}), {\mathbf{S}_\mu}^{[t_1,t_2]})$ is the least Euclidean distance between $(t, \mathbf{s})$ and ${\mathbf{S}_\mu}^{[t_1,t_2]}$.

Definition 6.2.4 (Distances from several MITL formulas)
Let $\mathcal{M}$ *a CTMC,* $[t_1, t_2] \subseteq \mathbb{R}_{\geq 0}$. *Let* $\sigma \in Path(\mathcal{M})$ *a trajectory observed until* t_2, *with* $t_{last} = \min\{t \in \mathbb{R}_{\geq 0}, \forall t' \in [t, t_2], \sigma@t' = \sigma@t\}$. *We define the distance* $d(\sigma, \varphi)$ *for the following kinds of temporal formulae* φ *built on top of an elementary propositional formula* μ*:*

- $\varphi = \mathbf{F}^{[t_1,t_2]}\mu$

$$d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu) = \begin{cases} d_e((t_{last}, \sigma@t_{last}), {\mathbf{S}_\mu}^{[t_1,t_2]}) & \text{if } t_{last} < t_1 \text{ and } \sigma@t_{last} \notin \mathbf{S}_\mu \\ \min\limits_{t \in [t_1,t_2]} d_e(\sigma@t, \mathbf{S}_\mu) & \text{elsewhere} \end{cases} \tag{6.1}$$

For non-elementary propositional formulae $\mu = \bigvee \mu_i$, *we define the distance:*

$$d(\sigma, \mathbf{F}^{[t_1,t_2]} \bigvee \mu_i) = \min_i d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu_i)$$

where μ_i are elementary formulae.

- $\varphi = \mathbf{G}^{[t_1,t_2]}\mu$

$$d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu) = \int_{t_1}^{t_2} d_e(\sigma@t, \mathbf{S}_\mu)dt \tag{6.2}$$

Similarly, for non-elementary propositional formulae, we define the distance

$$d(\sigma, \mathbf{G}^{[t_1,t_2]} \bigvee \mu_i) = \min_i d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu_i)$$

where μ_i are elementary formulae.

- $\varphi = \mu_1 \mathbf{U}^{[t_1,t_2]} \mu_2$

$$\begin{aligned} d(\sigma, \mu_1 \mathbf{U}^{[t_1,t_2]}\mu_2) = & d(\sigma, \mathbf{G}^{[0,t_1]}\mu_1) + d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu_2) + \\ & d(\sigma, \mathbf{G}^{[t_1,t_{min}]}(\mu_1 \vee \mu_2)) \end{aligned} \tag{6.3}$$

where $t_{min} = \min(\arg\min_{t\in[t_1,t_2]} d_e(\sigma@t, \mathbf{S}_{\mu_2}))$ is the earliest time corresponding to the closest point between σ and region μ_2.

These distances are illustrated on Figure 6.2.

$d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu)$ is a piecewise function. The first condition $t_{last} < t_1$ and $\sigma@t_{last} \notin \mathbf{S}_\mu$ means that the last transition/jump of the trajectory occurs before t_1, and the last state of the trajectory does not belong to $\mathbf{S}_\mu$. In this case, the satisfiability distance computes the Euclidean distance between $(t_{last}, \sigma@t_{last})$ and $\mathbf{S}_\mu{}^{[t_1,t_2]}$. This is related to the choice of ranking trajectories when no reactions occur before t_1. Indeed, the efficiency (but not correctness) of our new method for the estimation of Definition 4.3.2 relies on a particular aspect of satisfiability distances, which is not included in the definition. It should compare trajectories fairly by how far they are from satisfying the formula. Suppose $\sigma_1, \sigma_2 \in Path(\mathcal{M})$. If σ_1 is further to fulfil φ than σ_2, then the distance should verify $d(\sigma_1, \varphi) > d(\sigma_2, \varphi)$ for better efficiency.

An example of a trajectory is depicted in Figure 6.1. d_1 is the satisfiability distance of the violet trajectory because no reactions occur after t_1. d_2 would be the computed distance by the second sub-function in Equation 6.1. The closer t_{last} is to t_1, the smaller d_1. But d_1 is consistently greater than d_2. This makes a good ranking of trajectories in which no reaction occurs after t, $t < t_1$. This is particularly the case when a trajectory reaches an absorbing state before t_1: it implies no possible evolution of the system after t_{last} that could make the trajectory fulfil the formula. In this case, the further t_{last} is from t_1, the further the trajectory is from satisfying the formula.

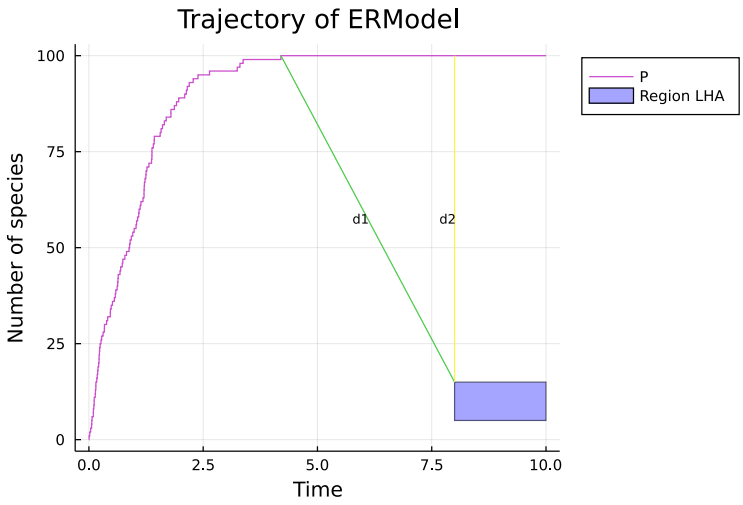


FIGURE 6.1: Example of a simulation. In purple: the trajectory. In blue and green: the distances d_1 and d_2. The rectangle is the F region.

.

Otherwise, the distance $d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu)$ is the minimal Euclidean distance of any point of σ within $[t_1, t_2]$ from $\mathbf{S}_\mu$.

$d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu)$ is the integral of the Euclidean distance from $\mathbf{S}_\mu$ of any point of σ within $[t_1, t_2]$. The bigger the distance, the more the trajectory is globally outside of $\mathbf{S}_\mu$ over $[t_1, t_2]$.

Concerning the Until formula, the distance (6.3) bears three components:

- $d(\sigma, \mathbf{G}^{[0,t_1]}\mu_1)$ accounts for the fact that a trajectory satisfying $(\mu_1 \mathbf{U}^{[t_1,t_2]}\mu_2)$ must never leave region μ_1 before t_1.
- $d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu_2)$ accounts for the fact that σ must enter region μ_2 within $[t_1, t_2]$
- $d(\sigma, \mathbf{G}^{[t_1,t_{min}]}(\mu_1 \vee \mu_2))$ accounts for the fact that there must be a time t_{min} where σ switches from region μ_1 to region μ_2 directly, without spending time in an intermediate region. If that is not the case (i.e. if σ within $[t_1, t_2]$ has points in the complementary region $\neg(\mu_1 \vee \mu_2)$), then the distance 6.3 is incremented by this term that quantifies how far we are from one of the two regions μ_1 or μ_2.

Remark 6.2.1 (Until distance)
In the following, we only designed an LHA ($\mathcal{A}_{G\wedge F}$) for computing the Until distance when $\mu_1 \veebar \mu_2$ (the exclusive disjunction) is always true over any state of the CTMC, which implies the third term of the distance is zero.

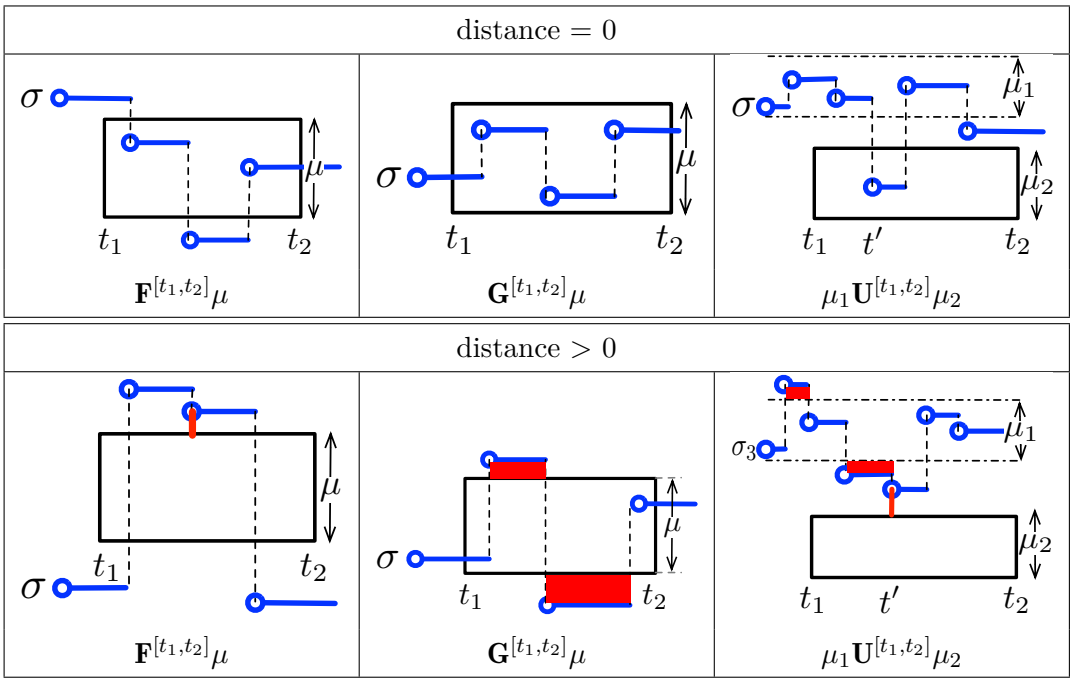


FIGURE 6.2: Examples of trajectories with zero-distance (top) and positive distance (bottom) from an **F**, a **G** and a **U** region (positive distances are depicted in red).

The correctness of the satisfiability distances introduced above is stated with the following proposition.

Proposition 6.2.1 (Soundness of distances from Definition 6.2.4)
Let $\sigma \in Path(\mathcal{M})$ be a path of a CTMC $\mathcal{M}$, and φ an MITL temporal that has the form $\mathbf{F}^{[t_1,t_2]}\mu$, $\mathbf{G}^{[t_1,t_2]}\mu$ or $\mu_1\mathbf{U}^{[t_1,t_2]}\mu_2$ with μ, μ_1, μ_2 propositional formulae. Let d the distance from Definition 6.2.4. Then:

$$\sigma \models \varphi \iff d(\sigma, \varphi) = 0$$

d is a satisfiability distance.

We should first prove this result for $\mathbf{F}^{[t_1,t_2]}\mu$ and $\mathbf{G}^{[t_1,t_2]}\mu$ where μ is elementary.

Lemma 6.2.1
For an elementary proposition μ and an MITL temporal formula $\varphi = \mathbf{F}^{[t_1,t_2]}\mu$, $\sigma \models \varphi \iff d(\sigma, \varphi) = 0$

Proof. $\Rightarrow$ Suppose that $\sigma \models \mathbf{F}^{[t_1,t_2]}\mu$. Let us prove $d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu) = 0$.

As $\sigma \models \varphi$, $\exists t^* \in [t_1, t_2], \mathbf{s}^* \in \mathbf{S}_\mu / \sigma@t^* = \mathbf{s}^*$. If $t_{last} < t_1$ and $\sigma@t_{last} \notin \mathbf{S}_\mu$, then $\forall t \in [t_1, t_2], \sigma@t \not\models \mu$, which is impossible.

Thus, by definition, $d_e(\sigma@t^*, \mathbf{S}_\mu) = \min_{\mathbf{s}' \in \mathbf{S}_\mu} \sqrt{\sum_{i=1}^{d} (\sigma@t^*[i] - \mathbf{s}'[i])^2}$. As $\mathbf{s}^* \in \mathbf{S}_\mu$, $d_e(\sigma@t^*, \mathbf{S}_\mu) = 0$. In fine, $d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu) = 0$.

$\Leftarrow$ Suppose that $d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu) = 0$. Let us prove $\sigma \models \mathbf{F}^{[t_1,t_2]}\mu$.

As $d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu) = 0$, there are two possible cases.

First case: $t_{last} < t_1$ and $\sigma@t_{last} \notin \mathbf{S}_\mu$. $d_e((t_{last}, \sigma@t_{last}), \mathbf{S}_\mu{}^{[t_1,t_2]}) = 0$ implies $t_{last} = t_1$: impossible.

Second case: $\exists t^* \in [t_1, t_2] / d_e(\sigma@t^*, \mathbf{S}_\mu) = 0$.

By definition, $d_e(\sigma@t^*, \mathbf{S}_\mu) = \min_{\mathbf{s}' \in \mathbf{S}_\mu} \sqrt{\sum_{i=1}^{d} (\sigma@t^*[i] - \mathbf{s}'[i])^2}$.
There exists $\mathbf{s}^* \in \mathbf{S}_\mu$ with $\sqrt{\sum_{i=1}^{d} (\sigma@t^*[i] - \mathbf{s}^*[i])^2} = 0$, which implies $\sigma@t^* = \mathbf{s}^*$. As $t^* \in [t_1, t_2]$ and $\sigma@t^* = \mathbf{s}^* \in \mathbf{S}_\mu$, the trajectory σ verifies $\mathbf{F}^{[t_1,t_2]}\mu$. □

Lemma 6.2.2
For an elementary proposition μ and an MITL temporal formula $\mathbf{G}^{[t_1,t_2]}\mu$, $\sigma \models \varphi \iff d(\sigma, \varphi) = 0$

Proof. $\Rightarrow$ Suppose that $\sigma \models \mathbf{G}^{[t_1,t_2]}\mu$. Let us prove $d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu) = 0$.

As $\sigma \models \mathbf{G}^{[t_1,t_2]}\mu$, $\forall t \in [t_1, t_2]$, $\sigma@t \in \mathbf{S}_\mu$. We have $\forall \mathbf{s}' \in \mathbf{S}_\mu$, $d_e(\mathbf{s}', \mathbf{S}_\mu) = 0$. So $\forall t \in [t_1, t_2], d_e(\sigma@t, \mathbf{S}_\mu) = 0$, hence $\int_{t_1}^{t_2} d_e(\sigma@t, \mathbf{S}_\mu)dt = 0$. In conclusion, $d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu) = 0$.

$\Leftarrow$ Suppose that $d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu) = 0$. Let us prove $\sigma \models \mathbf{G}^{[t_1,t_2]}\mu$.

We have: $\int_{t_1}^{t_2} d_e(\sigma@t, \mathbf{S}_\mu)dt = 0$. As $t \to d_e(\sigma@t, \mathbf{S}_\mu)$ is a non-negative continuous function and its integral equals 0, then this function is the null function over $[t_1, t_2]$. So, $\forall t \in [t_1, t_2], d_e(\sigma@t, \mathbf{S}_\mu) = 0$. This means $\forall t \in [t_1, t_2], \sigma@t \in \mathbf{S}_\mu$. In other words, $\sigma \models \mathbf{G}^{[t_1,t_2]}\mu$. □

Now we can tackle the proof of Proposition 6.2.1.

Proof.

$$
\begin{aligned}
d(\sigma, \mathbf{F}^{[t_1,t_2]} \bigvee \mu_i) = 0 &\Leftrightarrow \min_i d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu_i) = 0 \\
&\Leftrightarrow \exists i / d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu_i) = 0 \quad \big) \textit{ Result 6.2.1} \\
&\Leftrightarrow \exists i / \sigma \models \mathbf{F}^{[t_1,t_2]}\mu_i \quad \big) \textit{ Def. of disjunction} \\
&\Leftrightarrow \sigma \models \mathbf{F}^{[t_1,t_2]} \bigvee \mu_i
\end{aligned}
$$

$$
\begin{aligned}
d(\sigma, \mathbf{G}^{[t_1,t_2]} \bigvee \mu_i) = 0 &\Leftrightarrow \min_i d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu_i) = 0 \\
&\Leftrightarrow \exists i / d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu_i) = 0 \quad \big) \textit{ Result 6.2.2} \\
&\Leftrightarrow \exists i / \sigma \models \mathbf{G}^{[t_1,t_2]}\mu_i \quad \big) \textit{ Def. of disjunction} \\
&\Leftrightarrow \sigma \models \mathbf{G}^{[t_1,t_2]} \bigvee \mu_i
\end{aligned}
$$

For the Until formula, the result follows from $\sigma \models \mu_1 \mathbf{U}^{[t_1,t_2]}\mu_2 \Leftrightarrow \sigma \models \mathbf{G}^{[0,t_1]}\mu_1 \wedge \mathbf{G}^{[t_1,t_{min}]}\mu_1 \wedge \mathbf{F}^{[t_{min},t_2]}\mu_2$ with the proof of the distances for eventual and global regions. □

6.3 Linear Hybrid Automata to compute satisfiability distances

Based on the HASL formalism (Section 4.4), we discuss the definitions of hybrid automata to measure the satisfiability distances with synchronised simulations (Section 4.4.2). The automata presented here refer to temporal formulae built on top of a generic mono-dimensional (elementary) proposition $\mu = x_1 \leq x_O \leq x_2$ (where x_O denotes the population of an observable quantity O of a CTMC model $\mathcal{M}$, and $x_1 < x_2 \in \mathbb{N}$).

Each synchronised simulation of a trajectory is associated with the HASL trajectory expression $Last(d)$. The last value of the computed distance variable d is kept. The Linear Hybrid Automata update the distance on the fly of the simulation.

Remark 6.3.1 (Dashed red edge in the automata)
In each automaton, the dashed red edge corresponds to an asynchronous edge that is fired if the current state of the CTMC is absorbing, even if it does not fulfil the condition based on the time t.

More precisely $\xrightarrow{t \geq t_2 \wedge \gamma, U}$ *defines two edges:* $\xrightarrow{\sharp, t \geq t_2 \wedge \gamma, U}$ *and* $\xrightarrow{\sharp, E(x)=0 \wedge \gamma, U}$, *where $E(x)$ is the total exit rate of the state. This implies the definition of a variable test_abs that equal* `true` *if no more reaction can occur, which is updated at each*

reaction occurence by the total exit rate. It allows for the acceptance by the automaton of any trajectory, even if it has reached an absorbing state.

This variable will not be mentioned in the presentation of the three automata.

6.3.1 Distance automaton $\mathcal{A}_F$

Automaton $\mathcal{A}_F$ (Figure 6.3) is designed to measure the satisfiability distance (Equation 6.1) of trajectories of a CTMC model $\mathcal{M}$ from the time-bounded region associated with $\mathbf{F}^{[t_1,t_2]}(x_1 \leq x_O \leq x_2)$, i.e. the time-bounded satisfiability region $\mathbf{S}^{[t_1,t_2]}_{(x_1 \leq x_O \leq x_2)}$.

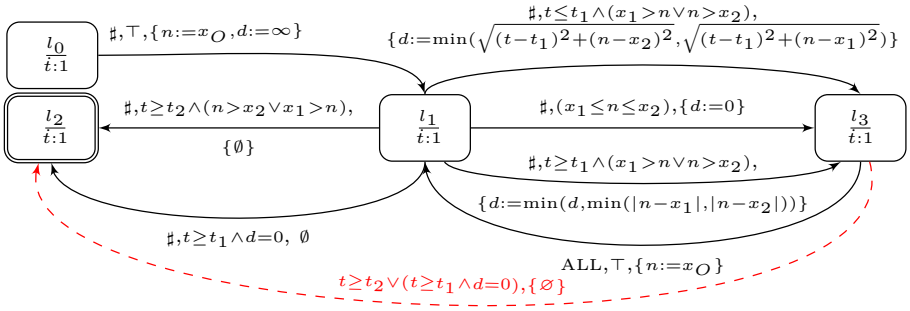


FIGURE 6.3: Automaton for eventual property $\mathcal{A}_F$.

Three variables are defined:

- d, the satisfiability distance,
- t, the current time,
- n, the population of the observed species O.

The initial location is l_0, and the final location is l_2. First, the distance is initialised to $d := \infty$ $(l_0 \to l_1)$, and the initial value of the observed species is stored in $n := x_O$.

Once in l_1, there are two ways of reaching the final location l_2. If the distance d is zero and time exceeds t_1, the trajectory has entered the time-bounded satisfiability region. Then, the transition $l_1 \xrightarrow{t \leq t_1 \wedge d=0, \emptyset} l_2$ fires. Otherwise, if the time exceeds t_2, the second transition to l_2 is traversed without updating the variables.

If, while in l_1, the trajectory has not entered the time-bounded satisfiability region yet, the distance d must be updated. It deals with three autonomous transitions $l_1 \to l_3$.

On the one hand, in case x_O has entered $[x_1, x_2]$, the distance is set to $d := 0$ ($l_1 \xrightarrow{(x_1 \leq n \leq x_2), \{d:=0\}} l_3$). Indeed, since CTMC trajectories are stepwise functions, if the next reaction occurs at time $t > t_1$, the current trajectory has at least one point within the time-bounded satisfiability region.

On the other hand, if x_O has not entered $[x_1, x_2]$, d is computed with two different equations. First, if $t \leq t_1$, $l_1 \xrightarrow{\sharp, (x_1 > n \vee n > x_2) \wedge (t \leq t_1), \{d:=\min(\sqrt{(t-t_1)^2+(n-x_2)^2}, \sqrt{(t-t_1)^2+(n-x_1)^2})\}} l_3$ fires: it computes the distance between the trajectory and the nearest corner of the eventual region. It corresponds to the first sub-function of the piecewise Equation 6.1.

Second, if $t \geq t_1$, the third transition fires. d is computed as the minimum distance between d and the distance of n to the bounds x_1 and x_2, which corresponds to the second sub-function of the piecewise Equation 6.1.

In l_3, two edges can be traversed. First, the asynchronous transition brings the automaton to the final location l_2 if either the distance is zero within $[t_1, t_2]$, time exceeds t_2 or the trajectory reaches an absorbing state. In addition, the transition $l_3 \xrightarrow{ALL, \mathsf{true}, \{n:=x_O\}} l_1$ is traversed whenever a new reaction occurs, which brings the automaton back to l_1 so that the distance can be updated accordingly.

Remark 6.3.2 (Left-closed constraints on transition predicates)
In the definition of an LHA, transition predicates of autonomous transitions have to be left closed linear constraints (Definition 4.4.2). But, for example $t \geq t_2 \wedge (n > x_2 \vee x_1 > n)$ *is not a left-closed constraint. Thus, the autonomous transition* $l_1 \xrightarrow{\sharp, t \geq t_1 \wedge (x_1 > n \vee n > x_2), d=...} l_3$ *defines, in fact, two autonomous transitions:* $l_1 \xrightarrow{\sharp, t \geq t_1 \wedge x_1 + 1 \geq n, d=...} l_3$ *and* $l_1 \xrightarrow{\sharp, t \geq t_1 \wedge n \geq x_2 + 1, d=...} l_3$ *as* n, x_1 *and* x_2 *are integers.*

Script 7 ($\mathcal{A}_F$ LHA in Cosmos)
An example of automaton $\mathcal{A}_F$ *in Cosmos is available at* `code/chap5/cosmos/ER/dist_F.lha`.

6.3.2 Distance automaton $\mathcal{A}_G$

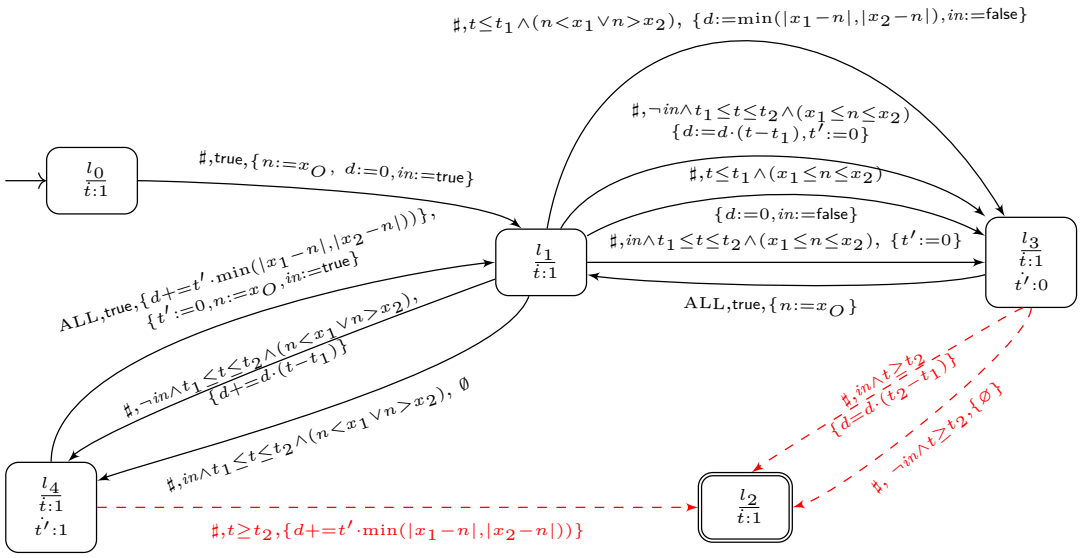


FIGURE 6.4: Automaton for global property $\mathcal{A}_G$.

Automaton $\mathcal{A}_G$ (Figure 6.4) is designed to measure the satisfiability distance (Equation 6.2) of trajectories of a CTMC model $\mathcal{M}$ from the time-bounded region associated with $\mathbf{G}^{[t_1,t_2]}(x_1 \leq x_O \leq x_2)$, i.e. the time-bounded satisfiability region $\mathbf{S}^{[t_1,t_2]}_{(x_1\leq x_O\leq x_2)}$.

It uses the same variables as $\mathcal{A}_F$ plus an extra timer t', to measure the duration of a segment falling outside the satisfiability region within $[t_1, t_2]$, and a boolean flag *in*, which is set to true if the last state of the path is originating in $[t_1, t_2]$ and is outside of the region $[x_1, x_2]$. *in* is used to distinguish cases where the path is out of the region $[x_1, x_2]$ with $t' < t_1$, and a new event occurs at time $t'' > t_1$, to add $(t'' - t_1) * min(|n - x_1|, |n - x_2|)$ instead of $(t'' - t') * min(|n - x_1|, |n - x_2|)$.

After the initialisation of the variables ($l_0 \to l_1$), analysis begins in l_1. For jumps occurring before t_1, we distinguish two cases. If $\sigma@t \in [x_1, x_2]$, the distance is set to zero ($l_1 \to l_3$ top arc). Otherwise, d is the distance of $\sigma@t$ from $[x_1, x_2]$ ($l_1 \to l_3$ midway arc). Indeed, if, for example, the next jump of σ happens at $t > t_2$, then the final distance is given by $d \cdot (t_2 - t_1)$ ($l_1 \to l_2$ bottom arc).

For jumps occurring within $t \in [t_1, t_2]$, if $\sigma@t \notin [x_1, x_2]$ (sequence $l_1 \to l_4 \to l_1$), the distance is incremented by the surface contoured with the trajectory segment (of duration t') laying outside $[x_1, x_2]$, and the closest border of $[x_1, x_2]$. The distance is left unchanged if $\sigma@t \in [x_1, x_2]$ (sequence $l_1 \to l_3 \to l_1$).

Script 8 ($\mathcal{A}_G$ LHA in Cosmos)
An example of automaton $\mathcal{A}_G$ in Cosmos is available at `code/chap5/cosmos/ER/`

dist_G.lha.

6.3.3 Distance automaton $\mathcal{A}_{G\wedge F}$

Automaton $\mathcal{A}_{G\wedge F}$ (Figure 6.5) is designed to compute the distance of trajectories from a sequence of time-bounded regions consisting of a formula $\mathbf{G}^{[t_1,t_2]}(x_1 \leq x_O \leq x_2)$ (related to an observed quantity O) followed by a formula $\mathbf{F}^{[t_3,t_4]}(x_3 \leq x_{O'} \leq x_4)$ (related to an observed quantity O'). This automaton is associated with the MITL formula $\varphi = \mathbf{G}^{[t_1,t_2]}(x_1 \leq x_O \leq x_2) \wedge \mathbf{F}^{[t_3,t_4]}(x_3 \leq x_{O'} \leq x_4)$. We assume $t_2 \leq t_3$, i.e. the global region precedes the eventual region, while $x_1, x_2, x_3, x_4 \in \mathbb{N}$, and x_O, resp. $x_{O'}$, denotes the population of species O, resp. O'.

In particular, it deals with the satisfiabily distance of *conditional time-bounded reachability* because $\mathbf{G}^{[0,s]}(x_1 \leq x_O \leq x_2) \wedge \mathbf{F}^{[s,t]}(x_3 \leq x_{O'} \leq x_4) = (x_1 \leq x_O \leq x_2)\mathbf{U}^{[s,t]}(x_3 \leq x_{O'} \leq x_4)$ when $\mu = (x_1 \leq x_O \leq x_2)\underline{\vee}(x_3 \leq x_{O'} \leq x_4)$ is verified for all states of any possible trajectory of the CTMC (see Remark 6.2.1).

We combine the automata $\mathcal{A}_G$ and $\mathcal{A}_F$ by linking them with an autonomous transition $l_{2G} \rightarrow l_{1F}$. d stores the distance for the **G** region, while d' stores the distance for the **F** region. The sum of the two distances is stored in d at the end of the simulation.

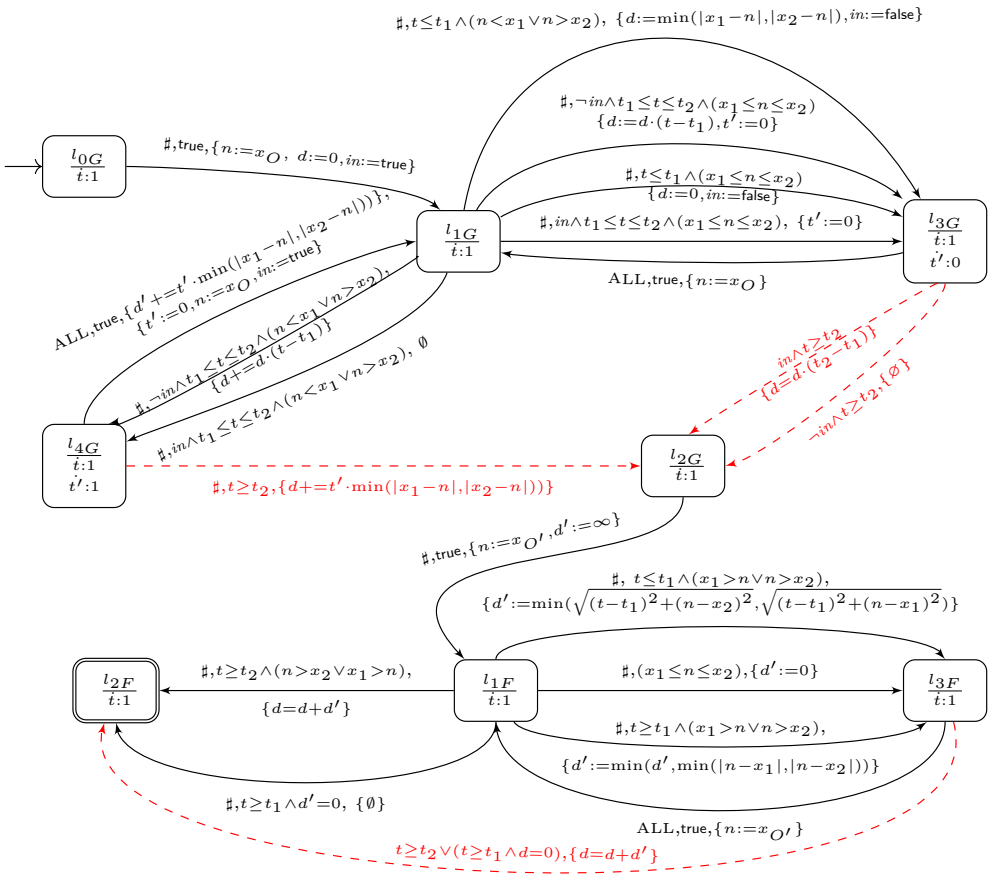


FIGURE 6.5: Automaton for global and eventual property $\mathcal{A}_{G\wedge F}$ when $t_2 \leq t_3$.

Script 9 ($\mathcal{A}_{G\wedge F}$ LHA in Cosmos)

An example of automaton $\mathcal{A}_{G\wedge F}$ in Cosmos is available at `code/chap5/cosmos/ER/dist_G_F.lha`.

6.4 Automaton-ABC algorithm with LHA satisfiability distances

In this part, we define the automaton-ABC framework with satisfiability distances to estimate the satisfaction probability function (Section 6.1). The framework aims to first find the subset of parameters of a parametric CTMC model $(\mathcal{M}_\theta)_{\theta\in\Theta}$ for which a reachability formula φ can be satisfied by efficiently exploring the whole parameter space, and then estimate the satisfaction function probability. The intuition behind this method is that the exploration of the parameter space can be driven efficiently by

taking into account the notion of distance of a trajectory $\sigma \in Path(\mathcal{M}_\theta)$ from the time-bounded satisfiability region corresponding to φ (see Section 6.2). In this formulation of the ABC algorithm, the estimation of the posterior distribution ($\pi_{\varphi-ABC}$) is no longer computed as a limit approximation (i.e. $\lim_{\epsilon \to 0} \widehat{\pi^{\epsilon}_{ABC}}(.|y_{exp})$), as with classical ABC, but rather as an estimation of the exact ABC distribution, since trajectories are accepted exclusively if their distance to the satisfiability region is zero.

6.4.1 Simple ABC with satisfiability distance.

We consider a parametric CTMC $(\mathcal{M}_\theta)_{\theta \in \Theta}$, and we define a prior distribution π over Θ. In Algorithm 17, we propose a modified version of the general automaton-ABC Algorithm 15 adapted to satisfiability distances. At each iteration, we draw a parameter θ' from the prior $\pi(.)$, we simulate a path σ' from the CTMC $\mathcal{M}_{\theta'}$ synchronised with an LHA $\mathcal{A}_\varphi$ (Section 4.4.2) and accept θ' if the distance from φ is $d(\sigma', \varphi) = 0$. (i.e. if $\sigma' \models \varphi$ by Proposition 6.2.1).

Algorithm 17 Automaton-ABC with $\mathcal{A}_\varphi$ automaton

Require: $(\mathcal{M}_\theta)_{\theta \in \Theta}$ a pCTMC, $\pi(.)$ prior,
$\mathcal{A}_\varphi$ automaton distance for MITL formula φ, N: number of particles
Ensure: $(\theta^{(i)})_{1 \leq i \leq N}$ drawn from $\pi_{\varphi-ABC}$
for $i = 1 : N$ **do**
  repeat
    $\theta' \sim \pi(.)$
    $d(\sigma', \varphi) \sim (Last(d), \mathcal{A}_\varphi) \times \mathcal{M}_{\theta'}$
  until $d(\sigma', \varphi) = 0$
  $\theta^{(i)} \leftarrow \theta'$
end for

The Theorem 6.4.1 below links the Algorithm 17 with the satisfaction probability function.

Theorem 6.4.1

Let $(\mathcal{M}_\theta)_{\theta \in \Theta}$ be a parametric CTMC, φ an MITL formula, and the associated satisfaction probability function $\theta \to Pr(\varphi; \mathcal{M}_\theta)$. Given a prior distribution π over the parameter set Θ, the $(\theta^{(i)})_{1 \leq i \leq N}$ sampled by the Algorithm 17 are drawn from a density $\pi_{\varphi-ABC}$:

$$\pi_{\varphi-ABC}(\theta) = Pr(\varphi; \mathcal{M}_\theta) \frac{\pi(\theta)}{C}$$

where $C \in \mathbb{R}_{\geq 0}$ is a positive constant.

Proof. Let $C^{\varphi}_{\mathcal{M}_\theta} = \{\sigma \in Path(\mathcal{M}_\theta)/\sigma \models \varphi\}$. Then $Pr_{\mathcal{M}_\theta}(C^{\varphi}_{\mathcal{M}_\theta}) = Pr(\varphi; \mathcal{M}_\theta)$.

ABC is a reformulation of the accept-reject algorithm (see Section 3.2.2 and Algorithm 11). Thus, the samples $(\theta^{(i)}, \sigma^{(i)})_{1 \leq i \leq N}$ from Algorithm 17 are drawn from a density $\pi_{\varphi-ABC}$:

$$\pi_{\varphi-ABC}(\theta^{(i)}, \sigma^{(i)}) \propto \underbrace{\mathbb{1}_{d(.,\varphi)=0}(\sigma^{(i)})}_{\sigma^{(i)} \models \varphi} \underbrace{p_{\mathcal{M}_{\theta^{(i)}}}(\sigma^{(i)})}_{\sigma^{(i)} \sim \mathcal{M}_{\theta^{(i)}}} \underbrace{\pi(\theta^{(i)})}_{\text{prior}}$$

where $p_{\mathcal{M}_{\theta^{(i)}}}$ is the density related to the CTMC $\mathcal{M}_{\theta^{(i)}}$ with respect to a measure $\overline{\mu}$.

As $\sigma^{(i)} \in C^{\varphi}_{\mathcal{M}_{\theta^{(i)}}} \Leftrightarrow d(\sigma^{(i)}, \varphi) = 0$, $\mathbb{1}_{d(.,\varphi)=0}(\sigma^{(i)}) = \mathbb{1}_{C^{\varphi}_{\mathcal{M}_{\theta^{(i)}}}}(\sigma^{(i)})$. One can obtain the marginal distribution of θ by integration over the whole set of trajectories $Path(\mathcal{M}_\theta)$:

$$\begin{aligned} \pi_{\varphi-ABC}(\theta) &\propto \int_{\sigma \in Path(\mathcal{M}_\theta)} \mathbb{1}_{C^{\varphi}_{\mathcal{M}_\theta}}(\sigma) p_{\mathcal{M}_\theta}(\sigma) \pi(\theta) d\overline{\mu} \\ &\propto \pi(\theta) \int_{\sigma \in C^{\varphi}_{\mathcal{M}_\theta}} p_{\mathcal{M}_\theta}(\sigma) d\overline{\mu} \\ &\propto Pr_{\mathcal{M}_\theta}(C^{\varphi}_{\mathcal{M}_\theta}) \pi(\theta) \end{aligned} \quad \Big) \text{ density of } \mathcal{M}_\theta$$

As $Pr_{\mathcal{M}_\theta}(C^{\varphi}_{\mathcal{M}_\theta}) = Pr(\varphi; \mathcal{M}_\theta)$, we can conlude that there exists $C \in \mathbb{R}_{\geq 0}$ so that $\pi_{\varphi-ABC}(\theta) = Pr(\varphi; \mathcal{M}_\theta) \frac{\pi(\theta)}{C}$. □

This result transforms the regression of the smooth function f_φ into the density estimation of $\pi_{\varphi-ABC}$. First, each parameter sampled from the $\pi_{\varphi-ABC}$ gives information because it produces a simulation that verifies φ. Also, as $\pi_{\varphi-ABC}$ is a probability density function, the relative position of each parameter to the other sampled parameters gives much information about the satisfaction probability function. The denser in sampled parameters a subset of parameters space, the higher the satisfaction probability function over the subset.

Following the same approach, we formulate a comparable version for the ABC-SMC Algorithm 18, which leads to a smaller *runtime* than Algorithm 17, keeping the guarantee of Theorem 6.4.1. Indeed, Algorithm 17 do not really exploit the continuity of the distance. We only accept/reject trajectories depending on whether their distance is zero, i.e. if they satisfy φ. On the contrary, with Algorithm 18, we use the satisfiability distance to rank paths and accept the parameters whose corresponding paths are closer to the satisfiability regions (better ranked) than others, even if they

do not necessarily satisfy φ. This new feature can lead to faster convergence of the algorithm corresponding to a faster exploration of the parameter space.

Algorithm 18 has the same inputs as Algorithm 17, plus a kernel distribution K and a hyperparameter $\alpha \in]0,1[$ representing how fast the tolerance ϵ decreases with the iterations.

It works as follows. Initially, N parameters/particles $(\theta_0^{(i)})_{1\leq j\leq N}$ are drawn from the prior π, and the first tolerance level ϵ to reach equals the α-quantile of the distances d_i, resulting from the synchronised simulations $\mathcal{M}_{\theta_0^{(i)}} \times \mathcal{A}_\varphi$. Then, at each iteration m, a parameter is drawn among the parameters $\theta_m^{(i)}$ $(i \in \{1,\ldots,N\})$, and is moved by a kernel distribution K, and the synchronised simulation $\mathcal{M}_{\theta_m^{(i)}} \times \mathcal{A}_\varphi$ is performed. This procedure is done until the resulting distance d_i is below the current tolerance level ϵ, inducing that the parameter $\theta_m^{(i)}$ is kept. After doing so for the N parameters, we compute a new tolerance level ϵ that equals the α-quantile of the distances d_i. These iterations are repeated until the last tolerance level $\epsilon = 0$ is reached. The introduction of several steps with positive decreasing tolerances leads to an efficient exploration of the parameter space driven by $\mathcal{A}_\varphi$.

Another interesting point is that any algorithm inherited from the family of ABC methods keeps the guarantee of Theorem 6.4.1. Thus depending on the problem, one can use any Monte-Carlo based sampler (ABC-MCMC, ABC-SMC and so on) and still estimate the satisfaction probability function.

6.4.2 Estimation of the satisfaction probability function

Based on the samples $(\theta^{(i)})_{1\leq i\leq N}$ (Algorithm 17 or Algorithm 18), we can estimate the satisfaction probability function thanks to Theorem 6.4.1. This procedure is twofold: estimation of the ABC posterior density and estimation of the constant C.

Estimation of the $\pi_{\varphi-ABC}$ posterior distribution

In the applications, we estimate the ABC posterior $\pi_{\varphi-ABC}$ based on the samples $(\theta^{(i)})_{1\leq i\leq N}$ with kernel density estimation (Section 3.4): $\forall\theta \in \mathbb{R}, \hat{\pi}_h(\theta) = \frac{1}{N}\sum_{i=1}^{d} K_h(\theta,\theta^{(i)})$. Two kernels are used: Gaussian and Beta kernels. The last one is helpful when we have to estimate densities over bounded supports with positive probabilities on the boundaries. The optimal bandwidth is obtained by Least Squares Cross-Validation.

Algorithm 18 Automaton-ABC Sequential Monte Carlo with $\mathcal{A}_\varphi$ automaton

Require: $(\mathcal{M}_\theta)_{\theta\in\Theta}$ a pCTMC, π prior,
$\mathcal{A}_\varphi$ distance automaton for MITL formula φ,
N : number of particles, $\alpha \in (0,1)$, K kernel distribution
Ensure: $(\omega^{(i)}, \theta^{(i)})_{1\leq i\leq N}$ weighted samples drawn from $\pi_{\varphi-ABC}$
$\theta_0^{(i)} \sim \pi, i \in 1,\dots,N$
$d_i \sim (Last(d), \mathcal{A}_\varphi) \times \mathcal{M}_{\theta_0^{(i)}}, i \in 1,\dots,N$
$\epsilon \leftarrow quantile(\alpha, (d_i)_{1\leq i\leq N})$
$(\omega_0^{(i)})_{1\leq i\leq N} \leftarrow \frac{1}{N}$
$m \leftarrow 1$
while $\epsilon > 0$ **do**
  for $i = 1 : N$ **do**
    repeat
      Take θ' from $(\theta_{m-1}^{(j)})_{1\leq j\leq N}$ with probabilities $(\omega_{m-1}^{(j)})_{1\leq j\leq N}$
      $\theta_m^{(i)} \sim K(.|\theta')$
      $d_i \sim (Last(d), \mathcal{A}_\varphi) \times \mathcal{M}_{\theta_m^{(i)}}$
    until $d_i \leq \epsilon$
    $\omega_m^{(i)} \leftarrow \frac{\pi\left(\theta_m^{(i)}\right)}{\sum_{i'=1}^{N} \omega_{m-1}^{(i')} K(\theta_m^{(i)}|\theta_{m-1}^{(i')})}$
  end for
  Normalise $(\omega_m^{(i)})_{1\leq i\leq N}$
  $\epsilon \leftarrow quantile(\alpha, (d_i)_{1\leq i\leq N})$
  $m \leftarrow m + 1$
end while
return $(\omega_m^{(i)}, \theta_m^{(i)})_{1\leq i\leq N}$

Estimation of the constant C

An estimation of C can be obtained by a single-point estimation of $Pr(\varphi; \mathcal{M}_{\theta^*})$ and $\pi_{\varphi-ABC}(\theta^*)$. θ^* should be chosen wisely: verifying φ should not be rare and θ^* should be in a region where $\pi_{\varphi-ABC}$ can be well approximated (a region of high posterior probability). Then, $Pr(\varphi; \mathcal{M}_{\theta^*})$ can be estimated with statistical model checkers. In order to have a more stable kernel density estimation, one can also choose several θ^* and estimate the constants and compute the mean.

6.5 Applications

We applied the automaton-ABC method with satisfiability distances to estimate the satisfaction probability function of Poisson processes and three models of biological systems: an enzymatic reaction network (Michaelis-Menten kinetics), a model of viral infection and SIR chemical reaction network. Our results are compared with model checking or statistical model checking from Prism or Cosmos. We also provide an estimation of the experiments' computational times.

6.5.1 An example with Poisson processes

Let $\mathcal{M}_\lambda$ be the CTMC defined on a space set $\mathbf{S} = \mathbb{N}$ by the initial distribution and the transition rate matrix Q:

$$\begin{aligned} \alpha &= \delta_0 \text{ (the initial state is 0)} \\ Q(i, i+1) &= \lambda \text{ for } i \in \mathbb{N} \\ Q(i, i) &= -\lambda \text{ for } i \in \mathbb{N} \\ Q(i, j) &= 0 \text{ otherwise} \end{aligned}$$

We have defined a Poisson process of parameter λ, described by the Chemical Reaction Network

$$X \xrightarrow{\lambda} 2X$$

where the kinetic rate is constant and equals λ (it does not follow mass-action law).

The associated stochastic process is denoted $S^\lambda = (S_t^\lambda)_{t \geq 0}$. Let us solve the reachability problem:

$$\text{Estimate the function } \lambda \to Pr_{\mathcal{M}_\lambda}(\sigma \models \mathbf{F}^{[t_1,t_2]}(x_1 \leq \mathbf{s} \leq x_2)).$$

The set $C^{\varphi}_{\mathcal{M}_\lambda}$ is the set of trajectories of $\mathcal{M}_\lambda$ that satisfies $\varphi = \mathbf{F}^{[t_1,t_2]}(x_1 \leq \mathbf{s} \leq x_2)$ for the CTMC $\mathcal{M}_\lambda$. There are powerful known results on Poisson processes such as:

i. The trajectories are increasing step functions with jumps of value one (i.e has the form $1 \xrightarrow{t_1} 2 \xrightarrow{t_2} 3 \to \ldots$).

ii. $\forall t, v \in \mathbb{R}_{\geq 0}$ with $v \leq t, S^\lambda_t - S_{\lambda,v} \sim \mathcal{P}(\lambda(t-v))$, i.e. a Poisson distribution of parameter $\lambda(t-v)$.

With the property i), having one point in the time-bounded satisfiability region of $\mathbf{F}^{[t_1,t_2]}(x_1 \leq \mathbf{s} \leq x_2)$ is equivalent to having $\sigma@t_1 \leq x_2$ and $\sigma@t_2 \geq x_1$. Indeed, if $\sigma@t_1 > x_2$, the trajectory is on the top of the region, and if $\sigma@t_2 < x_1$, the trajectory is below the region. Else, the trajectory must traverse $[x_1, x_2]$.

With this remark, $\mathbb{P}_{S^\lambda}(C^{\varphi}_{\mathcal{M}_\lambda}) = \mathbb{P}(S^\lambda_{t_1} \leq x_2, S^\lambda_{t_2} \geq x_1)$.

$$
\begin{aligned}
\mathbb{P}(S^\lambda_{t_1} \leq x_2, S^\lambda_{t_2} \geq x_1) &= \mathbb{P}(S^\lambda_{t_2} \geq x_1 | S^\lambda_{t_1} \leq x_2)\mathbb{P}(S^\lambda_{t_1} \leq x_2) \\
&= \textstyle\sum_{k=0}^{x_2} \mathbb{P}(S^\lambda_{t_2} \geq x_1 | S^\lambda_{t_1} = k)\mathbb{P}(S^\lambda_{t_1} = k | S^\lambda_{t_1} \leq x_2) \\
&= \textstyle\sum_{k=0}^{x_2} \mathbb{P}(S^\lambda_{t_2} - S^\lambda_{t_1} \geq x_1 - k)\mathbb{P}(S^\lambda_{t_1} = k) \\
&= \textstyle\sum_{k=0}^{x_2} (1 - \mathbb{P}(S^\lambda_{t_2} - S^\lambda_{t_1} < x_1 - k))\mathbb{P}(S^\lambda_{t_1} = k) \\
&= \textstyle\sum_{k=0}^{x_2} (1 - \mathbb{P}(S^\lambda_{t_2} - S^\lambda_{t_1} \leq x_1 - k - 1))\mathbb{P}(S^\lambda_{t_1} = k)
\end{aligned}
$$

$\{S^\lambda_{t_1} = k\}_{k=[[0,x_2]]}$ *is a partition of* $\{S^\lambda_{t_1} \leq x_2\}$

As $S^\lambda_{t_2} - S^\lambda_{t_1} \sim \mathcal{P}(\lambda(t_2 - t_1)))$ and $S^\lambda_{t_1} \sim \mathcal{P}(\lambda t_1)$, $\mathbb{P}_{S^\lambda}(C^{\varphi}_{\mathcal{M}_\lambda})$ can be expressed analytically. As $\mathbb{P}_{S^\lambda} = Pr_{\mathcal{M}_\lambda}$ (Proposition 2.2.1), the satisfaction probability function $\lambda \to Pr_{\mathcal{M}_\lambda}(\sigma \models \mathbf{F}^{[t_1,t_2]}(x_1 \leq \mathbf{s} \leq x_2))$ is expressed analytically. We have solved the time-bounded reachability problem analytically, which offers the possibility to test our method with exact continuous solutions.

The Figure 6.6 below shows the run of the automaton-ABC algorithm with a prior $\pi(.) \sim \mathcal{U}(0, 30)$, $N = 500$ particles with $x_1, x_2 = 5.0, 7.0$ and $t_1, t_2 = 0.75, 1.0$. The estimated satisfaction probability function with Gaussian kernel estimation of the ABC posterior is illustrated in red, whereas the true satisfaction probability function is in blue. One can see that our estimation suits well the exact satisfaction probability function.

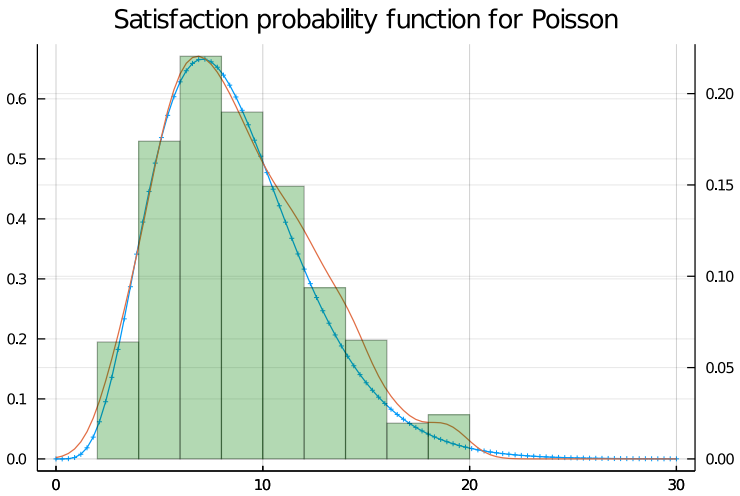


FIGURE 6.6: Histogram of the automaton-ABC posterior with $N = 500$ particles.
In red: estimated satisfaction probability function.
In blue: the analytical satisfaction probability function.

6.5.2 Enzymatic reaction system

Model

We consider the Enzymatic Reaction system (Michaelis-Mentens kinetics (Michaelis et al., 2011)) described in Equations 6.4 in which a *substrate* species S is converted into a *product* P through the mediation of an *enzyme* E. The dynamics depend on the reaction rate constants that define a parameter vector $\theta = \{k_1, k_2, k_3\}$. We thus consider the underlying parametric CTMC $(\mathcal{M})_{\theta \in [0,100]^3}$. The initial state is $(E_0, S_0, ES_0, P_0) = (100, 100, 0, 0)$.

$$\begin{aligned} R_1 &: E + S \xrightarrow{k_1} ES \\ R_2 &: ES \xrightarrow{k_2} E + S \\ R_3 &: ES \xrightarrow{k_3} E + P \end{aligned} \tag{6.4}$$

Figure 6.7 shows two (4-dimensional) trajectories sampled from the CTMC model $\mathcal{M}_\theta$ of the enzymatic reaction system with parameters $\theta = (1, 1, 1)$ (left) and $\theta = (0.1, 1, 0.1)$ (right).

The dynamic of the ER system (Figure 6.7) is such that the totality of the substrate (initially $S_0 = 100$) is converted into the product at a speed dependent on parameters θ. With $\theta = (1, 1, 1)$, the totality of S is converted before $time = 5$, whereas with a tenfold speed reduction in the formation of the ES complex and

synthesis of P (i.e. $\theta = (0.1, 1, 0.1)$), we have that only about 30% of S has been converted at $time = 5$.

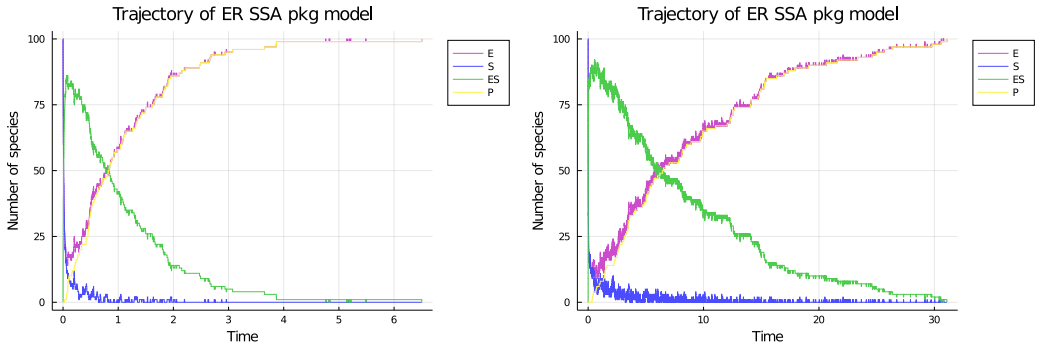


FIGURE 6.7: Trajectories of the ER system with $\theta_{\text{left}} = (1, 1, 1)$ and time scale $[0, 6]$, $\theta_{\text{right}} = (0.1, 1, 0.1)$ and time scale $[0, 30]$.

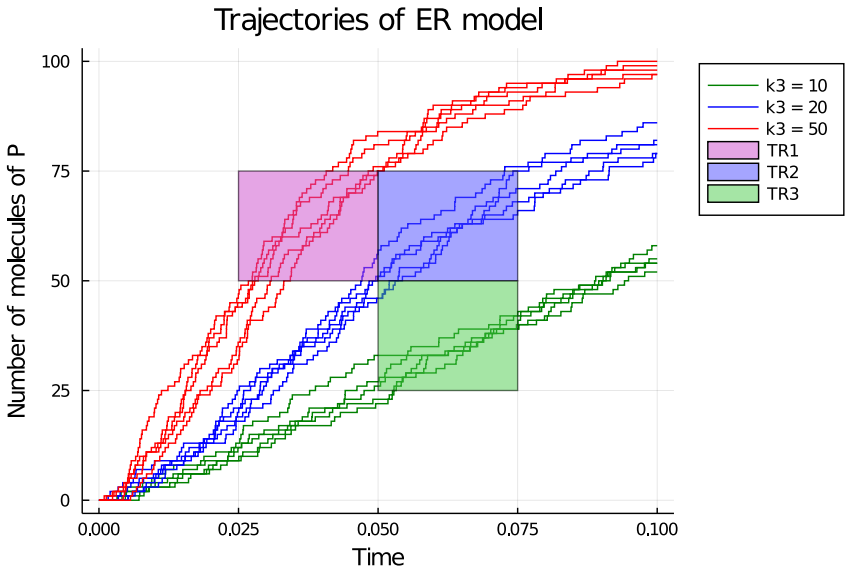


FIGURE 6.8: Trajectories of ER system for product P, with TR1, TR2 and TR3 regions and $k_3 \in \{10, 20, 50\}$.

Experimental setting

In the following, we run a few experiments on the ER system. We consider six time-bounded reachability formulae and their associated time-bounded satisfiability regions:

- $\varphi_1 : \mathbf{F}^{[0.025,0.05]}(50 \leq P \leq 75)$ associated with region $TR1$,
- $\varphi_2 : \mathbf{F}^{[0.05,0.075]}(50 \leq P \leq 75)$ associated with region $TR2$,
- $\varphi_3 : \mathbf{F}^{[0.05,0.075]}(25 \leq P \leq 50)$ associated with region $TR3$,

- $\varphi_4 : \mathbf{F}^{[8.0,10.0]}(5 \leq P \leq 15)$ associated with region $TR4$,
- $\varphi_5 : \mathbf{G}^{[0.0,0.8]}(50 \leq E \leq 100)$ associated with region $TR5$,
- $\varphi_6 : \mathbf{G}^{[0.0,0.8]}(50 \leq E \leq 100) \wedge \mathbf{F}^{[0.8,0.9]}(30 \leq P \leq 100)$ associated with region $TR6$.

$\varphi_1, \varphi_2, \varphi_3$ correspond to formulae that are probably fulfilled when k_3 varies over $[0, 100]$ (with $k_1 = k_2 = 1$), whereas $\varphi_4, \varphi_5, \varphi_6$ are formulae for which only a small subset of $k_1, k_2 \in [0, 100]^2$ (with $k_3 = 1$) can produce trajectories that fulfil these formulae.

Test of LHA distances

Figure 6.8 shows batches of trajectories of the ER model for product P, corresponding to parameters $\theta_1 : (1, 1, 50)$, $\theta_2 = (1, 1, 20)$ and $\theta_3 = (1, 1, 10)$. It appears that trajectories for $\mathcal{M}_{\theta_1}$ (red) are more likely to traverse TR_1, those for $\mathcal{M}_{\theta_2}$ (blue) to traverse $TR2$ and those for $\mathcal{M}_{\theta_3}$ (green) to traverse $TR3$.

Such intuition is confirmed by plots in Figure 6.9, which depicts the average value of the distance of trajectories from time regions $TR1$, $TR2$ and $TR3$ as a function of k_3. We use the Statistical Model Checker Cosmos (Ballarini et al., 2015) with specific instances of $\mathcal{A}_F$ i.e. $\mathcal{A}_{\varphi_1}$, $\mathcal{A}_{\varphi_2}$ and $\mathcal{A}_{\varphi_3}$. We observe that for example the measured distance from region $TR1$ monotonically decreases as k_3 increases and cancels for $k_3 \geq 30$, while the distance from region $TR3$ is zero when $10 \leq k_3 \leq 15$ whereas it grows as we increase k_3.

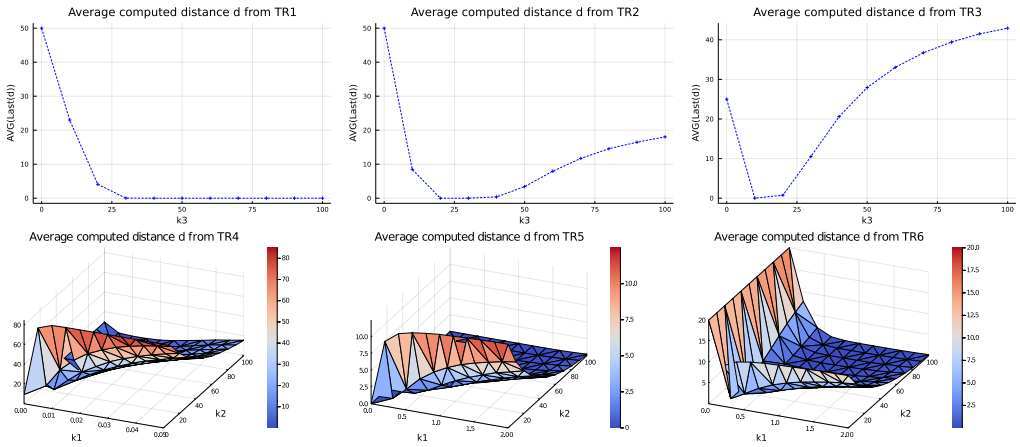


FIGURE 6.9: Average distances of the automata for the six formulae φ_i, $i \in \{1, \ldots, 6\}$, computed by the Statistical Model Checker Cosmos with approximation of 0.1 and 99% level of confidence. First row: $\mathcal{A}_{\varphi_1}, \mathcal{A}_{\varphi_2}$ and $\mathcal{A}_{\varphi_3}$. Second row: $\mathcal{A}_{\varphi_4}, \mathcal{A}_{\varphi_5}$ and $\mathcal{A}_{\varphi_6}$.

Satisfaction probability function estimation

We apply the automaton-ABC Algorithm 18 to the ER parametric CTMC defined over $[0, 100]^3$, with formulae described in Section 6.5.2.

$TR1$, $TR2$ and $TR3$ correspond to the one-dimensional experiments: only k_3 varies over $[0, 100]$ ($k_1 = k_2 = 1.0$ are fixed), a uniform prior $\pi(.) \sim \mathcal{U}(0, 100)$ is set. $TR4$, $TR5$ and $TR6$ correspond to the two-dimensional experiments: k_1 and k_2 varies over $[0, 100]$ ($k_3 = 1.0$ is fixed), a uniform prior $\pi(.) \sim \mathcal{U}(0, 100)$ over each parameter is set.

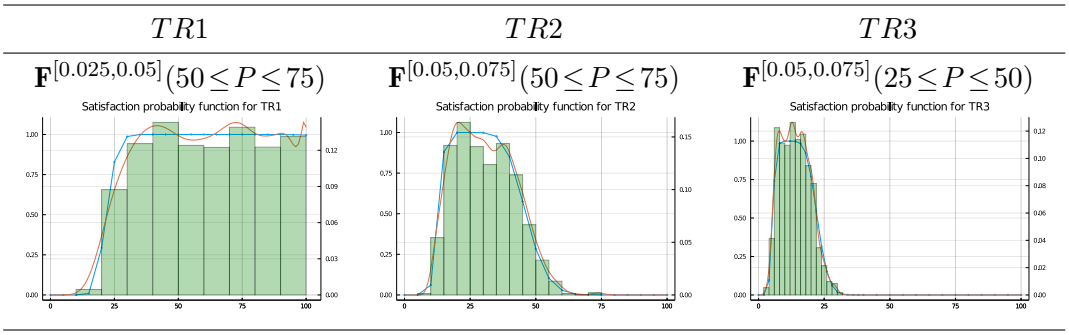


FIGURE 6.10: Weighted histogram of automaton-ABC posteriors with 1000 particles. In each experiment, $k_1 = k_2 = 1$ and $\pi_{k_3}(.) \sim \mathcal{U}(0, 100)$.
In blue: the satisfaction probability function estimated on a selection of points with Prism model checker by numerical method.
In red: the satisfaction function estimated through kernel density estimation method based on automaton-ABC samples.

Figure 6.10 illustrates the evaluation of the posterior distribution $\pi_{\varphi-ABC}$ and the estimation of the satisfaction probability as a function of parameter k_3 (1D experiments) obtained with the automaton-ABC method (Algorithm 18) over the formulae $\varphi_1, \varphi_2, \varphi_3$.

The estimated function of $TR1$ exhibits a uniform profile with a 95% credibility interval that φ_1 is satisfied for $k_3 \in [20, 100]$ (approximately), which is in agreement with the average distance measure (Figure 6.9). When the average distance is zero, the estimated probability by both Prism model checking and automaton-ABC is 1. Instead, the estimated functions for $TR2$ and $TR3$ result in narrower 95% credibility intervals with $k_3 \in [15, 50]$ ($TR2$), and $k_3 \in [5, 25]$ ($TR3$), again in line with measured distance (Figure 6.9).

Figure 6.11 depicts the results of the 2D experiments on the ER system with $\varphi_4, \varphi_5, \varphi_6$ formulae. The triangular profile of the joint posterior in experiments $TR4$ and $TR5$ (computed with $k_3 = 1$ and $\pi_{k_1}(.), \pi_{k_2}(.) \sim \mathcal{U}(0, 100)$) indicates that only very low values of k_1 ($k_1 \leq 0.015$ for $TR4$, $k_1 \leq 1$ for $TR5$) combined with rather high

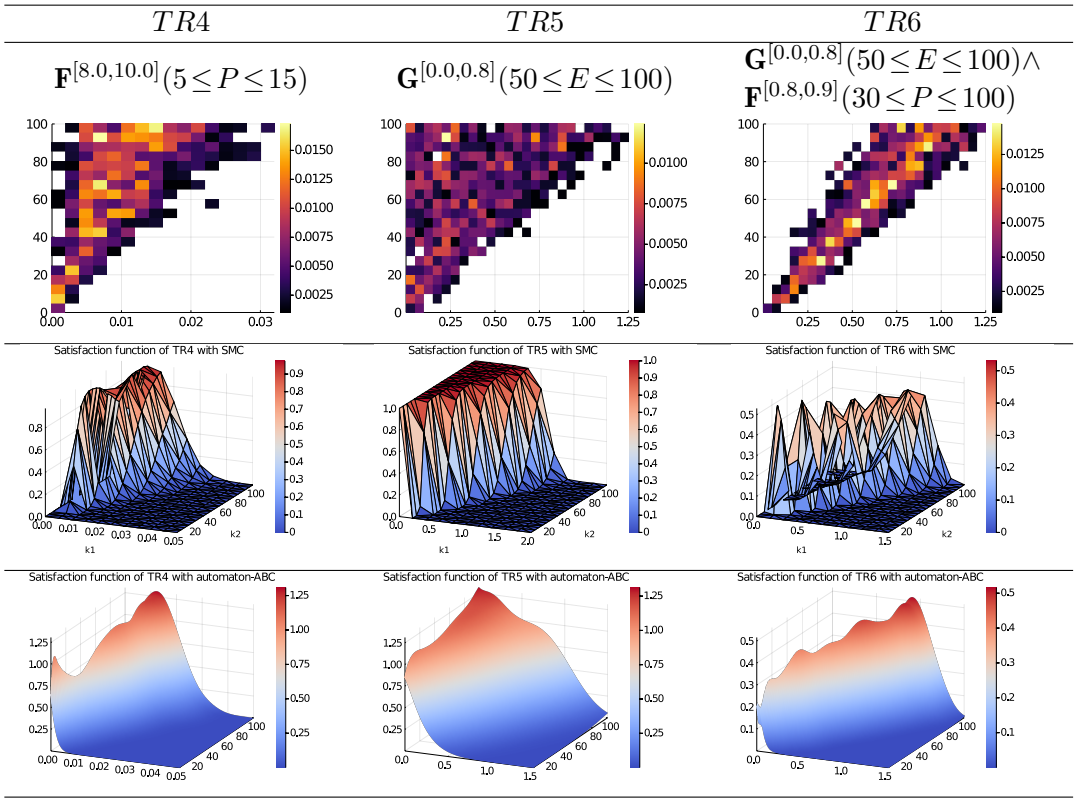


FIGURE 6.11: Results of 2D experiments of ER system with 1000 particles $(\pi_{k_1}(.), \pi_{k_2}(.) \sim \mathcal{U}(0, 100),\ k_3 = 1)$.
Top: the 2D weighted histograms of the automaton-ABC posteriors.
Middle: estimation of the satisfaction probability function with Prism by Statistical model checking (99% confidence interval with approximation 0.01).
Bottom: kernel density estimation of the satisfaction probability function.

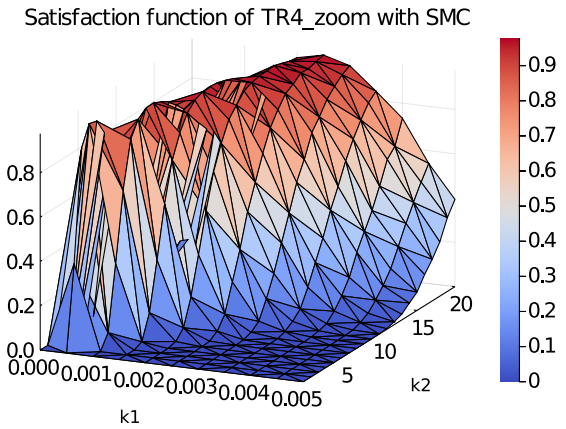


FIGURE 6.12: Statistical Model Checking over $[0.0, 0.0005] \times [0.0, 20.0]$ with 10 points for the first axis and 20 points for the second.

values of k_2 (i.e. $k_2 \in [40, 100]$ for $TR4$, $k_2 \in [50, 100]$ for $TR5$) result in trajectories entering $TR4$, or never leaving $TR5$, which means that the algorithm managed to catch the correlation between the parameters. This is intuitively correct in both cases. In fact, $TR4$ corresponds to a very low synthesis of P, which is not compatible with fast creation of the ES complex (i.e. only very small k_1 are not ruled out), and even the compensation effect obtained by fast decomplexation (i.e. large k_2) will not suffice for the different trajectories to stay in $TR4$.

One can notice that the estimated satisfaction probability function of region $TR4$ (most-left bottom picture of Figure 6.11) has much probability mass around $(0, 0)$. At first glance, one can conclude in a problem of bias of the kernel density estimator since the satisfaction probability function estimated by Statistical Model Checking (second row, first column picture) does not show the same shape in this area. But, if we execute Statistical MC with a refined grid around zero (Figure 6.12), one can see surprisingly that probability values are high around $(0, 0)$. This behaviour was not expected and automaton-ABC algorithm permits to discover this small area of high satisfaction probability that was not caught by Statistical Model Checking over the grid of 20 points per axis.

Similarly to experiment $TR4$, $TR5$ limits the speed of the initial decrease of E (which is initially $E_0 = 100$), to 50 within $t \leq 0.8$. Again, this behaviour is only compatible with a slow ES complexation and cannot be compensated by fast decomplexation. The $TR6$ experiment caught an even more important correlation between parameters. In fact, the support of $TR6$ posterior is contained within the support of $TR5$ posterior. Indeed, a trajectory that verifies φ_6 also verifies φ_5.

Remarks

Results have been obtained by running Algorithm 18 with a sample size $N = 1000$. There are no notable differences in performance between Algorithm 17 and 18 for 1D experiments on regions $TR1$, $TR2$, $TR3$ because of the large size of the resulting distributions. However, simple automaton-ABC Algorithm 17 is not worth considering for $TR4$ and $TR5$. Given the large size of the support for the considered priors ($[0, 100] \times [0, 100]$) we remark that the probability of sampling (a pair of) parameters in the resulting posterior-distribution is about $\frac{\frac{90 \times 0.03}{2}}{100 * 100} \approx 10^{-4}$. This leads to a tiny probability of drawing from the prior $N = 1000$ particles that fall in such a narrow distribution, let alone that even a parameter sampled in the obtained distribution could produce paths that do not satisfy φ. By adding several transitional steps with the SMC version (Algorithm 18), the problem becomes treatable. The results for $TR4$ required about 2×10^5 simulations of the model, which is the highest number of simulations in all experiments.

6.5.3 SIR

We consider the SIR compartmental model (Kermack and McKendrick, 1927) already described in Equations 2.15. We tested our algorithm on a single formula $\varphi = \mathbf{G}^{[0,100]}(I > 0) \wedge \mathbf{F}^{[100,120]}(I = 0)$ with one 1D experiment and one 2D experiment. This formula means that the considered epidemic is active within the time interval $[0, 100]$ but disappears during the time interval $[100, 120]$. In the 1D experiment, $k_i = 0.0012$ is fixed, whereas $k_r \sim \mathcal{U}(0.005, 0.2)$. In the 2D experiment, both parameters vary: $k_i \sim \mathcal{U}(5 \cdot 10^{-4}, 0.003)$ and $k_r \sim \mathcal{U}(0.005, 0.2)$. Figure 6.13 reports the results of both experiments, including comparing of the satisfaction probability function obtained with the automaton-ABC method with that obtained with the Prism model checker (numerical method for both 1D and 2D experiments). One can see that the satisfaction probability function is well reconstructed in both cases.

Whereas success is quite expected in the 1D experiment because the histogram suggests a Gaussian-like shape of the density, the result for the 2D experiment is more remarkable because it is hard to guess the density shape based on the 2D histogram. However, our algorithm managed to reproduce the same complex shape and values given by the Model Checking estimates.

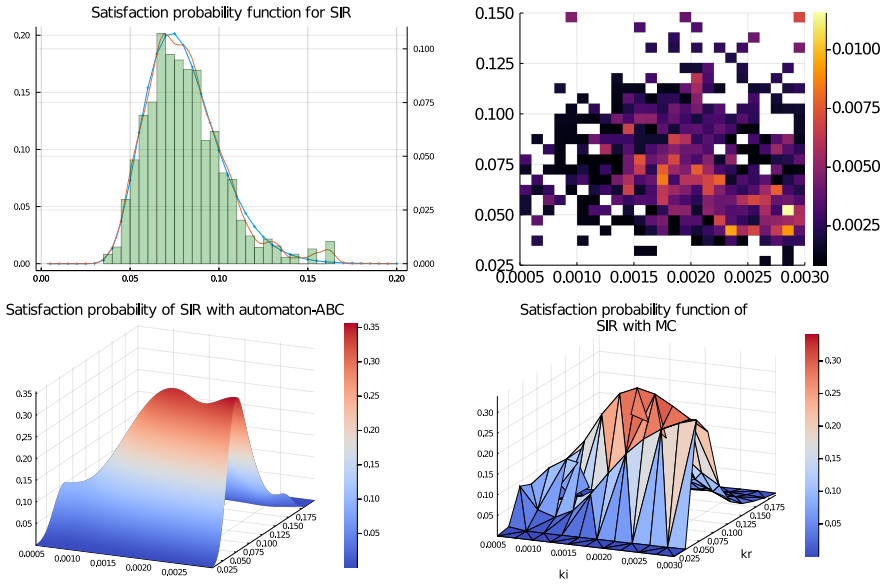


FIGURE 6.13: Results for the SIR model with $\varphi = \mathbf{G}^{[0,100]}(I > 0) \wedge \mathbf{F}^{[100,120]}(I = 0)$.

On the top left figure: weighted histogram of automaton-ABC posterior with 1000 particles for the 1D experiment; in blue: the true satisfaction probability function computed with the Prism model checker using the numerical engine of Prism over 40 points; in red: the estimated satisfaction function with the kernel density estimation method. The three other figures correspond to the 2D experiment. On the top right figure: the 2D histogram of the automaton-ABC posterior. On the bottom left figure: the kernel density estimation of the satisfaction probability function. On the bottom right figure: the estimation of the satisfaction probability function by the Prism model checker (numerical method).

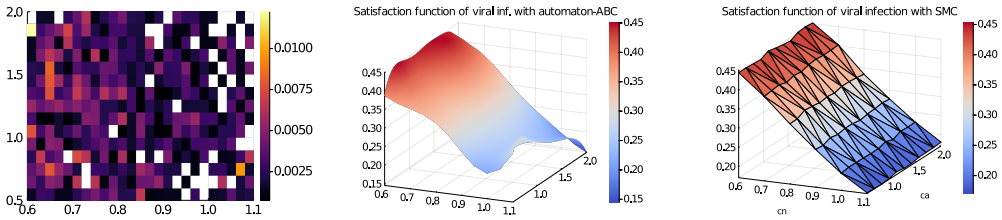


FIGURE 6.14: Results for the intracellular viral infection model with $\varphi = \mathbf{G}^{[0,50]} G \leq 10 \wedge \mathbf{F}^{[50,200]} G > 100$.
Left figure: the 2D histogram of the automaton-ABC posterior.
Middle figure: kernel density estimation of the satisfaction probability function.
Right figure: estimation of the satisfaction probability function by Monte-Carlo simulations (based on a 99% confidence level and approximation 0.01).

6.5.4 Intracellular viral infection

We consider a model of cell viral infection (Haseltine and Rawlings, 2002) described by Equations 6.5.4.

$$\begin{array}{ll} N + T \xrightarrow{k_1 X_T c_n} G + T & T \xrightarrow{k_4 X_T} \varnothing \\ N + G \xrightarrow{k_2 X_G c_n} T & S \xrightarrow{k_5 X_S} \varnothing \\ N + A + T \xrightarrow{k_3 X_T c_n c_a} S + T & G + S \xrightarrow{k_6 X_G X_S} V \end{array}$$

N represents the nucleotides and A the amino acids. G is the genomic nucleic acids, T the template nucleic acids, S the viral structural protein and V the secreted virus. In this model, we assume nucleotides and amino acids have constant concentrations c_n and c_a: we suppose these species are in large number.

Satisfaction probability function estimation A 2D experiment is considered with the following logical property: $\varphi = \mathbf{G}^{[0,50]}(G \leq 10) \wedge \mathbf{F}^{[50,200]}(G > 100)$. This property expresses that the species G remains stable and low within time $[0, 50]$, and then a burst of speed in the creation of G occurs within $[50, 100]$, which gives the necessary material for the creation of the virus V.

We vary the concentrations of nucleotides and amino acids: $c_n \sim \mathcal{U}(0.6, 1.1)$ and $c_a \sim \mathcal{U}(0.5, 2.0)$. Figure 6.14 reports the results of the experiment. Considering the end time of the formula, each simulation of this model is computationally expensive. But our methodology still allows a proper run of the experiment, and we get a reasonable approximation of the satisfaction function (it required about $7 \cdot 10^3$ simulations).

6.5.5 About the implementation of the automaton-ABC method

The implementation of automaton-ABC methods led to the development of a Julia package. It includes both simulation and synchronised simulation of CTMC, easy modeling of CTMCs based on Chemical Reaction Network, and ABC related methods. Many details are provided in Appendix A.

ABC related algorithms are distributed and the above applications were performed using HPC resources from the "Mésocentre" computing center of CentraleSupélec and École Normale Supérieure Paris-Saclay supported by CNRS and Région Île-de-France (`http://mesocentre.centralesupelec.fr/`).

Tables 6.1 and 6.2 show the performance results for the whole set of experiments. We omit the Kernel Density Estimation. It only depends on the number of particles N, the length of the grid, and the choice of the kernel we set. Still, the computational time of Least Squares Cross-Validation can be prohibitive when the kernel is a multivariate beta kernel with a high number of particles ($N \geq 1000$) and grid search (1000 is already high even if the computation of LSCV is distributed).

When the number of jobs is 120, the run was distributed on the Mesocentre HPC cluster. Otherwise, it was run on a Dell XPS 9370 with CPU Intel i7 8550U @ 1.8GHz x 8 cores.

Our tool for automaton-ABC in Julia is very efficient because, in our applications, any experiment run is rarely higher than one minute. For example, the experiment about the region TR4 of the ER model performs 256000 simulations within a minute over the cluster (without counting the other computations of the ABC algorithm), knowing that each simulation that reaches $TR4$ is about 2000 steps of Stochastic Simulation Algorithm 1. In Appendix A, we compare the performance of our tool with state-of-the-art softwares.

The SIR 2D experiment has a higher computational time than the 1D experiment, even if it is run with 120 jobs. This is explained by the fact that creating and dealing with many jobs has a higher computational cost when the execution time per job is low. It is the case here: less than 7 seconds of computation has to be distributed over 120 jobs, which is not helpful. The viral infection model has a higher computational cost per simulation because it is a much more complex model than the two others.

This computational time has to be put in perspective to classical Statistical Model Checking methods. For example, in experiment $TR4$, Statistical Model Checking has been performed over a grid of 200 points of a much smaller set than $[0, 100] \times$

Exp	ER TR1	ER TR2	ER TR3	SIR 1D
Number of jobs	1	1	1	1
Number of sim.	2263	4896	7197	25404
Time (sec)	7.5	10.9	8.7	7.1

TABLE 6.1: Performance results for the one-dimensional experiments of automaton-ABC.

Exp	ER TR4	ER TR5	ER TR6	SIR 2D	Viral infection
Number of jobs	120	120	120	120	120
Number of sim.	256641	32367	47649	17284	6125
Time (sec)	66.18	29.2	31.9	13.1	70.1

TABLE 6.2: Performance results for the two-dimensional experiments of automaton-ABC.

$[0, 100]$. It lasted more than one hour, and we saw this grid missed a region of high probability for φ_4 formula.

Script 10 (Automaton-ABC experiments related scripts)
Several scripts can be found in the git repository of the thesis:

- `code/chap5/prism` *are related to model checking and statistical model checking estimations of the satisfaction probability function.*
- `code/chap5/cosmos` *are related to the computation of average distances with Cosmos.*
- `code/chap5/julia` *are related to the run of automaton-ABC algorithm, which are based on our package (Appendix A), as well as illustrations of the Applications Section 6.5.*

6.6 A comparison with Smoothed Model Checking

In this chapter, we have presented a method that tackles the *estimation problem* (Definition) in stochastic model checking. As seen in Chapter 3, it is in line with the work of (Bortolussi, Milios, and Sanguinetti, 2016) where the so-called Smoothed Model Checking (Smoothed MC) method estimates the satisfaction probability function of parametric CTMC. Methods belonging to this family do not solve the parameter synthesis problem explicitely but provide reasonable estimates of the satisfaction function that can lead to parameter synthesis, e.g. (Bortolussi and Silvetti, 2018) which is also based on Smoothed MC. In this section, we discuss some results of Smoothed MC algorithm for which a Python version is given in (Bortolussi and Silvetti, 2018). We have already described this method in Section 4.3.1. For the sake of comparison, we

reproduced the 1D experiments for the ER model (Section 6.5.2) using the Smoothed MC tool.

Figure 6.15 depicts plots of the satisfaction probabilities obtained with Smoothed MC, which show a good agreement with those obtained with the automaton-ABC approach and Prism model checker (Figure 6.10). Table 6.3 reports the number of trajectories generated by both algorithms, showing a clear advantage for the automaton-ABC approach. In this table, we omit simulations involved in estimating $Pr(\varphi; \mathcal{M}_{\theta*})$ for the automaton-ABC. On the contrary, this table does not show the cost of normalisation constant estimation in the Smoothed MC posterior (Equation 4.1) by Expectation-Propagation. Also, a default value of 600 trajectories is set for each satisfiability probability estimation by Statistical Model Checking, which should be higher if one wants a high confidence level and a precise approximation.

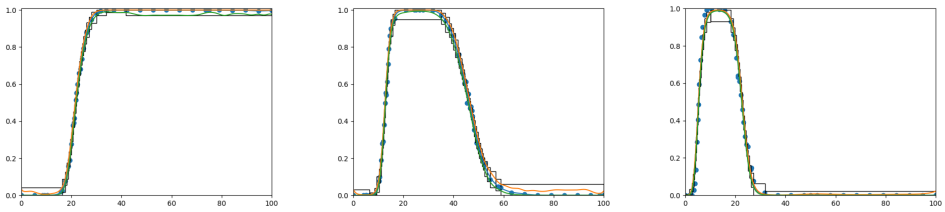


FIGURE 6.15: Estimation of the satisfaction probability functions in experiments TR1, TR2, TR3 for the ER system reported in Figure 6.10 with Smoothed MC algorithm. In blue: the estimated function; in green: the lower bound; in orange: the upper bound.

	TR1	TR2	TR3
Smoothed MC	27000	37200	32400
Automaton-ABC	2263	4896	7197

TABLE 6.3: Number of simulations before termination for both algorithms. For automaton-ABC: number of $N = 1000$ particles. For smoothed MC: each point of the dataset is estimated with 600 trajectories (default value).

In return, the ABC method does not have the same statistical guarantees as Smoothed MC. Indeed, we do not assume any specific form for the density $\pi_{\varphi-ABC}$: we minimise Least Squares Cross-Validation criterion to have the better trade-off between bias and variance over the $N = 1000$ particles in kernel density estimation. To run the 2D experiments, we have adapted the available Smoothed MC code, but preliminary tests seem to indicate a prohibitive computational time. Indeed, convergence for 2D experiments such as $TR4$ required many more trajectories simulation ($\sim 2 \cdot 10^5$). The end time of simulations is much higher than the 1D experiments ($t_2 = 10.0 \gg 0.075$).

The 2D experiments on the ER system show that our method gives an efficient way to identify the region of the parameter space where the satisfaction function is positive. After such an exploration is performed, one can use either kernel density estimation procedure, or use regression function methods such as Smoothed MC, that trade higher computational cost for better statistical guarantees.

Note that since the main computational cost of the ABC approach is the simulation of trajectories, our method is well suited for distributed computing, which is not the case for all regression methods (e.g. Smoothed MC is, by nature, sequential).

6.7 Discussion

6.7.1 About the distance of automaton $\mathcal{A}_F$ before t_1

In Section 6.2, we detailed a particular aspect of the automaton: if the entire system does not evolve between a time $t' < t_1$ until t_2, the distance computed by $\mathcal{A}_F$ equal the first part of Equation 6.1. However, with this behaviour, if the system evolves but the number of species x_O remains the same, the distance will be updated. For example, in Figure 6.1, if reactions occur between $[t_1, t_2] = [8.0, 10.0]$ that does not affect the copy number of P, the output of $\mathcal{A}_F$ is d_2 (the smaller one). It can be problematical if there are reactions completely uncorrelated to the others that can still occur (for example, adding a reaction $\emptyset \to A$ to the ER system).

If one wants to change this behaviour, a possible solution is to add an extra variable n' that stores the second to last value of the number of the studied species O. Then, the edges $l_1 \to l_3$ must contain a new predicate $n \neq n'$ to know whether the automaton should update the distance or not. This approach increases the computational time of the synchronised simulation.

Another solution is to defined the set $\mathbf{R^O} \subseteq ALL$ of reactions that change the number of the studied species O. If $R \in \mathbf{R^O}$, the species O is a reactant or a product of the reaction R (its population number changes). If $R \in ALL \setminus \mathbf{R^O}$ occurs, the number of species O remains equal. Then, we only have to change ALL by $\mathbf{R^O}$ in the automata edges. This also improves the computational cost, because n and the distance d are updated only when a reaction in $\mathbf{R^O}$ occurs.

6.7.2 Linear Hybrid Automata for non-elementary regions

The presented automata and results were developed for elementary regions μ. However, the automata are extendable to non-elementary regions. Here, we discuss two cases that induce non-elementary regions and describe automatic rules to construct automata for these non-elementary regions.

First, when the condition implies two species O and P, we have to make disjunction elimination. Let us consider $\mu = (x_1^{(O)} \leq x_O \leq x_2^{(O)}) \vee (x_1^{(P)} \leq x_P \leq x_2^{(P)})$. We duplicate automaton variables (e.g d becomes $d^{(O)}$ and $d^{(P)}$). Each asynchronous edge of the two automata can be written as $l \xrightarrow{\gamma_t \wedge \gamma_O, U_O} l'$ where:

- γ_t refers to the conditions relative to the time t and γ_O the other ones (these are the conditions that depend on $x_O, x_1^{(O)}, x_2^{(O)}$ in our case),
- U_O refers to the updates of variables related to O (all of the updates are related to O in our case).

Thus, any asynchronous edge has to be replaced by three edges:

- $l \xrightarrow{\gamma_t \wedge \gamma_O \wedge \gamma_P, \{U_O, U_P\}} l'$
- $l \xrightarrow{\gamma_t \wedge \gamma_O \wedge \neg\gamma_P, \{U_O\}} l'$
- $l \xrightarrow{\gamma_t \wedge \neg\gamma_O \wedge \gamma_P, \{U_P\}} l'$

Each edge that goes to the final location has to update the final distance, $d_{final} = \min(d^{(O)}, d^{(P)})$. Also, the synchronised transitions must update $n^{(O)}$ and $n^{(P)}$.

Second, for a species O, we consider $\mu = (x_1 \leq x_O \leq x_2) \vee (x_3 \leq x_O \leq x_4)$ when it is not an elementary region (it means $x_1 \leq x_2 < x_3 \leq x_4$). Thus, one can apply the methodology detailed above by taking $P = O$ and $x_1^{(O)}, x_2^{(O)}, x_1^{(P)}, x_2^{(P)} = x_1, x_2, x_3, x_4$.

These rules can be iterated to obtain automata with non-elementary regions that define $n \in \mathbb{N}$ elementary regions. Finally, one can construct a new $\mathcal{A}_{G \wedge F}$ automata based on these transformations. Indeed, this automaton is composed of the two automata $\mathcal{A}_G$ and $\mathcal{A}_F$ linked by an asynchronous transition.

6.8 Summary

In this Chapter:

- We have developed a new method to estimate the satisfaction probability function related to time-bounded reachability problems. The new procedure involves automaton-ABC with satisfiability distances between a trajectory and a property, computed by synchronised simulation of a CTMC with a Linear Hybrid Automaton.
- We have proved their efficiency in terms of correctness of the results and the efficiency of the implemented methods over several biological models.

Several tracks are worth considering to continue and improve this work:

- Any satisfiability distance in accordance with an MITL formula can be considered if the Proposition 6.2.1 still holds.
- Develop a program in our package that automatically creates the extended automata based on rules detailed in Section 6.7.2, and design an automaton for all types of Until formulae.
- Creates a new procedure of parameter synthesis based on automaton-ABC. Indeed, in Algorithm 18 the considered ϵ tolerance is zero in this new ABC procedure: it guarantees the link of the ABC posterior with the satisfaction probability function. However, one can think to relax this tolerance to have better confidence over the subset of parameters that do not fulfil the property φ (i.e. the complementary of the parameter space subset estimated by automaton-ABC).
- Describe how automaton-ABC statistically behaves if the trajectories are approximatively simulated with, for example, Tauleap Algorithm 3.

Chapter 7

Conclusion

Chemical Reaction Networks model a wide range of biological phenomena. If traditionally their dynamics are described by Ordinary Differential Equations, this approach is not relevant when the population is low. It is especially the case when one studies interactions between molecular species at the scale of the living cell (genetic regulatory networks, signalling pathways). By the reduction of such assumption, stochastic dynamics of CRN are described by CTMCs.

This thesis addressed several tasks of statistical inference and verification for Continuous-Time Markov Chains described by Chemical Reaction Networks. We have endeavoured to prove the efficiency of the synchronised simulation of a CTMC with a Linear Hybrid Automaton within the ABC algorithm. This new method called automaton-ABC has been applied to several problems from Systems Biology.

First, we gave an overview of the literature for Bayesian statistical inference and statistical model checking for parametric CTMCs. We introduced the notations from both fields, detailed what makes the considered problems complex, and discussed state-of-the-art techniques.

Second, we used automaton-ABC for statistical inference of oscillatory genetic networks. The nature of observations concerns the oscillations. They are described by sentences such as "we observe N oscillations between values L and H with mean period duration $\bar{t_p}^{(obs)}$". This qualitative behaviour is measured by an LHA $\mathcal{A}_{per}$ (Ballarini et al., 2015), and we define a distance over such property of an oscillatory trajectory. The use of automaton-ABC over two oscillatory models (doping three-way oscillator and repressilator) allowed to show the parameter space subsets where specific kind of oscillations are the most probable.

Third, we used automaton-ABC to gain computational time in classical inference. HASL formalism offers a framework that tailors the simulation according to a

specification. In this context, it was used to stop the simulation each time the Euclidean distance to observations exceeds the tolerance ϵ in the ABC procedure. In an experiment with the repressilator model, we showed that ABC computational time was divided by more than three compared to classical ABC.

Fourth, we used automaton-ABC for time-bounded reachability problems. The satisfiability distances represent the distance of a trajectory σ from specific forms of MITL formula φ. We designed LHA to compute these distances and used automaton-ABC to address quantitative model checking for parametric CTMCs. More precisely, we highlighted the link between the satisfaction probability function $Pr(\varphi; \mathcal{M})$ and the automaton-ABC posterior with $\mathcal{A}_\varphi$ named $\pi_{\varphi-ABC}$. Automaton-ABC-SMC allows exploring the parameter space concerning the formula φ efficiently. The results were obtained over several models of CRN and constantly compared to either an analytical solution, model checking with Prism, or statistical model checking with Cosmos or Prism. Due to optimisation and parallelisation of our implementation, the runs of automaton-ABC did not exceed two minutes, and outperformed state-of-the-art statistical model checking methods.

Finally, since these methods mix several concepts from both fields, we have tried to make their use accessible and efficient. We chose to use Julia because of its high-level syntax, efficiency, and great possibility of metaprogramming and code generation. It led to a Julia package that is much detailed in Appendix A. All the Julia scripts related to the results are documented within the Chapters, and the Scripts List above the introduction.

7.1 Limits and Perspectives

7.1.1 Scope of our work

This work focused on CTMCs defined by Chemical Reaction Networks. Indeed, they quickly challenge statistical inference or verification problems. However, the methods are easily generalisable to generalised semi-Markov processes (which corresponds to Generalised Stochastic Petri Nets).

7.1.2 Automaton-ABC for statistical inference

In Section 5.3, we gave a new approach to study oscillations in genetic networks whose stochastic dynamics are described by CTMCs. Another qualitative behaviour

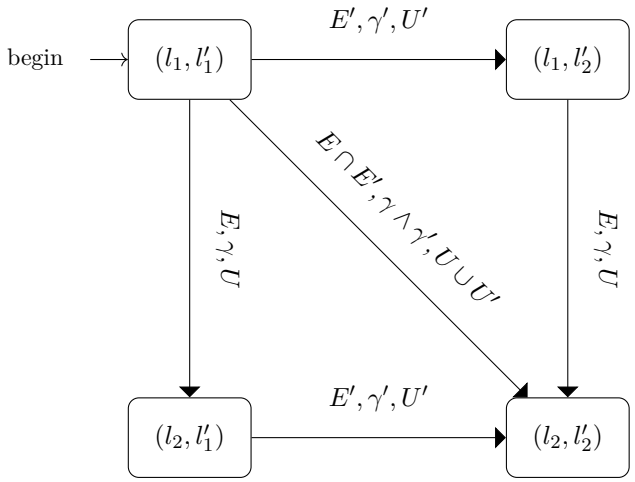
much observed in biological systems is bistability (Tyson et al., 2008). Bistable models are characterised by the existence of two steady states (and possibly unstable states). However, such stochastic models can produce trajectories that switch between two steady states. Thus, one can imagine an LHA that would characterises this behaviour, for example, by quantifying the time spent in a detected steady state or the duration time of a transition between two steady states, and use this LHA within the automaton-ABC procedure to identify the regions of the parameter space that reproduces this behaviour.

7.1.3 Automaton-ABC for time-bounded reachability

As already mentioned in Section 6.7.2, the method only covers a fragment of MITL. Further work has to be done to cover all MITL formulae. Instead of trying to design an automaton for each kind of formula, defining operations on HASL trajectory formulae could improve the coverage of MITL.

To illustrate this, we consider the computation of satisfiability distances of MITL formulae $\varphi_1 \wedge \varphi_2$ or $\varphi_1 \vee \varphi_2$. We suppose φ_1 and φ_2 are formulae whose satisfiability distances are computed by already defined LHA $\mathcal{A}_{\varphi_1}$ and $\mathcal{A}_{\varphi_2}$ with HASL trajectory formulae $(Last(d_{\varphi_1}), \mathcal{A}_{\varphi_1})$ and $(Last(d_{\varphi_2}), \mathcal{A}_{\varphi_2})$. The idea is to define an LHA $\mathcal{A}_{\varphi_1} \times \mathcal{A}_{\varphi_2}$ that integrates the behaviours of $\mathcal{A}_{\varphi_1}$ and $\mathcal{A}_{\varphi_2}$ into one LHA. One can construct this automaton by defining a product over two LHA, whose location set is the cartesian product of $\mathcal{A}_{\varphi_1}$ and $\mathcal{A}_{\varphi_2}$; variables and flows are the concatenation of those from the two automata. For example, let us consider these two simple LHA:

Its product could be defined as:



With such operations, the distances from both formulae φ_1 and φ_2 would be computed even if simultaneous updates of the two automata variables occur. Indeed, $(l_1, l'_1) \rightarrow (l_2, l'_2)$ is fired if the occurred event is in $E \cap E'$ and the predicate $\gamma \wedge \gamma'$ is fulfilled. Thus, the satisfiability distance of $\varphi_1 \wedge \varphi_2$ could be $(Last(d_{\varphi_1} + d_{\varphi_2}), \mathcal{A}_{\varphi_1} \times \mathcal{A}_{\varphi_2})$. For $\varphi_1 \vee \varphi_2$, the satisfiability distance could be $(Last(d_{\varphi_1} * d_{\varphi_2}), \mathcal{A}_{\varphi_1} \times \mathcal{A}_{\varphi_2})$.

As (Bortolussi and Silvetti, 2018) did with (Bortolussi, Milios, and Sanguinetti, 2016), a statistical parameter synthesis method may be constructed on top of automaton-ABC for time-bounded reachability. The stake is to statistically control the approximation of kernel density estimator based on automaton-ABC samples. For example, one can relax the tolerance by setting $\epsilon > 0$ to have better confidence for parameters that do not verify a formula φ.

In complex CTMC models, exact simulation has a high computational cost. One can use approximate simulation methods such as Tauleaping (Algorithm 3) to reduce the computational cost. The smaller the time resolution τ, the better the simulation approximation. The Multi-Level ABC method (Lester, 2018), evoked in Section 5.1.2, could be a solution to improve the performance of our automaton-ABC procedure.

7.1.4 Implementation

In our implementation detailed in Appendix A, defining a CTMC by a CRN and apply ABC-SMC and automaton-ABC-SMC is easy with the LHA presented in this thesis. However, further work will be performed to improve our tool:

- It is possible but challenging (without help) to design its own LHA. Thus, work is in progress to render the creation of LHA as simple as CTMCs in the package.

- Our implementation does not cover the whole HASL expression grammar, which would be interesting for more sophisticated runs of automaton-ABC.

7.2 Last words

Many phenomena in Systems Biology involve complex behaviours that are stochastic and complex. Performing inference or verification on such models is generally difficult. In this thesis, we attempted to show how Bayesian inference and statistical verification, via the Approximate Bayesian Computation (ABC) methods and the Hybrid Automata Stochastic Language (HASL), could both benefit from each other. Bayesian inference has benefited from HASL by its capacity of formally describe such behaviours. Statistical verification has benefited from Bayesian inference by its efficiency to explore the parameter space of statistical models.

Appendix A

MarkovProcesses.jl : A Julia package for efficient simulation, statistical inference and verification methods of Markov Processes.

A.1 Introduction

This thesis introduced a framework for statistical inference and verification of CTMCs with quite heavy notations and concepts. Thus, there is a need of developing an interface that allows users to be familiar with these methods easily and run them efficiently.

We choose Julia (Bezanson et al., 2014) as a programming language to address this task because it is both an efficient language and an easily usable one. Julia is a high-level multi-paradigm language, combining functional and object-oriented programming. It is adapted to mathematical notations, near Python syntax, can easily call and import Python and R objects, and can interact easily with C and Fortran code.

We wanted to construct a package that implements the concepts presented in this thesis with qualities such as:

- Computationally efficient.
- Easy to use.

- Deeply tested, both in terms of execution and statistical relevance.

Efforts have been made to conjugate these three requirements. The two first ones, which seem contradictory, are reachable via the use of the Julia language. Indeed, Julia allows a simple mathematical oriented syntax, but performance is reachable with 100% Julia code using computational diagnosis tools mainly described in the web page (JuliaLang, 2020).

Remark A.1.1 (Some Julia notations)
In the following, `T1<:AT` *means the type* `T1` *is a subtype of* `AT`*. It can be interpreted as "T1 inherits from AT" in classical object-oriented programming.* `var::T1` *means* `var` *has type* `T1` *(*`typeof(var)<:T1`*).*

In Julia, naming a function with an exclamation mark (for example, `set_x0!`*) is a convention to tell the user that one of the variables may be modified to prevent side effects.*

Script 11 (MarkovProcesses.jl package)
MarkovProcesses.jl package is available at the git repository `https://gitlab-research.centralesupelec.fr/2017bentrioum/markovprocesses.jl`*.*

A.2 A few introductory examples

A.2.1 Simulation of the SIR model

Let us start with the simulation of the SIR model.

```
1  using MarkovProcesses
2  load_plots()
3  load_model("SIR")
4  σ = simulate(SIR)
5  plot(σ)
```

Listing A.1: Simulation of the SIR model.

Code A.1 simulates the SIR model and plots Figure A.1. Let us describe this code line by line.

i. `using MarkovProcesses` loads the package.

ii. `load_plots()` loads methods about plotting special objects of our package such as `Trajectory` .

iii. `load_model("SIR")` searches on the `models/` directory of the package repository and includes the file `SIR.jl`. It creates a variable called `SIR` of type `T<:ContinuousTimeModel` (this means `T` is a type that inherits from `ContinuousTimeModel` in Julia language).

iv. `σ = simulate(SIR)` simulates the `SIR` model and stores the simulation in the variable σ. It has a type `typeof(σ)<:AbstractTrajectory` that stores the values of the simulations, the times and the transitions/events.

v. `plot(σ)` plots the trajectory σ.

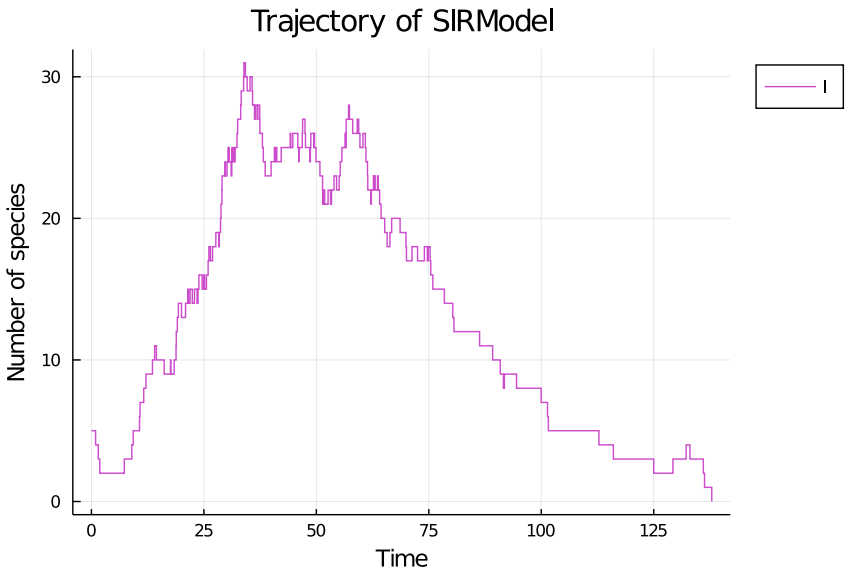


FIGURE A.1: Simulation of the SIR model with observation function $g = [:I]$.

Script 12 (Simulation of the SIR model with MarkovProcesses.jl)
Figure A.1 can be generated by running the Julia script located in `code/appendix_pkg/sim_sir.jl`.

One can notice that the plotted trajectory has only one dimension. Indeed, any `Model` object contains an observation model described by a function g (defined in Section 5.1.1 of Chapter 5). In this case, g is the function $(X_S, X_I, X_R) \to X_I$, implemented in Julia by `SIR.g = [:I]` . The output of `println(SIR)` is given in Listing A.2.

```
SIR SSA pkg model (ContinuousTimeModel)
- variables :
* S (index = 1 in state space)
* I (index = 2 in state space)
* R (index = 3 in state space)
- parameters :
* kr (index = 2 in parameter space)
* ki (index = 1 in parameter space)
- transitions : R1,R2
- observed variables :
* I (index = 1 in observed state space, index = 2 in state space)
p = [0.0012, 0.05]
x0 = [95, 5, 0]
t0 = 0.0
time bound = Inf
```

LISTING A.2: Output of the print of the SIR variable created in Code A.1.

One can easily create a model that is not available in `models/` thanks to the `@network_model` macro.

```
1        my_model = @network_model begin
2        Infection: (S+I => 2I, ki*S*I)
3        Recovery: (I => R, kr*I)
4        end "My awesome SIR"
```

LISTING A.3: Creation of the SIR model with @network_model.

Code A.3 creates a SIR model and stores it in `my_model` . Each reaction is described by a couple of expressions. The first expression describes the interactions between species (*reactants* => *products*). The second expression describes the kinetic rate (Equation 2.4). Each symbol that cannot be found in any reaction is considered as a parameter. The output of `println(my_model)` is given in Listing A.4.

```
My awesome SIR model (ContinuousTimeModel)
- variables :
* S (index = 1 in state space)
* I (index = 2 in state space)
* R (index = 3 in state space)
- parameters :
* kr (index = 2 in parameter space)
* ki (index = 1 in parameter space)
- transitions : Infection,Recovery
- observed variables :
* S (index = 1 in observed state space, index = 1 in state space)
* I (index = 2 in observed state space, index = 2 in state space)
* R (index = 3 in observed state space, index = 3 in state space)
p = [0.0, 0.0]
x0 = [0, 0, 0]
t0 = 0.0
time bound = Inf
```

Listing A.4: Output of the model created by a user in Code A.3.

The parameters and initial state of models created with `@network_model` are set to zero, and all the variables are observed. One can change that with `set_x0!` , `set_param!` , `set_time_bound!` or `set_observed_var!` .

A.2.2 Simulation of the ER model synchronised with $\mathcal{A}_F$ automaton

The package also allows the simulation of a CTMC synchronised with a Linear Hybrid Automaton described in Section 4.4 of Chapter 4.

```
using MarkovProcesses
load_plots()
load_model("ER")
load_automaton("automaton_F")
A_F = create_automaton_F(ER, 50.0, 75.0, 0.025, 0.05, :P)
sync_ER = A_F * ER
σ = simulate(sync_ER)
plot(σ)
```

Listing A.5: Simulation of the ER model synchronised with a $\mathcal{A}_F$ automaton.

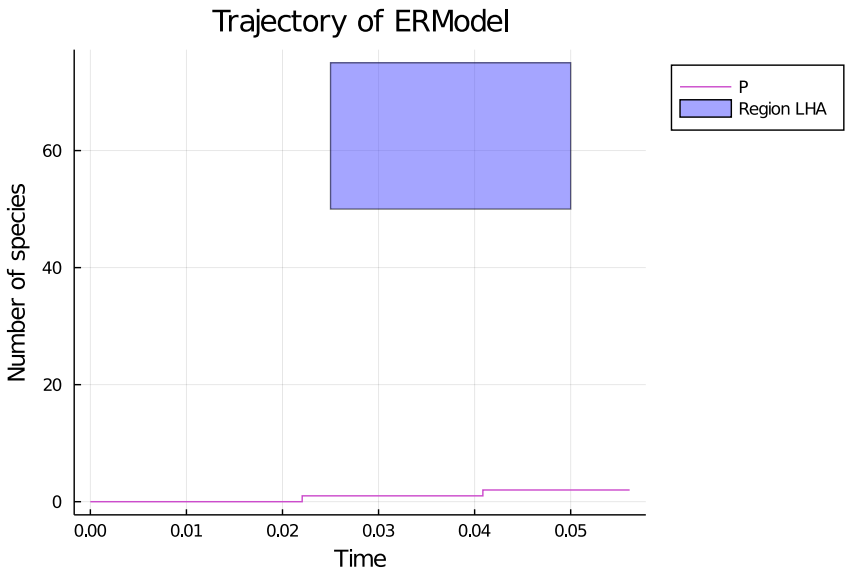


FIGURE A.2: Simulation of the ER model synchronised with a $\mathcal{A}_F$ automaton.

Code A.5 simulates the Enzymatic Reaction model and plots Figure A.2. A watchful eye may have noticed that this the configuration of $\mathcal{A}_F$ in experiment R1 of Section 6.5.2. Let us describe the crucial lines of the code.

i. `load_model("ER")` loads the ER model.

ii. `load_automaton("automaton_F")` loads a function `create_automaton_F` that creates an automaton $\mathcal{A}_F$ described in Figure 6.3 of Chapter 6.

iii. `A_F = create_automaton_F(ER, 50.0, 75.0, 0.025, 0.05, :P)` creates an automaton $\mathcal{A}_F$ with constants $x_1 = 50$, $x_2 = 100$, $t_1 = 0.025$, $t_2 = 0.05$.

iv. `σ = simulate(sync_ER)` simulates the `sync_ER` model and stores the simulation in the variable σ.

v. `plot(σ)` plots the trajectory σ.

Script 13 (Synchronised simulation of the ER model with MarkovProcesses.jl)
Figure A.2 can be generated by running the Julia script located in `code/appendix_pkg/sim_sync_er.jl`.

A.2.3 Run of the automaton-ABC algorithm

```
1  using MarkovProcesses
2  using Plots
3  load_model("ER")
4  load_automaton("automaton_F")
5  A_F_R1 = create_automaton_F(ER, 50.0, 75.0, 0.025, 0.05, :P)
6  sync_ER = A_F_R1*ER
7  parametric_sync_ER = ParametricModel(sync_ER, (:k3, Uniform(0.0,
   ↪ 100.0)))
8  res = automaton_abc(parametric_sync_ER; nbr_particles = 1000)
9  histogram(res.mat_p_end', weights = res.weights, normalize =
   ↪ :density)
```

LISTING A.6: Automaton-ABC: Experience R1 of ER model with $\mathcal{A}_F$ automaton.

Code A.6 runs the automaton-ABC algorithm of the ER model with $\mathcal{A}_F$ and $k_3 \sim \mathcal{U}(0, 100)$. The result is showed in Figure A.6. Let us describe the last three lines:

- Line 7 defines a prior distribution on the parameter k_3 for the `SynchronizedModel` `sync_er` . It is represented by the creation of the variable `parametric_sync_er` (of type `ParametricModel`).
- Line 8 runs the automaton-ABC algorithm and stores the results in `res` . One can access the ABC empirical posterior distribution by `res.mat_p_end` . As there is only one degree of freedom in our parametric model, the size of `mat_p_end` is 1x1000.
- Line 9 plots the histogram of the ABC empirical distribution.

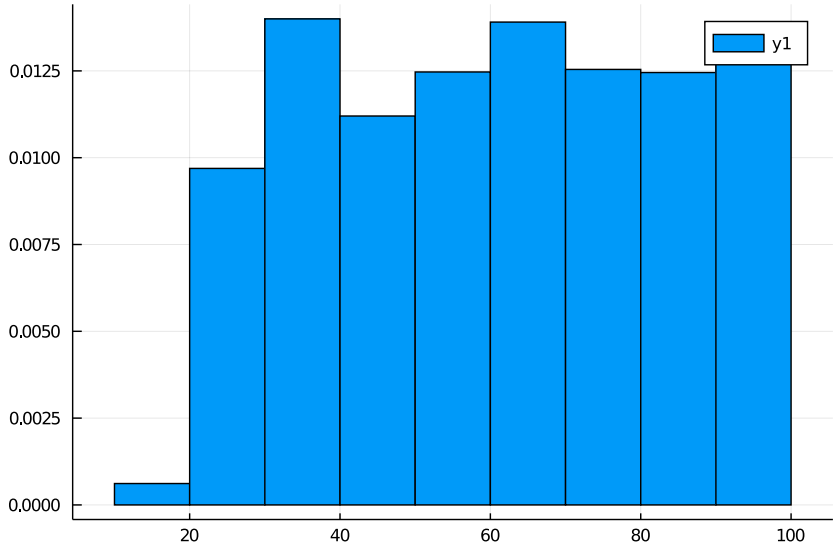


FIGURE A.3: Simulation of the ER model synchronised with a $\mathcal{A}_F$ automaton.

Script 14 (Experiment R1 of automaton-ABC with MarkovProcesses.jl)
Figure A.3 can be generated by running the Julia script located in `code/appendix_pkg/R1.jl`.

A.3 Structure of the package

The package proposes several tools for Markovian models, i.e. models whose transition function only depends on the last value of the state. In its minimum requirements, it has to:

i. Simulate a CTMC,

ii. Model Linear Hybrid Automata,

iii. Implements the synchronised simulation detailed in Section 4.4 of Chapter 4,

iv. Run automaton-ABC-SMC (Algorithm 16) and ABC-SMC (Algorithm 14).

This is the minimal requirement list. It can, and it will be extended to other features. The idea is to present a framework where statistical inference or verification algorithms can be implemented easily with the framework.

The source code is organised as follows:

- `core/common.jl`: Contains the definition of the common types (`struct`) and constructors of the variables of our package. This file regroups all the required type definitions.
- `core/model.jl`: Contains methods related to models, for example, simulation, access or modification of fields.
- `core/trajectory.jl`: Contains methods related to trajectories (a trajectory is a type of variable returned by a simulation of a model).
- `core/lha.jl`: Contains methods related to Linear Hybrid Automata.
- `core/network_model.jl`: Implementation of `@network_model` .
- `models/`: Each file is an implementation of a model. It creates a variable whose type is `T <: Model` .
- `algorithms/`: Contains statistical inference and verification algorithms.
- `tests/`: Contains tests of our package powered by the Test Julia package.
- `bench/`: Contains benchmarks of several tasks powered by the BenchmarkTools Julia package.

A.4 Type diagram

In this section, we describe the general structure of the project. The package was developed with an object-oriented point of view because it is more suited for the various concepts in this thesis.

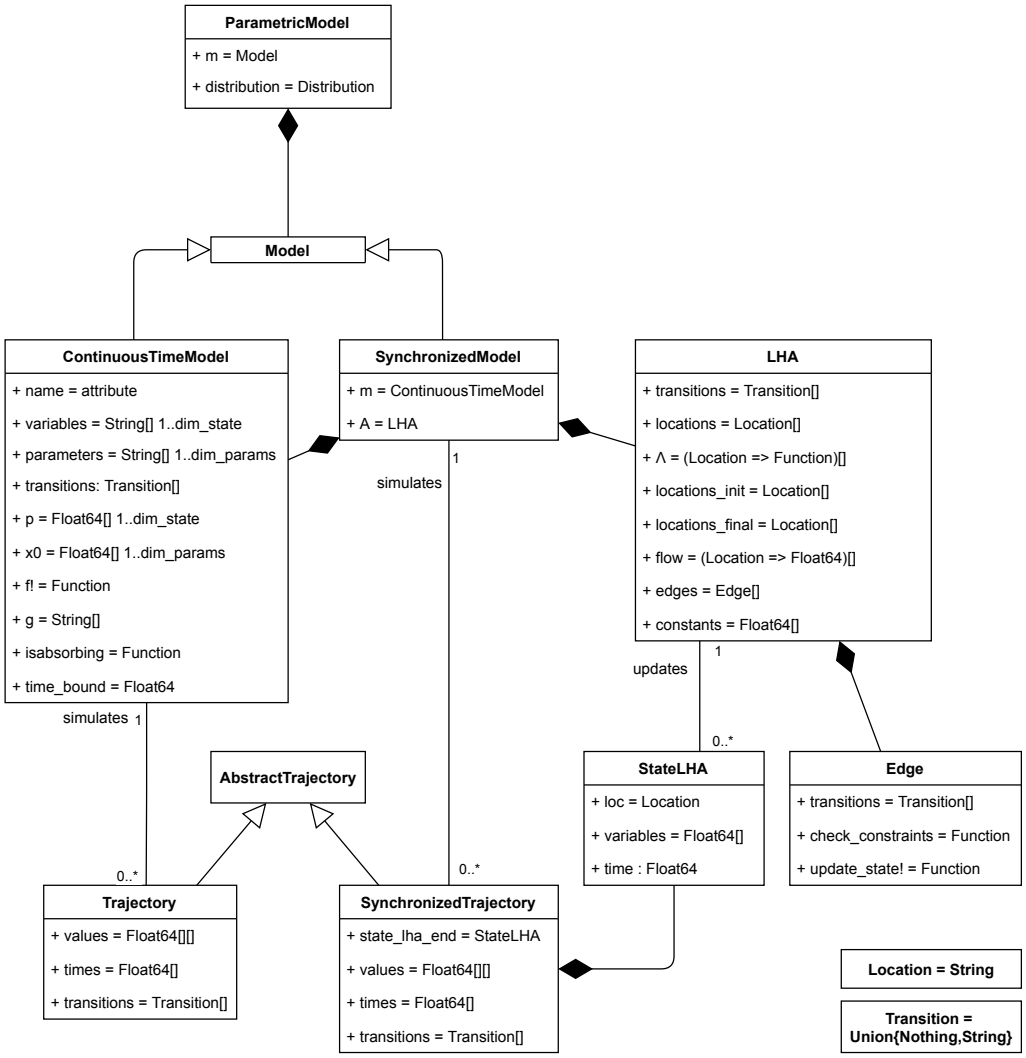


FIGURE A.4: UML graph of the different types defined in the package.

Figure A.4 is a UML description of the types/classes involved in the package. Let us describe these types with a top-down approach.

i. A `ParametricModel` is a `Model` associated with a prior distribution over the parameters of the model. It implements Definition 4.3.1 with a prior over the parameter space.

ii. A `Model` is an `abstract type` . Any type that is inherited from `Model` should have the possibility to be simulated, i.e. a method `simulate(model::T) where T <: Model` should be implemented.

iii. A `ContinuousTimeModel` is a Markovian model, i.e. a model that can be simulated only by knowing the previous state. In practice, it means we can define a function `f!(xnplus1, ..., xn,...)` that updates the state $n+1$ by accessing the state x_n. More details are discussed later in Section A.5.1.

iv. An `LHA` is a Linear Hybrid Automaton. The fields follow Definition 4.4.2.

v. A `SynchronizedModel` is a `ContinuousTimeModel` synchronised with an `LHA` .

vi. An `AbstractTrajectory` is a simulation/observation of a model. A `Trajectory` is the result of a `ContinuousTimeModel` and a `SynchronizedTrajectory` is the result of a `SynchronizedModel` .

A.5 Implementation

A.5.1 Simulation of `ContinuousTimeModel`

In this section, we describe the simulation of an object `model` of type `ContinuousTimeModel` .

The method `simulate(model::ContinuousTimeModel)` relies on the evaluation of a method `model.f!` , which must have the signature:

```
f!(xnplus1::Vector{Int}, l_t::Vector{Float64},
   l_tr::Vector{Union{Nothing,String}},
   xn::Vector{Int}, tn::Float64, p::Vector{Float64})
```

This function represents the transition function between two time steps of any simulation. It can either compute the simulation of the next reaction as in the SSA (Algorithm 1) or a leap of several reactions as in the tau leap Algorithm 3. The pseudo-code 19 shows how the model is simulated with this method.

Algorithm 19 Simulation of a ContinuousTimeModel.

Require: A ContinuousTimeModel with parameters p and initial state x_0.

Ensure: A Trajectory object that contains the simulation.

Init: $x_n \leftarrow x_0$, $t_n \leftarrow t_0$, $tr_n \leftarrow nothing$

Allocate : $values, times, transitions$

Allocate : x_{n+1}, l_t, l_tr

repeat

$f!(x_{n+1}, l\_t, l\_tr, x_n, t_n, p)$

$push!(values, x_{n+1})$

$push!(times, l\_t[1])$

$push!(transitions, l\_tr[1])$

Copy: $x_n, t_n, tr_n \leftarrow x_{n+1}, l\_t[1], l\_tr[1]$

until $t_n > time\_bound$ or x_n is an absorbing state.

The variables x_{n+1}, l_t, l_tr are vectors in which the simulation of the next time step is stored. This kind of implementation is used to decrease memory allocation, which gives better performance. That leads to l_t being a 1-length vector for the next time and l_{tr} being a 1-length vector for the next transition/event.

The pseudo-code is simplified. Among other things such as the observation of the model and how the variables are mapped to an index, we have implemented a buffering system for allocating memory at two levels in order to reach better performance.

- The field `model.estim_min_states` gives an approximative estimation of the number of states. Then, the simulator will allocate a vector of length `model.estim_min_states` for each variable instead of pushing the values step by step, requiring (approximately) reallocation at each step during execution. The vectors are resized if there are not enough simulation steps. By default, the value is set to 50.
- The field `model.buffer_size` is the length of the buffer after reaching `model.estim_min_states` . Then, the same buffering system described above is executed. The default value is 10.

Even if there is a slight improvement of the performance, the gain is a bit disappointing because by default, Julia allocates and manages arrays ingeniously (for example preallocation).

A.5.2 Simulation of `SynchronizedModel`

In this section, we describe the simulation of an object `sync_model` of type `SynchronizedModel`. It is the synchronised simulation detailed in Section 4.4 of Chapter 4. `sync_model` is a couple of an `LHA` and a `ContinuousTimeModel`. The simulation is similar to a `ContinuousTimeModel`, except that at each simulation, we update a `StateLHA` related to the automaton, thanks to te `next_state!` method. Also, we add a stopping criterion: the state of the LHA is in one of the final locations.

Algorithm 20 Simulation of a SynchronizedModel.

Require: A well defined SynchronizedModel with LHA A, parameters p and initial state x_0.

Ensure: A SynchronizedTrajectory object that contains the simulation.

Init: $x_n \leftarrow x_0$, $t_n \leftarrow t_0$, $tr_n \leftarrow nothing$

$S_n \leftarrow init\_state(A, x_0)$

Allocate : $values, times, transitions$

Allocate : $S_{n+1}, x_{n+1}, l\_t, l\_tr$

repeat

$f!(x_{n+1}, l\_t, l\_tr, x_n, t_n, p)$

$next\_state!(S_{n+1}, x_{n+1}, l\_t, l\_tr, S_n, xn, p)$

$push!(values, x_{n+1})$

$push!(times, l\_t[1])$

$push!(transitions, l\_tr[1])$

Copy: $S_n, x_n, t_n, tr_n \leftarrow S_{n+1}, x_{n+1}, l\_t[1], l\_tr[1]$

until $t_n > time\_bound$ or x_n is an absorbing state or $S_n.loc$ is a final location.

A.5.3 About trajectories

An object `Trajectory` contains any simulation or observation of a `ContinuousTimeModel`. Values of a trajectory σ are easily accessible:

- `σ.I` or `σ[:I]` returns the whole 1-dimensional trajectory of the variable I.
- `σ.I[4]` or `σ[:I,4]` returns the value of variable I in the $4-t$ state.
- `σ[4]` returns the whole $4-th$ state. Be careful with this notation because it allocates a new array. This is due to the data structure of `σ.values` (`typeof(σ.values)<:Vector{Vector{Float64}}`).

- `times(σ)` returns the times of the transitions related to `transitions(σ)` (for example, the times when a reaction occurred in the SIR model). If a transition equals `nothing` , then no transition occurred at the time (for any simulation, `transitions(σ)[1] == nothing`). If the trajectory is simulated from a bounded model, then the last transition is `nothing` and vice versa.

An object `SynchronizedTrajectory` is similar to `Trajectory` , except it contains the last state of the LHA.

A.5.4 Synchronisation with `LHA`

The synchronisation of a `ContinuousTimeModel` with an `LHA` is mainly embodied by the next_state! method:

```
function next_state!(Snplus1::StateLHA, A::LHA,
                     xnplus1::Vector{Int}, tnplus1::Float64,
                     ↪ tr_nplus1::Transition,
                     Sn::StateLHA, xn::Vector{Int}, p::Vector{Float64};
                     ↪ verbose::Bool = false)
```

After the simulation of the next transition / event whose characteristics are stored in `xnplus1,tnplus1,tr_nplus1` (next state, next time, next transition), next_state! is executed. The automaton fires the edges until one synchronous edge that contains `tr_nplus1` is fired. The fire of an edge `E::Edge` is embodied by the execution of `E.update_state!(Snplus1, A.constants, xnplus1, p)` .

A.6 Use of ABC methods

A.6.1 Classical ABC

ABC-SMC Algorithm 14 is run through the method `abc_smc` :

```
function abc_smc(pm::ParametricModel, l_obs::AbstractVector,
↪ func_dist::Function;
                 nbr_particles::Int = 100, tolerance::Float64 = 0.0,
                 ↪ alpha::Float64 = 0.75,
```

```
3                duration_time::Float64 = Inf,
                 ↪  dir_results::Union{Nothing,String} = nothing,
4                bound_sim::Int = typemax(Int), save_iterations::Bool =
                 ↪  false,
5                init_mat_p::Union{Matrix{Float64},Nothing} = nothing,
6                init_weights::Union{Vector{Float64},Nothing} = nothing,
                 ↪  init_vec_dist::Union{Vector{Float64},Nothing} =
                 ↪  nothing)
```

- `func_dist(l_sim, l_obs)` is the distance function between simulations and observations. It corresponds to $\rho(\eta(y_sim), \eta(y_exp))$.
- `l_obs::Vector{T2}` is a collection of observations.
- `func_dist` must have the signature `func_dist(l_sim::Vector{T1}, l_obs::Vector{T2})` where `T1` is the type of the simulation returned by the model in `pm` .

The keyword arguments are:

- `nbr_particles` is the number of particles
- `tolerance` is the final tolerance ϵ.
- `duration_time` is a threshold for the time execution. When it is exceeded, the algorithm finishes its current execution and ends.
- `bound_sim` is a threshold for the number of simulations. When it is exceeded, the algorithm finishes its current execution and ends.
- If `save_iterations` equals `true` , the parameters matrix, the associated weights and computed distances are saved at each iteration.
- If `init_mat_p` (a parameters' matrix), `init_weights` (a weights' vector) and `init_vec_dist` (a distances' vector) are specified, the ABC-SMC algorithm begins with these values instead of simulating from the prior.

Examples can be found in Script 4.

A.6.2 ABC with synchronised simulation

Automaton-ABC-SMC Algorithm 16 is run through the method `automaton_abc` :

```
function automaton_abc(pm::ParametricModel, l_obs::AbstractVector,
    func_dist::Function;
                       nbr_particles::Int = 100, tolerance::Float64 = 0.0,
                       alpha::Float64 = 0.75,
                       duration_time::Float64 = Inf,
                       dir_results::Union{Nothing,String} = nothing,
                       bound_sim::Int = typemax(Int), save_iterations::Bool
                       = false,
                       sym_var_aut::VariableAutomaton = :d,
                       init_mat_p::Union{Matrix{Float64},Nothing} = nothing,
                       init_weights::Union{Vector{Float64},Nothing} =
                       nothing,
                       init_vec_dist::Union{Vector{Float64},Nothing} =
                       nothing)
```

The `ParametricModel pm` has to contain a `SynchronizedModel`. Keywords are the same as described in Section A.6.1 above. `sym_var_aut` is the last value of the synchronised simulation to keep. It corresponds to the HASL trajectory expression $Last(sym\_var\_aut)$ in the automaton-ABC algorithm. At the time of writing, the use of other HASL trajectory expression is not implemented.

Several examples with different automata are available:

- $\mathcal{A}_{per}$ period automaton in Script 5,
- $\mathcal{A}_{ABC,\epsilon}$ ABC euclidean distance automaton in Script 6,
- $\mathcal{A}_F, \mathcal{A}_G, \mathcal{A}_{G\wedge F}$ automata for satisfiability distances of time-bounded reachability in Script 10.

A.7 Tests

In the following, we describe the tests of the package located in `tests/`. Each sub-directory represents a specific group of tests, and each Julia file of a sub-directory is a test.

For example, `tests/unit` contains all the unit tests of the package. Each file tests a specific characteristic. These tests are grouped in the file `tests/run_unit.jl` with the Test package and `@testset` macro. Running `tests/run_unit.jl` runs all the unit tests contained in `tests/unit`. This is done for every sub-directory.

One can run `tests/run_all.jl` to run all the tests group.

A.7.1 Execution test

These tests check if the execution behaves correctly for essential package functions.

- `tests/unit` groups the unit tests.
- `tests/dist_lp` groups the tests of the L^p distance between two trajectories. Some trajectories are hard-written to test specific behaviours, whereas other computations are tested with a Riemann sum.
- `tests/automata` groups tests of linear hybrid automata and synchronised simulations.
- `tests/automaton_abc` checks if `automaton_abc` method runs without errors.
- `tests/abc_smc` checks if `abc_smc` method runs without errors.

A.7.2 Cosmos based statistical tests

A more original approach to test our methods is done by statistically testing the behaviour of our algorithms with the statistical model checker Cosmos (Ballarini et al., 2015). Each automaton implemented in our package is written in Cosmos with a `.lha` file. For now, only the distance values of $\mathcal{A}_F, \mathcal{A}_G, \mathcal{A}_{G\wedge F}$ automata are tested.

The basic structure of a test follows:

- A configuration is set:
  - Type of model, initial state and parameters.
  - Type of automaton and its constants.
  - Statistical configuration: width, level
- The test runs a system call of Cosmos and parses the results files.
- The same configuration is run by the package, with the same Cosmos number of simulations.
- It tests if the package result of the average distance is statistically consistent with the width of the confidence interval. It also tests if all the simulated trajectories are accepted by both our package and Cosmos runs because automata $\mathcal{A}_F, \mathcal{A}_G, \mathcal{A}_{G\wedge F}$ were designed to accept all the trajectories.

These tests are located in `tests/cosmos`. For now, there are three Julia files, one file per automaton.

- `tests/cosmos/distance_F/ER_1D.jl` runs the test described above for R1, R2 and R3 configurations of the $\mathcal{A}_F$ automaton for the ER model described in Section 6.5.2. k_3 varies over the range $[0, 100]$ with a step of 10.
- `tests/cosmos/distance_G/ER_R5.jl` runs the test for R5 configurations of the automaton $\mathcal{A}_G$ for ER model. k_1 varies in $\{0, 0.5, 1.0, 1.5\}$ and k_2 in $\{0, 40, 80\}$.
- `tests/cosmos/distance_G_F/ER_R6.jl` runs the test for the R5 configuration of the automaton $\mathcal{A}_{G \wedge F}$ for the ER model. k_1 varies in $\{0, 0.5, 1.0, 1.5\}$ and k_2 varies in $\{0, 40, 80\}$.

We do not pretend these tests guarantee correct behaviours of any user-defined automaton, but the distances automata can be relatively trusted.

A.8 Benchmarks

In this thesis, the computational efficiency of simulations is critical because likelihood-free inference and statistical verification methods require a massive amount of simulations. In this section, we present benchmarks of the execution time of simulations. In this package, efforts have been made to reduce useless computations and limit memory allocation to reach better performances. All the benchmarks were run on a Dell XPS 9370 with CPU Intel i7 8550U @ 1.8GHz x 8 cores.

Script 15 (Benchmarks of MarkovProcesses.jl)
Benchmark scripts are available at `code/appendix_pkg/bench`. *Each file runs a benchmark and generates LaTeX files, which are included in this appendix.*

A.8.1 Versus Cosmos

Cosmos is a program developed in C++. It is a well-known compiled language very often used when performance is critical. Julia is a high-level dynamic language, and Julia's compiler produces LLVM native code, a compiler written in C++.

The idea of this benchmark is to show how close the package can be from a program written in C++. For such a scripting language as Julia, it is easy to use syntaxes that allocate too much memory considering our tasks. A simple illustrating

example is Julia's slicing array. The syntax `v[m:n]` creates a copy of the accessed vector v, which can be too computationally expensive if we only want to read these values (for the record, it is solvable with `@view`). Thus, we paid much attention to the computational cost of methods that can be executed a considerable number of times, such as `model.f!` or `next_state!` . For example, any `ContinuousTimeModel` created by `@network_model` creates a function `f!` that allocates 0 bytes after a few executions.

We designed several benchmarks:

- Benchmark 1: Compute the mean distance of $\mathcal{A}_G$ automaton of configuration R5 (ER model) with a width confidence interval equal to 0.1, confidence level equal to 0.99, and parameter vector $p = [1.5, 1.0, 1.0]$.
- Benchmark 2: Compute the mean distance of $\mathcal{A}_G$ automaton of configuration R5 (ER model) with a width equal to 0.01, confidence level equal to 0.99, and parameter vector $p = [1.5, 40.0, 1.0]$

Each benchmark implies the following steps:

- A configuration is set:
  - Type of model, initial state and parameters.
  - Type of automaton and its constants.
  - Statistical configuration: width of confidence interval, confidence level interval
- The benchmark runs a system call of Cosmos and parses the results files: number of simulations, time, used memory.
- The same task is run by the package with the same number of simulations.

Bench 1	Mean Time (s)	Max. Time (s)	Min. Time (s)	Mean Memory (MB)	Simulations
Package	3.41	5.13	2.59	20.51	20000
Cosmos	7.3	10.13	5.76	559.86	20000

TABLE A.1: Cosmos: Benchmark 1; 100 runs.

Bench 2	Mean Time (s)	Max. Time (s)	Min. Time (s)	Mean Memory (MB)	Simulations
Package	117.22	126.51	109.63	62.64	62000
Cosmos	179.04	195.98	167.41	549.94	62000

TABLE A.2: Cosmos: Benchmark 2; 5 runs.

Bench 2	Mean Time (s)	Max. Time (s)	Min. Time (s)	Mean Memory (MB)	Simulations
Package	47.68	51.14	44.19	X	62000
Cosmos	64.15	65.08	62.38	647.98	62000

TABLE A.3: Cosmos: Benchmark 2 with 8 jobs (parallel execution); 5 runs.

Table A.1 groups the results for Benchmark 1. We can see that all the execution statistics are of the same order. For the mean execution time, our package slightly outperforms Cosmos, and we use much less memory.

Table A.2 and Table A.3 group the results of Benchmark 2 with two configurations: a sequential run and a parallel run. We can see that our package outperforms Cosmos. However, the execution times are of the same order (the difference is always less than the execution time itself).

We can observe that the package allocates less memory compared to Cosmos. A deep memory allocation analysis was performed on the package to reduce efficiently useless allocations, which implies some specific knowledge about the Julia language.

These encouraging results need to be put into perspective because our package does not have all the functionalities of Cosmos, which is a more evolved machinery. Our package does not handle all HASL expressions, which requires computations at each simulation step. For now, only the last value of an automaton variable is available. Also, our benchmarks were not scaled to more complex models. Thus, it is impossible to state that our package will perform better for every model and task. However, it shows that our package is comparable in terms of execution time and performs better for the tasks considered in the thesis. This is quite challenging for such a high-level dynamic language as Julia compared to a C++ written program.

A.8.2 Versus Catalyst.jl/DifferentialEquations.jl

We developed a new package because we wanted to implement synchronisation of CTMC in Julia. Nevertheless, simulation of Chemical Reaction Networks is already available in Julia with Catalyst.jl. This package designs any model based on the Chemical Reaction Network representation. Then, this model can be used by the DifferentialEquations.jl package to solve any simulation problem, i.e. simulates the model by different methods such as Gillepsie or SDE. DifferentialEquations.jl is a well-known efficient package that can solve numerically any type of differential equations.

Thus, the combination of the two packages can simulate any `ContinuousTimeModel` , except that there is less flexibility in their `@reaction_network` macro than ours. Indeed, in Catalyst.jl, they assume mass-action law for the kinetics, whereas any form can be written for the propensity function in our package. In the following, we can compare the performance of our package to the combination of Catalyst.jl and DifferentialEquations.jl.

We made two benchmarks of performance:

- Benchmark 1: One simulation of the ER model with parameter vector $p = [0.2, 40.0, 1.0]$. This parameter vector is chosen because the number of steps is significant (at least 7000 states for a simulated trajectory in practice).
- Benchmark 2: Creation of the ER model + the simulation task described in Benchmark 1.

Bench 1	Mean Time (ms)	Max. Time (ms)	Min. Time (ms)	Mean Memory (KB)
Package	0.54	1.95	0.32	376.08
Catalyst.jl	0.91	3.6	0.56	807.22

TABLE A.4: Catalyst.jl: Benchmark 1

Table A.4 groups the results of Benchmark 1. We can observe that our mean execution time is 40% to that of Catalyst.jl. This performance gain is non-negligible if this task is repeated a considerable number of times, which is our case. Also, we allocate two times less memory while we also store the occurred reactions/transitions.

Bench 2	Mean Time (ms)	Max. Time (ms)	Min. Time (ms)	Mean Memory (KB)
Package	27.95	106.88	24.6	1303.95
Catalyst.jl	1.01	4.57	0.66	862.83

TABLE A.5: Catalyst.jl: Benchmark 2

Table A.5 groups the results of Benchmark 2. Creating a model is more costly for our package than Catalyst.jl. It needs to be put into perspective: creating a model only occurs a few time in a single run of a program.

These benchmarks do not even take into account the loading of modules. For example, a `using Catalyst` can take about 20 seconds at the first execution of a Julia session (but not the first-ever, which means the package is already precompiled), whereas `using MarkovProcesses` takes about 3 seconds. This is non-negligible if one distributes computations over an HPC cluster, for example.

A.9 Conclusion

A.9.1 Summary

We have developed a Julia package to simulate CTMCs and synchronised CTMCs with a Linear Hybrid Automata, and the automaton-based ABC methods developed in this thesis. The call of methods is high-level (it takes only a few lines). Tests were developed to verify our experiments statistically. Benchmarks were made to compare performance to other existing programs. We showed that our package performs better than one of Julia's reference package. Execution times are of the same order, which is encouraging, considering that Cosmos is a C++ program.

A.9.2 Perspectives

However, we believe this work builds the foundations of a tool with room for improvement in many ways.

- Use of any form of HASL trajectory expression in automaton-ABC.
- A macro that facilitates the creation of LHA. For now, they are Julia handwritten coded.

- Tests with other more complicated models.
- Statistical tests of created models with `--loop` option of Cosmos.
- Statistical tests with automata that are not related to time-bounded reachability.

For the integration of this package into Julia's community, perspectives are:

- Creation of a `ReactionSystem` object of the ModelingToolkit.jl API.
- Analyse how our methods can be integrated into the Turing.jl framework, which regroups many Bayesian inference methods.

Appendix B

Measure theory

This appendix aims at giving the main tools of probability theory for the construction of probability measures.

Section 2.2 constructs a measure and proves the equality of measures over a σ-algebra generated by a semiring. The consistency of the construction of these measures is based on Caratheodory's extension theorem.

B.1 Results from measure theory: Caratheodory's theorem

The definitions and theorems presented in this section are based on (Klenke, 2008, Chapter 1).

Let X be a set of elements. In the following definitions, $\mathcal{A} \subset \mathcal{P}(X)$ is a subset of the power set of X.

Definition B.1.1 (Semiring of sets.)
Let X be a set of elements and $\mathcal{A} \subset \mathcal{P}(X)$. $\mathcal{A}$ is called a semiring of sets if :

- *i) $\emptyset \in \mathcal{A}$.*
- *ii) It is stable by set difference ($A, B \in \mathcal{A} \Rightarrow A \setminus B$ is a finite disjoint union of elements of $\mathcal{A}$).*
- *iii) It is stable by intersection ($A, B \in \mathcal{A} \Rightarrow A \cap B \in \mathcal{A}$).*

Definition B.1.2 (additive set function.)
Let $\mu : \mathcal{A} \to \overline{\mathbb{R}_{\geq 0}}$ be a set function. μ is additive if for any $A_1, \ldots, A_n$ ($n \in \mathbb{N}*$) that

are disjoint and verifies $\bigcup_{i=1}^{n} A_n \in \mathcal{A}$ *we have :*

$$\mu(\bigcup_{i=1}^{n} A_n) = \sum_{i=1}^{n} \mu(A_n)$$

μ is called a content.

Definition B.1.3 (σ-subadditive function.)
Let $\mu : \mathcal{A} \to \overline{\mathbb{R}_{\geq 0}}$ be a set function. μ is σ-subadditive if for any $(A_n)_{n\in\mathbb{N}}$ and $A \subset \bigcup_{n=0}^{+\infty} A_n$ we have :

$$\mu(A) \leq \sum_{n=0}^{+\infty} \mu(A_n)$$

Definition B.1.4 (σ-finite content.)
Let $\mu : \mathcal{R} \to \overline{\mathbb{R}_{\geq 0}}$ be a content (an additive set function) over a semiring $\mathcal{R}$. μ is σ-finite if $\exists (R_n)_{n\in\mathbb{N}}$ countable set of elements in $\mathcal{R}$ s.t.

$$\bigcup_{n=0}^{+\infty} R_n = \mathcal{R} \wedge \forall n \in \mathbb{N}, \mu(R_n) < +\infty$$

Definition B.1.5 (Definition of the outer measure $\mu*$.)
The outer measure $\mu^* : \mathcal{P}(X) \to \mathbb{R}_{\geq 0}$ of a set function μ over a semiring $\mathcal{R}$ is defined as :

$$\mu^*(A) = \inf\{\sum_{k=0}^{+\infty} \mu(A_k) / A_k \in \mathcal{R} \text{ and } A \subset \bigcup_{n\in\mathbb{N}} A_n\}$$

It is monotone and σ-subadditive.

Theorem B.1.1 (Caratheodory's theorem for extended measure)
Let $\mathcal{R}$ be a semiring and let $\mu_0 : \mathcal{R} \to \overline{\mathbb{R}_{\geq 0}}$. If μ_0 is :

- *additive*
- *σ-subadditive*
- *σ-finite*

Then it admits a unique extended measure μ defined on $\mathcal{F}(\mathcal{R})$, and $\mu_0 = \mu_{|\mathcal{R}} = \mu^*_{|\mathcal{R}}$.

B.2 Semiring of sets of $Path(\mathcal{M})$

Let $\mathcal{M}$ a CTMC (Section 2.1.2). The probability measure $Pr_{\mathcal{M}}$ is defined on the cylinder sets:

$$C(\mathbf{s}_0, I_0, \ldots, I_{k-1}, \mathbf{s}_k) = \{\sigma \in Path(\mathcal{M}), \forall i \in \{0, \ldots, k\}, \sigma[i] = \mathbf{s}_i, \delta(\sigma, i) \in I_i\}$$

with $I_j =]a_j, b_j[$.

Let $\mathcal{R}$ be the set that contains all the cylinder sets $C(\mathbf{s}_0, I_0, \ldots, I_{k-1}, \mathbf{s}_k)$ ($\mathcal{R} \subset \mathcal{P}(Path(\mathcal{M}))$).

Proposition B.2.1 (Semiring of sets of trajectories.)
Let $\mathcal{M}$ be a CTMC defined by an initial distribution α and a transition rate matrix Q. Let :

$$\begin{aligned}\mathcal{R} =&\{C(\mathbf{s}_0, I_0, \ldots, I_{k-1}, \mathbf{s}_k), k \in \mathbb{N}, (\mathbf{s}_i \in \mathbf{S})_{0 \leq i \leq k},\\ &\textit{and for } i \in \{0, \ldots, k-1\}, I_i =]a_i, b_i[\textit{ with } a_i, b_i \in \mathbb{R}_{\geq 0}\} \cup \{\emptyset\}\end{aligned}$$

where $C(\mathbf{s}_0, I_0, \ldots, I_{k-1}, \mathbf{s}_k)$ is the set of paths $\sigma \in Path(\mathcal{M})$ with $\sigma[i] = \mathbf{s}_i$ and $\delta(\sigma, i) \in I_i$. Then $\mathcal{R}$ is a semiring of sets of trajectories.

Proof. ii) Stable by $\setminus$

Let $A, B \in \mathcal{R}$. By definition of $\mathcal{R}$, $A = C(\mathbf{s}_0^{(A)}, I_0^{(A)}, \ldots, I_{k_A-1}^{(A)}, \mathbf{s}_{k_A}^{(A)})$ and $B = C(\mathbf{s}_0^{(B)}, I_0^{(B)}, \ldots, I_{k_B-1}^{(B)}, \mathbf{s}_{k_B}^{(B)})$.

Let $k^- = \min(k_A, k_B)$.

If $\mathbf{s}_{0:k^-}^{(A)} \neq \mathbf{s}_{0:k^-}^{(B)}$, $A \setminus B = A$.

Otherwise, let us distinguish two cases.

If $k_B \leq k_A$, then we can define : $\forall i \leq k_B, I_i^{(A \setminus B)} = I_i^{(A)} \setminus I_i^{(B)}$ and $\forall k_B \leq i \leq k_A, I_i^{(A \setminus B)} = I_i^{(A)}$.

Then $A \setminus B = C(\mathbf{s}_0^{(A)}, I_0^{(A \setminus B)}, \ldots, I_{k_A-1}^{(A \setminus B)}, \mathbf{s}_{k_A}^{(A)})$.

If $k_B > k_A$, we can define $\forall i \leq k_A, I_i^{(A \setminus B)} = I_i^{(A)} \setminus I_i^{(B)}$ and $\forall k_A \leq i \leq k_B, I_i^{(A \setminus B)} = \mathbb{R}_{\geq 0} \setminus I_i^{(B)}$. Then,

$$A \setminus B = \bigcup_{\mathbf{s}^*_{0:(k_B-k_A-1)} \in D} C(\mathbf{s}_0^{(A)}, I_0^{(A\setminus B)}, \dots, I_{k_A-1}^{(A\setminus B)}, \mathbf{s}_{k_A}^{(A)}, I_{k_A}^{(A\setminus B)}, \mathbf{s}_0^*, \mathbb{R}_{\geq 0}, \dots, \mathbb{R}_{\geq 0}, \mathbf{s}^*_{k_B-k_A-1}) \cup C(\mathbf{s}_0^{(A)}, I_0^{(A\setminus B)}, \dots, I_{k_A-1}^{(A\setminus B)}, \mathbf{s}_{k_A}^{(A)}, I_{k_A}^{(A\setminus B)}, \mathbf{s}_{k_A+1}^{(B)}, I_{k_A+1}^{(A\setminus B)}, \dots, I_{k_B-1}^{(A\setminus B)}, \mathbf{s}_{k_B}^{(B)})$$

with $D = \{\mathbf{s}'_{0:(k_B-k_A-1)} \in \mathbf{S}^{k_B-k_A-1}$ with $\mathbf{s}'_{0:(k_B-k_A-1)} \neq \mathbf{s}^{(B)}_{k_A+1:k_B}$ and $Q(\mathbf{s}_A, \mathbf{s}'_0) > 0, Q(\mathbf{s}'_i, \mathbf{s}'_{i+1}) > 0\}$. In other words, $A \setminus B$ is the set of paths in A that does not have the sequence $\mathbf{s}^{(B)}_{k_A+1:k_B}$ at the end or have this sequence but out of the intervals of B.

This union is disjoint and also finite if for each state the number of transitions is finite.

iii) Stable by $\cap$

Let $A, B \in \mathcal{R}$. By definition of $\mathcal{R}$, $A = C(\mathbf{s}_0^{(A)}, I_0^{(A)}, \dots, I_{k_A-1}^{(A)}, \mathbf{s}_{k_A}^{(A)})$ and $B = C(\mathbf{s}_0^{(B)}, I_0^{(B)}, \dots, I_{p-1}^{(B)}, \mathbf{s}_{k_B}^{(B)})$. We suppose without loss of generality that $k_A \geq k_B$.

It appears that if $\exists i \leq k_B, \mathbf{s}_i^{(A)} \neq \mathbf{s}_i^{(B)}$, then $A \cap B = \emptyset$

In the other case, $A \cap B = C(\mathbf{s}_0^{(A\cap B)}, I_0^{(A\cap B)}, \dots, I_{k_A-1}^{(A\cap B)}, \mathbf{s}_{k_A}^{(A\cap B)})$ where $\forall i \leq k_A, \mathbf{s}_i^{(A\cap B)} = \mathbf{s}_i^{(A)}$ and $I_i^{(A\cap B)} = I_i^{(A)} \cap I_i^{(B)}$. With this form, $A \cap B \in \mathcal{R}$.

In both cases $A \cap B \in \mathcal{R}$. □

B.3 Statistical model of CTMCs

B.3.1 Density of a CTMC

In classical statistical theory, we often consider a set of probability measures $(P_\theta)_{\theta\in\Theta}$ called a statistical model. To have good statistical guarantees, we want our family of measures to be dominated by some measure. In this section, we construct a measure $\overline{\mu}$ independent of $\mathbb{P}_S$ so that $\overline{\mu}$ dominates $\mathbb{P}_S$ (i.e $\mathbb{P}_S \ll \overline{\mu}$, absolute continuity).

The set of trajectories of a CTMC can be seen as a subset of $(\mathbf{S}, \overline{\mathbb{R}_{\geq 0}})^{\mathbb{N}}$. Then, $C(k, \mathbf{s}_{0:k}, I_{0:k-1})$ can be rewritten as $\{(\mathbf{s}_0, \dots, \mathbf{s}_k)\} \times (I_{0:k-1} \cup \{+\infty\})$ which is a subset of $(\mathbf{S}, \overline{\mathbb{R}_{\geq 0}})^{k+1}$.

We can define a product measure on $\bigotimes_{i=0}^{k}(\mathbf{S}, \overline{\mathbb{R}_{\geq 0}})$. δ^{k+1} is the counting measure on $\mathbf{S}^{k+1}$ and λ^k is the Lebesgue measure on $\mathbb{R}^k$.

Let $k \in \mathbb{N}*$. Let $\mu : \mathcal{R} \to \mathbb{R}_{\geq 0}$ be the monotone set function defined on the semiring $\mathcal{R}$:

$$\mu(C(k, \mathbf{s}_{0:k}, I_{0:k-1})) = 1 \cdot \lambda^{(k)}\left(I_{0:k-1}\right)$$

Basically, this operation means that if $A \in \mathcal{R}$, then the measure of A is the volume of $I_0 \times \ldots \times I_{k-1}$. The measure of any intersection of sets of $\mathcal{R}$ is defined because it is a semiring, so stable by $\cap$.

The measure of any union of sets in $\mathcal{F}(\mathcal{R})$ is defined by the outer measure μ^* in Definition B.1.5.

Remark B.3.1
On any set $A = C(\mathbf{s}_0, I_0, \ldots, I_{k-1}, \mathbf{s}_k)$, A can be represented as $\{\mathbf{s}_{0:k}\} \times I_0 \times \ldots \times I_{k-1}$. In this case, $\mu(A) = (\delta^{(k+1)} \otimes \lambda^{(k)})(\{\mathbf{s}_{0:k}\} \times (I_0 \times \ldots \times I_{k-1}))$

Proposition B.3.1
μ is (i) additive, (ii) σ-subadditive and (iii) σ-finite on $\mathcal{F}(\mathcal{R})$.

Proof. (i) (ii) Here we do not dive into the details, but the σ-subadditivity of $\mu*$ becomes an additivity in $\mathcal{F}(\mathcal{R})$. This is done by the definition of $\mathcal{M}(\mu)$ that defines the μ-measurable sets which is proven to be a σ-algebra, so $\mathcal{F}(\mathcal{R})$ is contained in it (Lemma 1.50 and 1.51 in Klenke, 2008).

(iii) Let us prove μ is σ-finite.

$C(k, \mathbf{s}_{0:k}, [0, n]^k)$ is entirely defined by

$$(\mathbf{s}_{0:k}, n) = (\mathbf{s}_0[1], \ldots, \mathbf{s}_0[d], \ldots, \mathbf{s}_k[1], \ldots, \mathbf{s}_k[d], n) \in \mathbb{N}^{d\cdot(k+1)+1}$$

i.e. a stationary sequence of integers. The set of stationary sequences of integers called $\mathcal{S}_0$ is countable (as a countable union of the sets of stationary sequences of order $i \in \mathbb{N}$).

$\{C(\mathbf{s}_{0:k}, [0, n]^k)\}_{(\mathbf{s}_{0:k}, n) \in \mathcal{S}_0}$ is countable, the union of all is $(\mathbf{S}, \overline{\mathbb{R}_{\geq 0}})^{\mathbb{N}}$ (set of any trajectory) and $\forall (\mathbf{s}_{0:k}, n) \in \mathcal{S}_0, \mu(C(\mathbf{s}_{0:k}, [0, n])) < +\infty$ (by definition). μ is σ-finite.

□

Proposition B.3.2 (Unique measure on $\mathcal{F}(Path(\mathcal{M}))$ defined on $\mathcal{R}$.)
There exists a unique measure $\overline{\mu}$ *on* $\mathcal{F}(Path(\mathcal{M}))$ *with* $\overline{\mu}_{|\mathcal{R}} = \mu = \mu^*_{|\mathcal{R}}$.

Proof. By proposition B.3.1, we can apply the extension theorem of Caratheodory B.1.1. □

Proposition B.3.3 (Absolute continuity of $\mathbb{P}_S$.)
Let S *be a time-homogeneous CTMC. Then* $\mathbb{P}_S \ll \overline{\mu}$.

Proof. Let $A \in \mathcal{F}(Path(\mathcal{M}))$ with $\overline{\mu}(A) = 0$. Let us prove $\mathbb{P}_S(A) = 0$.

By Caratheodory's theorem B.1.1, $\mu^*(A) = \overline{\mu}(A)$ so $\exists (A_n)_{n\in\ \mathbb{N}}$ in $\mathcal{R}$ with $A \subset \bigcup_{n=0}^{+\infty} A_n$ s.t. $\overline{\mu}(A) = \sum_{k=0}^{+\infty} \mu(A_n)$. We can rewrite A_n as $A_n = C(k_n, \mathbf{s}_{0:k_n}^{(A_n)}, I_{0:k_n-1}^{(A_n)})$.

We have a series with positive values whose limit sum is equal to zero, hence all the terms of the series are equal to zero.

On the other hand, $\mu(A_n) = \lambda^{k_n}(I_{0:k_n-1}^{(A_n)}) = 0$, i.e. $\forall n, \exists i < k_n,]a_i^{(A_n)}, b_i^{(A_n)}[= \emptyset$. It follows $\mathbb{P}_S(A_n) \propto \prod_{i=0}^{k_n-1}(e^{-E(\mathbf{s}_i)a_i^{(A_n)}} - e^{-E(\mathbf{s}_i)b_i^{(A_n)}}) = 0$.

Then $\mathbb{P}_S{}^*(A) \leq \sum_{n=0}^{+\infty} \mathbb{P}_S(A_n) = 0$ by definition of the outer measure with inf.

We can conclude $\mathbb{P}_S \ll \overline{\mu}$. □

Corollary B.3.1 (Density of a CTMC)
Let S *be a CTMC.* $\mathbb{P}_S$ *admits a density* p_S *with respect to* $\overline{\mu}$ *:*

$$p_S : Path(\mathcal{M}) \to \mathbb{R}_{\geq 0}$$
$$\sigma = \left(\mathbf{s}_0 \xrightarrow{t_0} \dots \xrightarrow{t_{k-1}} \mathbf{s}_k\right) \to P(\mathbf{s}_0) \prod_{i=1}^{k} Q(\mathbf{s}_{i-1}, \mathbf{s}_i) e^{-E(\mathbf{s}_{i-1})t_{i-1}}$$

Proof. Let S be a time-homogeneous CTMC with a infinitesimal generator matrix Q. First, by Radon-Nykodym and Proposition B.3.3, $\mathbb{P}_S$ has a density $\frac{d\mathbb{P}_S}{d\overline{\mu}}$. Let

$T_{\mathbf{s}} \sim Exp(E(s))$, $p_{T_{\mathbf{s}}}$ its density and :

$$\begin{aligned}
p_S^{(k)} :& \bigotimes_{i=0}^{k} (\mathbf{S}, \overline{\mathbb{R}_{\geq 0}}) \to \mathbb{R}_{\geq 0} \\
& (\mathbf{s}'_{0:k}, t'_{0:k-1}) \to P(\mathbf{s}_0) \prod_{i=1}^{k} P(\mathbf{s}'_{i-1}, \mathbf{s}'_i) p_{T_{\mathbf{s}}}(t'_{i-1}) \\
p_S :& (\mathbf{S}, \overline{\mathbb{R}_{\geq 0}})^{\mathbb{N}} \to \mathbb{R}_{\geq 0} \\
& (\mathbf{s}'_{0:k}, t'_{0:k-1}) \to p_S^{(k)}(\mathbf{s}'_{0:k}, t'_{0:k}) \\
\mu_S :& \mathcal{F}(\mathcal{R}) \to \overline{\mathbb{R}_{\geq 0}} \\
& A \to \int_{\sigma \in A} p_s(\sigma) d\overline{\mu}(\sigma)
\end{aligned}$$

μ_S is by construction a measure (it is sometimes denoted as $\mu_S = p_S \cdot \overline{\mu}$). Let us prove $\mu_{S|\mathcal{R}} = \mathbb{P}_{S|\mathcal{R}}$.

Let $C \in \mathcal{R}$. Then $C = C(\mathbf{s}_{0:k}, I_{0:k-1})$ $(= \{\mathbf{s}_{0:k}\} \times (I_0 \times \ldots \times I_{k-1}))$

$$\begin{aligned}
\mu_S(C) &= \int_{\sigma \in C} p_S(\sigma) d(\delta^{(k+1)} \otimes \lambda^{(k)})(\sigma) (\text{Remark B.3.1}) \\
&= \iint_{(\mathbf{s}', t'_{0:k}) \in \{\mathbf{s}_{0:k}\} \times I_0 \times \ldots \times I_{k-1}} p_S^{(k)}(\mathbf{s}', t') d(\delta^{(k+1)} \otimes \lambda^{(k)})(d\mathbf{s}', dt') \\
&= \int_{t' \in I_0 \times \ldots \times I_{k-1}} \left(\int_{\mathbf{s}' \in \{\mathbf{s}_{0:k}\}} p_S^{(k)}(\mathbf{s}', t') d\delta^{(k+1)}(d\mathbf{s}') \right) \lambda^{(k)}(dt') (\text{Fubini}) \\
&= P(\mathbf{s}_0) \prod_{i=1}^{k} P(\mathbf{s}_{i-1}, \mathbf{s}_i) \int_{t' \in I_0 \times \ldots \times I_{k-1}} \prod_{i=1}^{k} p_{T_{\mathbf{s}}}(t'_{i-1}) \lambda^{(k)}(dt') \\
&= P(\mathbf{s}_0) \prod_{i=1}^{k} P(\mathbf{s}_{i-1}, \mathbf{s}_i) \mathbb{P}(T_{\mathbf{s}_{i-1}} \in I_{i-1}) \\
&= \mathbb{P}_S(C)
\end{aligned}$$

We can conclude $\mu_{S|\mathcal{R}} = \mathbb{P}_{S|\mathcal{R}}$. Hence, as they are two measures on $\mathcal{F}(\mathcal{R})$, $\mu_S = \mathbb{P}_S$ by Theorem B.1.1.

Finally, $\frac{d\mathbb{P}_S}{d\overline{\mu}} = p_S$ ($\overline{\mu}$ almost everywhere).

Then, $\forall \sigma = \mathbf{s}_0 \xrightarrow{t_0} \ldots \xrightarrow{t_{k-1}} \mathbf{s}_k, p_S(\sigma) = P(\mathbf{s}_0) \prod_{i=1}^{k} Q(\mathbf{s}_{i-1}, \mathbf{s}_i) e^{-E(\mathbf{s}_{i-1}) t_{i-1}}$.

□

Appendix C

A simple analytical example of ABC inference

Let us illustrate the ABC method and kernel density estimation with a simple example. We consider n i.i.d observations $y_{exp} = (y^{(1)}, \dots, y^{(n)})$ from $\mathcal{N}(\mu, \sigma_0^2)$. We want to estimate $\theta = \mu$ via Approximate Bayesian Computation ($\sigma_0 = 1.0$ is known). The prior is uniform over $[-1, 1]$ ($p(\theta) \propto 1$).

C.1 Computation of the true posterior and ABC posterior

In this case, the different posteriors can be computed analytically, and the mean estimator is a sufficient statistic. We denote $\eta(y_{exp}) = \overline{y_{exp}} = \frac{1}{n}\sum_{i=1}^{n} y^{(i)}$.

First, let us compute the true posterior distribution.

$$\begin{aligned}
p(\theta|y_{exp}) &\propto p(y_{exp}|\theta)p(\theta) \\
&\propto \prod_{i=1}^{n} p(y^{(i)}|\theta) \\
&\propto e^{\frac{1}{2\sigma_0^2}(\sum_{i=1}^{n}(y^{(i)}-\theta)^2)} \\
&\propto e^{\frac{1}{2\sigma_0^2}(\sum_{i=1}^{n} y^{(i)})^2} e^{\frac{n\overline{y_{exp}}^2}{2\sigma_0^2}} \underbrace{e^{\frac{n}{2\sigma_0^2}(\overline{y_{exp}}-\theta)^2}}_{\propto \text{ density of } \mathcal{N}(\overline{y_{exp}}, (\frac{\sigma_0}{\sqrt{n}})^2)}
\end{aligned}$$

So $p(.|y_{exp}) \sim \mathcal{N}(\overline{y_{exp}}, (\frac{\sigma_0}{\sqrt{n}})^2)$ by the factorisation theorem and η is a sufficient statistic. The prior being uniform, $p(.|\theta) \sim \mathcal{N}(\theta, (\frac{\sigma_0}{\sqrt{n}})^2)$.

Let us compute the ABC posterior distribution. ABC can be seen as a regular bayesian method with an approximate likelihood 3.3.2:

$$p^{\epsilon}_{ABC}(y_{exp}|\theta) = \int \mathbb{1}(||y - y_{exp}|| \leq \epsilon)p(y|\theta)dy$$

$$\pi^{\epsilon}_{ABC}(\theta|y_obs) \propto p^{\epsilon}_{ABC}(y_{exp}|\theta)p(\theta)$$

Let us compute the approximate likelihood:

$$\begin{aligned} p^{\epsilon}_{ABC}(y_{exp}|\theta) &= p^{\epsilon}_{ABC}(\overline{y_{exp}}|\theta) \\ &= \int_{\mathbb{R}} \mathbb{1}(||\overline{y} - \overline{y_{exp}}||)p(\overline{y}|\theta)d\overline{y} \\ &= \int_{\overline{y_{exp}}-\epsilon}^{\overline{y_{exp}}+\epsilon} p_{\mathcal{N}(\theta,(\frac{\sigma_0}{\sqrt{n}})^2)}(\overline{y})d\overline{y} \end{aligned}$$

Then $p^{\epsilon}_{ABC}(y_{exp}|\theta) = \mathbb{P}(Y_\theta \leq \overline{y_{exp}}+\epsilon) - \mathbb{P}(Y_\theta \leq \overline{y_{exp}}-\epsilon)$ where $Y_\theta \sim \mathcal{N}(\theta, (\frac{\sigma_0}{\sqrt{n}})^2)$.

Hence, by uniform prior $\pi^{\epsilon}_{ABC}(\theta|y_{exp}) \propto p^{\epsilon}_{ABC}(y_{exp}|\theta)$.

By Fubini and by recognising a law $\mathcal{N}(\theta, (\frac{\sigma_0}{\sqrt{n}})^2)$,

$$\begin{aligned} \int_{\mathbb{R}} p^{\epsilon}_{ABC}(y_{exp}|\theta)d\theta &= \int_{\mathbb{R}} (\int_{\overline{y_{exp}}-\epsilon}^{\overline{y_{exp}}+\epsilon} \frac{\sqrt{n}}{\sqrt{2\pi}} e^{\frac{n}{2\sigma_0^2}(\overline{y}-\theta)^2} d\overline{y})d\theta \\ &= 2\epsilon \end{aligned}$$

In conclusion:

$$\pi^{\epsilon}_{ABC}(\theta|y_{exp}) = \frac{\mathbb{P}(Y_\theta \leq \overline{y_{exp}} + \epsilon) - \mathbb{P}(Y_\theta \leq \overline{y_{exp}} - \epsilon)}{2\epsilon}$$

If $Z \sim \pi^{\epsilon}_{ABC}(.|\overline{y_{exp}})$,

$$\begin{aligned} \mathbb{E}[Z] &= \int_{\mathbb{R}} \theta \pi_{ABC,\epsilon}(.|\overline{y_{exp}})d\theta \\ &= \frac{1}{2\epsilon} \int_{\mathbb{R}} \theta (\int_{\overline{y_{exp}}-\epsilon}^{\overline{y_{exp}}+\epsilon} \frac{\sqrt{n}}{\sqrt{2\pi}} e^{\frac{n}{2\sigma_0^2}(\overline{y}-\theta)^2} d\overline{y})d\theta \\ &= \frac{1}{2\epsilon} \int_{\overline{y_{exp}}-\epsilon}^{\overline{y_{exp}}+\epsilon} (\int_{\mathbb{R}} \theta \frac{\sqrt{n}}{\sqrt{2\pi}} e^{\frac{n}{2\sigma_0^2}(\overline{y}-\theta)^2})d\overline{y} \end{aligned}$$

We recognise the expectation of $\mathcal{N}(\overline{y}, \frac{\sigma_0^2}{n})$, then:

$$\mathbb{E}[Z] = \frac{1}{2\epsilon}\int_{\overline{y_{exp}}-\epsilon}^{\overline{y_{exp}}+\epsilon} \overline{y} d\overline{y} \quad = \frac{1}{4\epsilon}((\overline{y_{exp}}+\epsilon)^2 - (\overline{y_{exp}}-\epsilon)^2) = \overline{y_{exp}}$$

By the same ideas of computations:

$$\begin{aligned}
\mathbb{E}[Z^2] &= \frac{1}{2\epsilon}\int_{\overline{y_{exp}}-\epsilon}^{\overline{y_{exp}}+\epsilon} (\int_{\mathbb{R}} \theta^2 \frac{\sqrt{n}}{\sqrt{2\pi}} e^{\frac{n}{2\sigma_0^2}(\overline{y}-\theta)^2}) d\overline{y} \\
&= \frac{1}{2\epsilon}\int_{\overline{y_{exp}}-\epsilon}^{\overline{y_{exp}}+\epsilon} \overline{y}^2 \frac{\sigma_0^2}{n} d\overline{y} \\
&= \frac{\sigma_0^2}{n} + \frac{1}{6\epsilon}((\overline{y_{exp}}+\epsilon)^3 - (\overline{y_{exp}}-\epsilon)^3) \\
&= \frac{\sigma_0^2}{n} + \overline{y_{exp}}^2 + \frac{\epsilon^2}{3}
\end{aligned}$$

It follows: $\mathbb{V}[Z] = \mathbb{E}[Z^2] - \mathbb{E}[Z]^2 = \frac{\sigma_0^2}{n} + \frac{\epsilon^2}{3} \xrightarrow[\epsilon \to 0]{} \frac{\sigma_0^2}{n}$

In conclusion,

$$\begin{aligned}
\mathbb{E}[Z] &= \overline{y_{exp}} \\
\mathbb{V}[Z] &= \frac{\sigma_0^2}{n} + \frac{\epsilon^2}{3}
\end{aligned}$$

C.2 Simulations

We run the ABC-SMC algorithm with 100, 1000 and 10000 particles.

With $\epsilon = 0.01$.

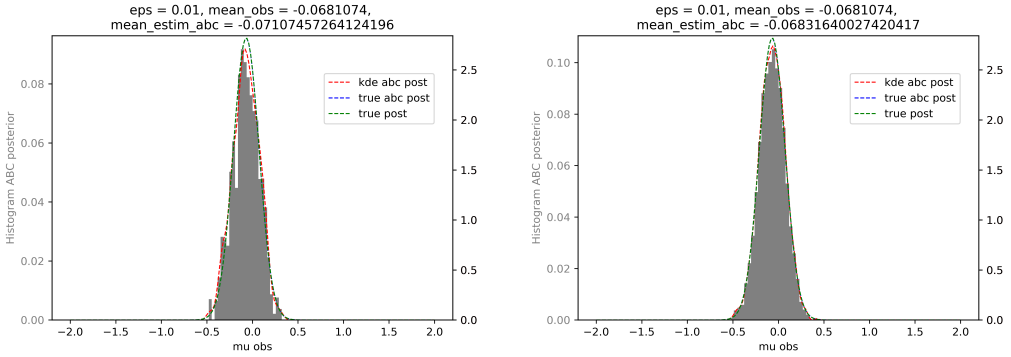


FIGURE C.1: Weighted histograms of ABC. In green the true posterior distribution, in blue the true ABC posterior and in red the estimated ABC posterior with gaussian kernel. On the left: 1000 particles. On the right: 10000 particles.

.

With $\epsilon = 0.1$.

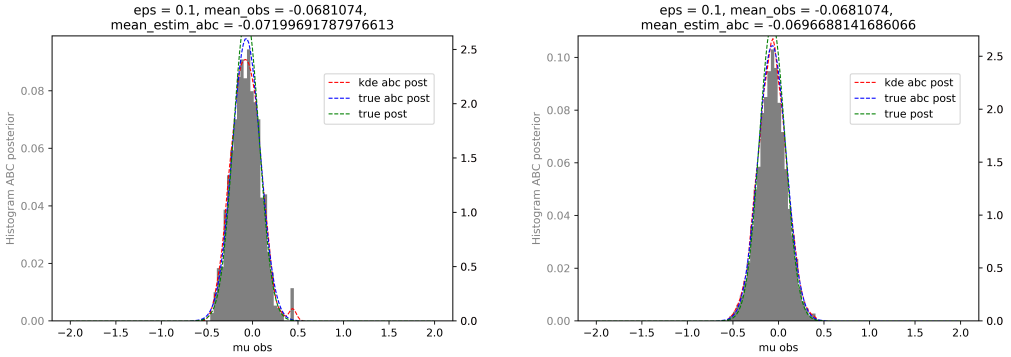


FIGURE C.2: Weighted histograms of ABC run. In green the true posterior distribution, in blue the true ABC posterior and in red the estimated ABC posterior with gaussian kernel. On the left: 1000 particles. On the right: 10000 particles.

Here is a detailed run with $\epsilon = 0.2$ and 1000 particies.

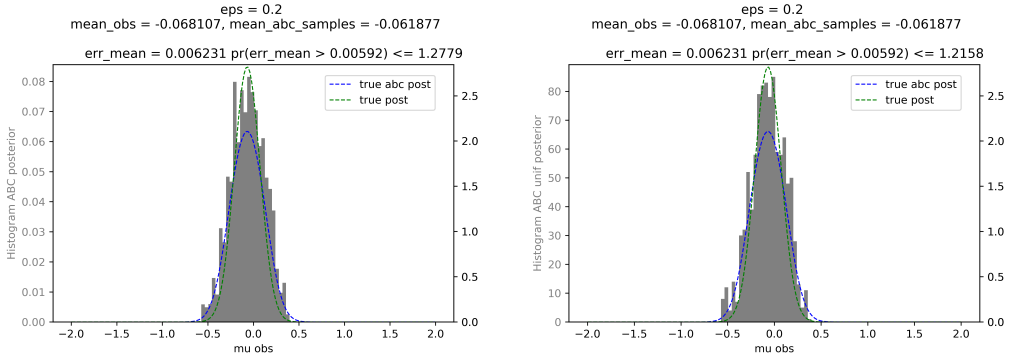


FIGURE C.3: Histogram of ABC run with 1000 particles. In green the true posterior distribution, in blue the true ABC posterior and in red the estimated ABC posterior with gaussian kernel. On the left: multinomial resampling with the weights. On the right: weighted estimator.

.

Appendix D

Synthèse en français

Les réseaux de réactions chimiques (CRN) constituent un formalisme utilisé pour modéliser des processus biologiques. Quand la population est de taille modérée et supposée bien mélangée, le processus stochastique sous-jacent pour décrire ses dynamiques est une chaîne de Markov en temps continu (CTMC). Ce processus est dit sans mémoire: l'état futur du système ne dépend que de l'état courant.

L'inférence statistique de ce type de CTMC est complexe: le calcul de la vraisemblance est en général difficile à résoudre. Les méthodes ABC (Approximate Bayesian Computation) forment une classe de méthodes bayésiennes sans calcul de vraisemblance qui permettent d'approcher la distribution postérieure avec des simulations de Monte-Carlo.

La vérification de modèles, qui fut à l'origine développée pour garantir la fiabilité de systèmes et logiciels informatiques, se penche de plus en plus sur la biologie des systèmes. En effet, il y a un réel besoin de comprendre les interactions complexes entre molécules dans les systèmes biologiques. Malheureusement, l'espace d'états d'un CTMC défini par un CRN explose généralement, voir est infini. Pour palier à cela, des méthodes de vérification statistiques ont été développées. Le principe est de simuler un certain nombre de fois le modèle et de calculer le ratio des simulations qui ont vérifié une propriété. Récemment, une logique temporelle appelée HASL a été introduite pour la vérification statistique de modèles: elle adopte intrinsèquement le point de vue statistique de la vérification.

Dans cette thèse, nous nous intéressons à l'inférence statistique et la vérification statistique de chaines de Markov en temps continu définies par un modèle de réseaux de réactions chimiques. Notre contribution tient principalement dans la formulation d'un algorithme ABC combiné avec le formalisme HASL appelé automaton-ABC.

Les trois premiers chapitres réfèrent à l'état de l'art des différents domaines

études. Le premier chapitre présente les bases la théorie des chaines de Markov. Après avoir discuté des principales définitions et propriétés des CTMC (chaines de Markov continues en temps), nous définissons la mesure de probabilité induit par un CTMC sur son espace de trajectoires. Ensuite, le formalisme des CRN (réseaux de réactions chimiques) est présenté, ainsi que les différentes manières de simuler ces modèles. Le chapitre 2 rappelle les méthodes d'inférence statistique basées sur les simulations de Monte-Carlo. On y présente les bases de l'inférence bayésienne, et de ses algorithmes associés (Importance Sampling, Markov Chain Monte Carlo, Sequential Monte Carlo). Ensuite, une revue des méthodes de type ABC (Approximate Bayesian Computation, famille de méthodes sans calcul de vraissemblance) est détailée. Plusieurs algorithmes d'échantillonnage sont présentés, dont la version séquentielle de Monte Carlo de la méthode ABC (ABC-SMC). Enfin, les méthodes d'estimation des densités par noyaux sont présentées. Le chapitre 3 discute de la vérification de modèles pour les CMTCs. On y présente des logiques temporelles (MITL, CSL), puis on détaille les différents problèmes de vérification (estimation, seuil) pour un CTMC ou une famille paramétrisée de CTMCs. Enfin, le formalisme HASL est présenté, avec la définition des automates linéaires hybrides (LHA), et la simulation synchronisée avec ces automates.

Les deux autres chapitres présentent les contributions. Le chapitre 4 discute de l'inférence statistique pour les CTMCs. On y développe la formulation d'un nouvel algorithme appelé automaton-ABC, qui combine la simulation synchronisée par un automate d'un CTMC permise par le formalisme HASL. Deux applications en sont faites. Premièrement, nous utilisons un automate issu du formalisme HASL qui détecte les comportements oscillatoires pour explorer les zones de paramètres de certains modèles biologiques définis par un CRN qui permettent de produire des oscillations. Deuxièmement, nous utilisons le formalisme HASL pour une implémentation plus efficace de l'inférence ABC classique. Le chapitre 5 développe une nouvelle méthode de vérification statistique basée sur l'algorithme automaton-ABC pour la vérification statistique quantitative pour des problèmes d'atteignabilité bornée en temps. Nous définissons la distance d'une trajectoire issu d'un CTMC à une formule logique MITL, pour ensuite construire des automates (LHA) qui calculent cette distance. A l'aide de ces automates, la méthode automaton-ABC permet de trouver les paramètres des modèles biologiques définis par un CRN qui vont satisfaire certaines formules MITL.

L'implémentation des méthodes présentées est documentée et a conduit au développement d'une bibliothèque dans le langage de programmation Julia. Les détails de l'implémentation sont présents en annexe.

Bibliography

Alharbi, Randa (2018). "Bayesian Inference for Continuous Time Markov Chains". In.

Alur, Rajeev, Tomás Feder, and Thomas A. Henzinger (1991). "The benefits of relaxing punctuality". In: *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing* 43.1, pp. 139–152. DOI: 10.1145/112600.112613.

Anderson, David F. (2007). "A modified next reaction method for simulating chemical systems with time dependent propensities and delays". In: *Journal of Chemical Physics* 127.21. ISSN: 00219606. DOI: 10.1063/1.2799998. arXiv: 0708.0370.

Andrieu, Christophe, Arnaud Doucet, and Roman Holenstein (2010). "Particle Markov chain Monte Carlo methods". In: *Journal of the Royal Statistical Society. Series B: Statistical Methodology* 72.3, pp. 269–342. ISSN: 13697412. DOI: 10.1111/j.1467-9868.2009.00736.x.

Andrieu, Christophe and Gareth O. Roberts (2009). "The pseudo-marginal approach for efficient Monte Carlo computations". In: *Annals of Statistics* 37.2, pp. 697–725. ISSN: 00905364. DOI: 10.1214/07-AOS574.

Aziz, Adnan (2000). "Model Checking Continuous Time Markov Chains". In: *ACM Transactions on Computational Logic*, pp. 162–170.

Aziz, Adnan et al. (1996). "Verifying Continuous Time Markov Chains". In: *LNCS*. Vol. 1102 of LN. ISBN: 978-3-540-61474-6. DOI: 10.1007/3-540-61474-5_75.

Baier, Christel. and Joost-Pieter. Katoen (2008). *Principles of model checking*. MIT Press, p. 975. ISBN: 9780262026499.

Baier, Christel et al. (2003). "Model-Checking Algorithms for Continuous-Time Markov Chains". In: *IEEE Transactions on Software Engineering*, pp. 524–541. DOI: https://doi.org/10.1109/TSE.2003.1205180.

Baldan, Paolo et al. (2010). *Petri nets for modelling metabolic pathways: A survey*. Vol. 9. 4, pp. 955–989. ISBN: 1104701091806. DOI: 10.1007/s11047-010-9180-6.

Ballarini, Paolo and Marie Duflot (2015). "Applications of an expressive statistical model checking approach to the analysis of genetic circuits". In: *Theoretical Computer Science* 599, pp. 4–33. ISSN: 03043975. DOI: 10.1016/j.tcs.2015.05.018. URL: http://dx.doi.org/10.1016/j.tcs.2015.05.018.

Ballarini, Paolo, Radu Mardare, and Ivan Mura (2009). "Analysing Biochemical Oscillation through Probabilistic Model Checking". In: *Electronic Notes in Theoretical Computer Science*. ISSN: 15710661. DOI: 10.1016/j.entcs.2009.02.002.

Ballarini, Paolo et al. (2011). "HASL: An expressive language for statistical verification of stochastic models". In: *VALUETOOLS 2011 - 5th International ICST Conference on Performance Evaluation Methodologies and Tools* May, pp. 306–315. DOI: 10.4108/icst.valuetools.2011.245710.

Ballarini, Paolo et al. (2015). "HASL: A new approach for performance evaluation and model checking from concepts to experimentation". In: *Performance Evaluation* 90, pp. 53–77. ISSN: 01665316. DOI: 10.1016/j.peva.2015.04.003.

Barbot, Benoît (2014). "Acceleration for Statistical Model Checking". In.

Bause, Falko and Pieter Kritzinger (2013). *Stochastic Petri Nets - An Introduction to the Theory*. Vieweg. ISBN: 3-528-15535-3.

Beaumont, Mark A. (2010). "Approximate Bayesian computation in evolution and ecology". In: *Annual Review of Ecology, Evolution, and Systematics* 41, pp. 379–406. ISSN: 1543592X. DOI: 10.1146/annurev-ecolsys-102209-144621.

Beaumont, Mark A. et al. (2009). "Adaptive approximate Bayesian computation". In: *Biometrika* 96.4, pp. 983–990. ISSN: 00063444, 14643510. URL: http://www.jstor.org/stable/27798882.

Bernton, Espen et al. (2019). "Approximate Bayesian computation with the Wasserstein distance". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 81. DOI: 10.1111/rssb.12312.

Bezanson, Jeff et al. (2014). "Julia: A Fresh Approach to Numerical Computing". In: 59.1, pp. 65–98. ISSN: 0036-1445. DOI: 10.1137/141000671. arXiv: 1411.1607. URL: http://arxiv.org/abs/1411.1607.

Billingsley, Patrick (1961). "Statistical Inference for Markov Processes". In: *The University of Chicago*. ISSN: 15372723. DOI: 10.1080/00401706.1963.10490116.

Bladt, Mogens and Bo Friis Nielsen (2017). *Matrix-Exponential Distributions in Applied Probability*. Vol. 81. ISBN: 978-1-4939-7047-6. DOI: 10.1007/978-1-4939-7049-0.

Bladt, Mogens and Michael Sørensen (2005). "Statistical inference for discretely observed Markov jump processes". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.3, pp. 395–410. DOI: https://doi.org/10.1111/j.1467-9868.2005.00508.x. URL: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2005.00508.x.

Blum, Michael G.B. B. et al. (2012). "A comparative review of dimension reduction methods in approximate Bayesian computation". In: *Statistical Science* 28.2, pp. 189–208. DOI: 10.1214/12-STS406. arXiv: 1202.3819.

Bortolussi, Luca, Dimitrios Milios, and Guido Sanguinetti (2016). "Smoothed model checking for uncertain Continuous-Time Markov Chains". In: *Information and Computation* 247, pp. 235–253. ISSN: 0890-5401. DOI: 10.1016/J.IC.2016.01.004. URL: https://www.sciencedirect.com/science/article/pii/S0890540116000055.

Bortolussi, Luca and Simone Silvetti (2018). "Bayesian statistical parameter synthesis for linear temporal properties of stochastic models". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10806 LNCS. Springer Verlag, pp. 396–413. ISBN: 9783319899626. DOI: 10.1007/978-3-319-89963-3_23.

Bouezmarni, Taoufik and Jeroen V.K. Rombouts (2010). "Nonparametric density estimation for multivariate bounded data". In: *Journal of Statistical Planning and Inference* 140.1, pp. 139–152. ISSN: 03783758. DOI: 10.1016/j.jspi.2009.07.013.

Brent, Richard (1971). "An Algorithm with Guaranteed Convergence for Finding a Zero of a Function." In: *Comput. J.* 14, pp. 422–425. DOI: 10.1093/comjnl/14.4.422.

Brim, Luboš, Milaň Ceška, and Sven Dražan (2013). *Exploring Parameter Space of Stochastic Biochemical Systems Using Quantitative Model Checking*. Tech. rep.

Brooks, Steve et al. (2011). *Handbook of Markov Chain Monte Carlo*. DOI: 10.1201/b10905.

Cao, Yang, Daniel T. Gillespie, and Linda R. Petzold (2005). "Avoiding negative populations in explicit Poisson tau-leaping". In: *Journal of Chemical Physics* 123.5. ISSN: 00219606. DOI: 10.1063/1.1992473.

— (2006). "Efficient step size selection for the tau-leaping simulation method". In: *Journal of Chemical Physics* 124.4. ISSN: 00219606. DOI: 10.1063/1.2159468.

Cappé, Olivier, Eric Moulines, and Tobias Rydén (2009). *Inference in Hidden Markov Models*. Tech. rep.

Casella, George and Roger Berger (2001). *Statistical Inference*. Duxbury Resource Center. ISBN: 0534243126.

Ceška, Milaň et al. (2014). *Precise Parameter Synthesis for Stochastic Biochemical Systems*. Tech. rep.

Chacón, José and Tarn Duong (2018). *Multivariate Kernel Smoothing and its Applications*. ISBN: 9780429485572. DOI: 10.1201/9780429485572.

Chen, Song (2000). "Probability Density Function Estimation Using Gamma Kernels". In: *Annals of the Institute of Statistical Mathematics* 52, pp. 471–480. DOI: 10.1023/A:1004165218295.

Chen, Song Xi (1999). *Beta kernel estimators for density functions*. Tech. rep., pp. 131–145. URL: www.elsevier.com/locate/csda.

Craciun, Gheorghe et al. (2013). "Statistical model for biochemical network inference". In: *Communications in Statistics: Simulation and Computation* 42.1, pp. 121–137. ISSN: 03610918. DOI: 10.1080/03610918.2011.633200.

Del Moral, Pierre, Arnaud Doucet, and Ajay Jasra (2006). "Sequential Monte Carlo samplers". In: *Journal of the Royal Statistical Society B* 68.3, pp. 411–436. ISSN: 1369-7412. DOI: 10.1111/j.1467-9868.2006.00553.x.

— (2012a). "An adaptive sequential Monte Carlo method for approximate Bayesian computation". In: *Statistics and Computing* 22.5, pp. 1009–1020. ISSN: 09603174. DOI: 10.1007/s11222-011-9271-y.

— (2012b). "On adaptive resampling strategies for sequential Monte Carlo methods". In: *Bernoulli* 18.1, pp. 252–278. ISSN: 13507265. DOI: 10.3150/10-BEJ335.

Djafri, Hilal (2012). "Numerical and statistical approaches for model checking of stochastic processes". In.

Doucet, Arnaud, Nando De Freitas, and Neil Gordon (2001). *Sequential Monte Carlo Methods in Practice*. Ed. by Arnaud Doucet, Nando De Freitas, and Neil Gordon. Springer, New York, NY. DOI: https://doi.org/10.1007/978-1-4757-3437-9.

Doucet, Arnaud and A M Johansen (2009). "A tutorial on particle filtering and smoothing: Fifteen years later". In: *Handbook of Nonlinear Filtering* December. Ed. by D Crisan and B Rozovsky, pp. 4–6. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.157.772{\&}rep=rep1{\&}type=pdf.

Duong, Tarn and Martin Hazelton (2005). "Cross-validation Bandwidth Matrices for Multivariate Kernel Density Estimation". In: *Scandinavian Journal of Statistics* 32, pp. 485–506. DOI: 10.1111/j.1467-9469.2005.00445.x.

Duong, Tarn et al. (2020). *ks: Kernel Smoothing. R package version 1.11.7*. ISBN: 9780429485572. DOI: 10.1201/9780429485572>.License. URL: https://cran.r-project.org/package=ks.

Elowitz, Michael B., Stanislas Leibler, and Stanislas Leibier (2000). "A synthetic oscillatory network of transcriptional regulators". In: *Nature* 403.6767, pp. 335–338. ISSN: 00280836. DOI: 10.1038/35002125. arXiv: NIHMS150003. URL: http://www.nature.com/nature/journal/v403/n6767/full/403335a0.html.

Eungdamrong, Narat and Ravi Iyengar (2004). "Modeling Cell Signaling Networks". In: *Biology of the cell / under the auspices of the European Cell Biology Organization* 96, pp. 355–362. DOI: 10.1016/j.biolcel.2004.03.004.

Fan, Y. and S. A. Sisson (2018). "ABC Samplers". In: pp. 1–46. arXiv: 1802.09650. URL: http://arxiv.org/abs/1802.09650.

Fasiolo, Matteo and Simon N. Wood (2015). "Approximate methods for dynamic ecological models". In: pp. 1–22. arXiv: 1511.02644. URL: http://arxiv.org/abs/1511.02644.

Feinberg, Martin (2019). *Foundations of Chemical Reaction Network Theory*. ISBN: 978-3-030-03857-1. DOI: 10.1007/978-3-030-03858-8.

Filippi, Sarah et al. (2011). "On optimality of kernels for approximate Bayesian computation using sequential Monte Carlo". In: arXiv: 1106.6280. URL: http://arxiv.org/abs/1106.6280.

Frazier, David T. et al. (2016). "Asymptotic Properties of Approximate Bayesian Computation". In: arXiv: 1607.06903. URL: http://arxiv.org/abs/1607.06903.

Geman, Stuart and Donald Geman (1984). "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6.6, pp. 721–741. ISSN: 01628828. DOI: 10.1109/TPAMI.1984.4767596.

Gibson, Michael A. and Jehoshua Bruck (2000). "Efficient exact stochastic simulation of chemical systems with many species and many channels". In: *Journal of Physical Chemistry A* 104.9, pp. 1876–1889. ISSN: 10895639. DOI: 10.1021/jp993732q.

Gilks, Walter, Sylvia Richardson, and D J E Spiegelhalter (1996). *Markov Chain Monte Carlo In Practice*. Vol. xvii. DOI: 10.1007/978-1-4899-4485-6_1.

Gillespie, Daniel (1977). "Exact Stochastic Simulation Of Coupled Chemical-Reactions". In: *J. of Physical Chemistry* 81, pp. 2340–2361. DOI: 10.1021/j100540a008.

— (2000). "The Chemical Langevin Equation". In: *Journal of Chemical Physics* 115, pp. 297–306. DOI: 10.1063/1.481811.

— (2007). "Stochastic Simulation of Chemical Kinetics". In: *Annual review of physical chemistry* 58, pp. 35–55. DOI: 10.1146/annurev.physchem.58.032806.104637.

Goffe, William, Gary Ferrier, and John Rogers (1994). "Global Optimization of Statistical Functions with Simulated Annealing". In: *Journal of Econometrics* 60, pp. 65–99. DOI: 10.1016/0304-4076(94)90038-8.

Gómez-Corral, A. et al. (2015). "Bayesian Inference of Markov Processes". In: *Wiley StatsRef: Statistics Reference Online*, pp. 1–15. DOI: 10.1002/9781118445112.stat07837.

Gordon, N. J., D. J. Salmond, and A. F.M. Smith (1993). "Novel approach to nonlinear/non-gaussian Bayesian state estimation". In: *IEE Proceedings, Part F: Radar and Signal Processing* 140.2, pp. 107–113. ISSN: 0956375X. DOI: 10.1049/ip-f-2.1993.0015.

Han, Tingting, Joost Pieter Katoen, and Alexandru Mereacre (2008). "Approximate parameter synthesis for probabilistic time-bounded reachability". In: *Proceedings - Real-Time Systems Symposium*, pp. 173–182. ISBN: 9780769534770. DOI: 10.1109/RTSS.2008.19.

Haseltine, Eric L. and James B. Rawlings (2002). "Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics". In: *Journal of Chemical Physics* 117.15, pp. 6959–6969. ISSN: 00219606. DOI: 10.1063/1.1505860.

Hastings, W (1970). "Monte Carlo Sampling Methods Using Markov Chains and Their Application". In: *Biometrika* 57. DOI: 10.1093/biomet/57.1.97.

Jasra, Ajay (2015). "Approximate Bayesian computation for a class of time series models". In: *International Statistical Review* 83.3, pp. 405–435. ISSN: 17515823. DOI: 10.1111/insr.12089. arXiv: 1401.0265.

Jasra, Ajay et al. (2019). "Multilevel Monte Carlo in approximate Bayesian computation". In: *Stochastic Analysis and Applications* 37.3, pp. 346–360. ISSN: 15329356. DOI: 10.1080/07362994.2019.1566006. arXiv: 1702.03628. URL: https://doi.org/10.1080/07362994.2019.1566006.

Jegourel, Cyrille, Jun Sun, and Jin Song Dong (2019). "Sequential schemes for frequentist estimation of properties in statistical model checking". In: *ACM Transactions on Modeling and Computer Simulation* 29.4. ISSN: 15581195. DOI: 10.1145/3310226.

JuliaLang (2020). *Performance tips of Julia language*. URL: https://docs.julialang.org/en/v1/manual/performance-tips/.

Karlebach, Guy and Ron Shamir (2008). "Modelling and analysis of gene regulatory networks". In: *Nature Reviews Molecular Cell Biology* 9.10, pp. 770–780. ISSN: 14710072. DOI: 10.1038/nrm2503.

Kermack, W. O. 0 and A. G. McKendrick (1927). "A Contribution to the Mathematical Theory of Epidemics". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 115.772, pp. 700–721. ISSN: 1364-5021. DOI: 10.1098/rspa.1927.0118. arXiv: 1301.2791. URL: http://rspa.royalsocietypublishing.org/http://rspa.royalsocietypublishing.org/cgi/doi/10.1098/rspa.1927.0118.

Klenke, Achim (2008). *Probability Theory*. DOI: 10.1007/3-540-33414-9.

Kong, Augustine and Jun S. Liu (1994). "Sequential imputations and Bayesian missing data problems". In: *Journal of the American Statistical Association* 89.425, pp. 278–288. ISSN: 1537274X. DOI: 10.1080/01621459.1994.10476469.

Koutroumpas, Konstantinos et al. (2016). "Bayesian parameter estimation for the Wnt pathway: An infinite mixture models approach". In: *Bioinformatics*. Bioinformatics. DOI: 10.1093/bioinformatics/btw471.

Kulkarni, Vidyadhar (1998). "Modeling and Analysis of Stochastic Systems". In: *Journal of the American Statistical Association* 93. DOI: 10.2307/2669884.

Kurtz, Thomas G (1980). *Representations of Markov Processes as Multiparameter Time Changes*. DOI: 10.1214/aop/1176994660.

Kwiatkowska, M, G Norman, and D Parker (2011). "{PRISM} 4.0: Verification of Probabilistic Real-time Systems". In: *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*. Ed. by G Gopalakrishnan and S Qadeer. Vol. 6806. LNCS. Springer, pp. 585–591.

Kwiatkowska, Marta, Gethin Norman, and David Parker (2008). "Using Probabilistic Model Checking in Systems Biology". In: *SIGMETRICS Perform. Eval. Rev.* 35.4, pp. 14–21. ISSN: 0163-5999. DOI: 10.1145/1364644.1364651. URL: https://doi-org.ezproxy.universite-paris-saclay.fr/10.1145/1364644.1364651.

Legay, Axel, Benoît Delahaye, and Saddek Bensalem (2010). "Statistical model checking: An overview". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6418 LNCS, pp. 122–135. ISSN: 03029743. DOI: 10.1007/978-3-642-16612-9_11. arXiv: 1005.1327.

Legay, Axel et al. (2019). "Statistical Model Checking". In: *Computing and Software Science: State of the Art and Perspectives*. Ed. by Bernhard Steffen and Gerhard Woeginger. Cham: Springer International Publishing, pp. 478–504. ISBN: 978-3-319-91908-9. DOI: 10.1007/978-3-319-91908-9_23. URL: https://doi.org/10.1007/978-3-319-91908-9{_}23.

Leloup, J.-C. and A. Goldbeter (2003). "Toward a detailed computational model for the mammalian circadian clock". In: *Proceedings of the National Academy of Sciences* 100.12, pp. 7051–7056. ISSN: 0027-8424. DOI: 10.1073/pnas.1132112100. URL: http://www.pnas.org/cgi/doi/10.1073/pnas.1132112100.

Lenive, Oleg, Paul D.W. Kirk, and Michael P.H. Stumpf (2016). "Inferring extrinsic noise from single-cell gene expression data using approximate Bayesian computation". In: *BMC Systems Biology*. ISSN: 17520509. DOI: 10.1186/s12918-016-0324-x.

Lester, Christopher (2018). "Multi-level Approximate Bayesian Computation". In: arXiv: 1811.08866. URL: http://arxiv.org/abs/1811.08866.

Li, Wentao and Paul Fearnhead (2017). "On the Asymptotic Efficiency of Approximate Bayesian Computation Estimators". In: *Biometrika* 105. DOI: 10.1093/biomet/asx078.

Li, Zhengda and Qiong Yang (2018). "Systems and synthetic biology approaches in understanding biological oscillators". In: *Quantitative Biology* 6.1, pp. 1–14. ISSN: 20954697. DOI: 10.1007/s40484-017-0120-7.

Liu, Jun S (2008). *Monte Carlo Strategies in Scientific Computing*. Corrected. ISBN: 0387952306,9780387952307. URL: http://gen.lib.rus.ec/book/index.php?md5=fc8bb07ef2ffed66509ec1bb3c9706e0.

Liu, Jun S. and Rong Chen (1998). "Sequential monte carlo methods for dynamic systems". In: *Journal of the American Statistical Association* 93.443, pp. 1032–1044. ISSN: 1537274X. DOI: 10.1080/01621459.1998.10473765.

Loskot, Pavel, Komlan Atitey, and Lyudmila Mihaylova (2019). "Comprehensive review of models and methods for inferences in bio-chemical reaction networks". In: *Frontiers in Genetics* 10.JUN. ISSN: 16648021. DOI: 10.3389/fgene.2019.00549. arXiv: 1902.05828.

Lotka, Aj (1932). "The growth of mixed populations: Two species competing for a common food supply". In: *J. Wash. Acad. Sci.* 22.

Marin, Jean-Michel et al. (2011). "Approximate Bayesian Computational methods". In: 1, pp. 1167–1180. ISSN: 0960-3174. DOI: 10.1007/s11222-011-9288-2. arXiv: 1101.0955. URL: http://arxiv.org/abs/1101.0955.

Marjoram, Paul et al. (2003). "Markov chain Monte Carlo without likelihoods". In: *Proceedings of the National Academy of Sciences* 100.26, pp. 15324–15328. ISSN: 0027-8424. DOI: 10.1073/pnas.0306899100. URL: https://www.pnas.org/content/100/26/15324.

Metropolis et al. (1953). "Equation of state calculations for fast computing machines". In: *Journal of Chemical Physics 6* 21, pp. 1087–.

Michaelis, Leonor et al. (2011). "The Original Michaelis Constant: Translation of the 1913 Michaelis-Menten Paper". In: *Biochemistry* 50, pp. 8264–8269. DOI: 10.1021/bi201284u.

Minka, Thomas P. (2013). "Expectation Propagation for approximate Bayesian inference". In: pp. 362–369. arXiv: 1301.2294. URL: http://arxiv.org/abs/1301.2294.

Molyneux, Gareth W. and Alessandro Abate (2020). "ABC (SMC) 2 : Simultaneous Inference and Model Checking of Chemical". In: *Computational Methods in Systems Biology, 18th International Conference, CMSB 2020*, pp. 255–279. URL: http://dx.doi.org/10.1007/978-3-030-60327-4{_}14.

Nadjahi, Kimia et al. (2020). "Approximate Bayesian Computation with the Sliced-Wasserstein Distance". In: pp. 5470–5474. DOI: 10.1109/ICASSP40776.2020.9054735.

Nicholas Metropolis (1987). "The Beginning of the Monte Carlo Method". In: *Los Alamos Science* 15, pp. 125–130.

O'Brien, Erin L., Elizabeth Van Itallie, and Matthew R. Bennett (2012). *Modeling synthetic gene oscillators*. DOI: 10.1016/j.mbs.2012.01.001.

Perkins, Theodore J (2017). *Maximum likelihood trajectories for continuous-time Markov chains*. Tech. rep.

Pfeuffer, Marius (2017). "Ctmcd: An R package for estimating the parameters of a continuous-time Markov chain from discrete-time data". In: *R Journal* 9.2, pp. 127–141. ISSN: 20734859. DOI: 10.32614/rj-2017-038.

Picchini, Umberto and Adeline Samson (2018). "Coupling stochastic EM and approximate Bayesian computation for parameter inference in state-space models". In: *Computational Statistics* 33.1, pp. 179–212. ISSN: 16139658. DOI: 10.1007/s00180-017-0770-y. arXiv: 1512.04831.

Prangle, Dennis (2015). "Summary Statistics in Approximate Bayesian Computation". In: ISSN: 0031-5125. DOI: 10.2466/06.30.PMS.120v19x9. arXiv: 1512.05633.

Pritchard, Jonathan K et al. (1999). "Population Growth of Human Y Chromosomes: A Study of Y Chromosome Microsatellites". In: *Mol. Biol. Evol* 16.12, pp. 1791–1798. ISSN: 0737-4038.

Purcell, Oliver et al. (2010). "A comparative analysis of synthetic genetic oscillators". In: *Journal of the Royal Society Interface* 7.52, pp. 1503–1524. ISSN: 17425662. DOI: 10.1098/rsif.2010.0183.

Rathinam, Muruhan et al. (2003). "Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method". In: *Journal of Chemical Physics* 119.24, pp. 12784–12794. ISSN: 00219606. DOI: 10.1063/1.1627296.

Ratmann, Oliver et al. (2007). "Using likelihood-free inference to compare evolutionary dynamics of the protein networks of H. pylori and P. falciparum". In: *PLoS Computational Biology*. ISSN: 1553734X. DOI: 10.1371/journal.pcbi.0030230.

Robert, Christian P. (2019). *Asymptotics of ABC*. URL: https://fr.slideshare.net/xianblog/asymptotics-of-abc.

Robert, Christian P. and George Casella (2004). *Monte Carlo Statistical Methods*. ISBN: 978-1-4419-1939-7. DOI: 10.2307/1270959. arXiv: arXiv:1011.1669v3.

Rosenblatt, M (1956). "Remarks on Some Nonparametric Estimate of a Density Function". In: *The Annals of Mathematical Statistics* 27, pp. 832–835.

Schnoerr, David, Guido Sanguinetti, and Ramon Grima (2017). "Approximation and inference methods for stochastic biochemical kinetics - A tutorial review". In: *Journal of Physics A: Mathematical and Theoretical* 50.9. ISSN: 17518121. DOI: 10.1088/1751-8121/aa54d9. arXiv: 1608.06582.

Silverman, B W (1986). *Density Estimation for Statistics and Data Analysis*. London: Chapman & Hall.

Sisson, S. A., Y. Fan, and M. A. Beaumont (2018). "Overview of Approximate Bayesian Computation". In: 1, pp. 1–66. arXiv: 1802.09720. URL: http://arxiv.org/abs/1802.09720.

Sisson, S. A., Y. Fan, and Mark M. Tanaka (2007). "Sequential Monte Carlo without likelihoods". In: *Proceedings of the National Academy of Sciences of the United*

States of America 104.6, pp. 1760–1765. ISSN: 00278424. DOI: 10.1073/pnas.0607208104.

Sisson, Scott A., Yanan Fan, and Mark A. Beaumont (2019). *Handbook of Approximate Bayesian Computation*. Ed. by Scott A. Sisson, Yanan Fan, and Mark A. Beaumont. Chapman and Hall/CRC. ISBN: 9780367733728.

Spieler, David (2014). "Numerical Analysis of Long-Run Properties for Markov Population Model". In.

Stroock, Daniel (2005). "An Introduction to Markov Processes". In: 230. DOI: 10.1007/b138428.

Tavaré, Simon et al. (1997). "Inferring coalescence times from DNA sequence data". In: *Genetics* 145.2, pp. 505–518. ISSN: 00166731. DOI: 10.1093/genetics/145.2.505.

Thattai, M. and A. Van Oudenaarden (2001). "Intrinsic noise in gene regulatory networks". In: *Proceedings of the National Academy of Sciences of the United States of America* 98.15, pp. 8614–8619. ISSN: 00278424. DOI: 10.1073/pnas.151588598.

Toni, Tina et al. (2009). "Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems". In: *Journal of the Royal Society Interface* 6.31, pp. 187–202. ISSN: 17425662. DOI: 10.1098/rsif.2008.0172. arXiv: 0901.1925.

Tyson, John J. et al. (2008). "Biological switches and clocks". In: *Journal of the Royal Society Interface* 5.SUPPL. 1. ISSN: 17425662. DOI: 10.1098/rsif.2008.0179.focus.

Warne, David J., Ruth E. Baker, and Matthew J. Simpson (2019). "Simulation and inference algorithms for stochastic biochemical reaction networks: From basic concepts to state-of-the-art". In: *Journal of the Royal Society Interface* 16.151. ISSN: 17425662. DOI: 10.1098/rsif.2018.0943. arXiv: 1812.05759.

université PARIS-SACLAY | ÉCOLE DOCTORALE Interfaces: matériaux, systèmes, usages

Titre: Inférence et vérification statistiques de réseaux de réactions chimiques.

Mots clés: Réseaux de réactions chimiques, inférence, vérification, ABC, HASL, Julia

Résumé: Les réseaux de réactions chimiques (CRN) constituent un formalisme utilisé pour modéliser des processus biologiques. Sous certaines hypothèses, le processus stochastique sous-jacent pour décrire ses dynamiques est une chaîne de Markov en temps continu (CTMC): l'état futur du système ne dépend que de l'état courant. L'inférence statistique de ce type de CTMC est complexe: le calcul de la vraisemblance est en général difficile à résoudre. Les méthodes ABC (Approximate Bayesian Computation) forment une classe de méthodes bayésiennes sans calcul de vraisemblance qui permettent d'approcher la distribution postérieure avec des simulations de Monte-Carlo. La vérification de modèles se penche de plus en plus sur la biologie des systèmes. En effet, il y a un réel besoin de comprendre les interactions complexes entre molécules dans les systèmes biologiques. Des méthodes de vérification statistiques ont été développées pour appliquer ces méthodes dans des modèles plus complèxes. Le principe est de simuler un certain nombre de fois le modèle et de calculer le ratio des simulations qui ont vérifié une propriété. Récemment, une logique temporelle appelée HASL a été introduite pour la vérification statistique de modèles: elle adopte intrinsèquement le point de vue statistique de la vérification. Dans cette thèse, nous nous intéressons à l'inférence statistique et la vérification statistique de chaines de Markov en temps continu définies par un modèle de réseaux de réactions chimiques. Notre contribution tient principalement dans la formulation d'un algorithme ABC combiné avec le formalisme HASL appelé automaton-ABC. Nous appliquons cette méthode haut niveau sur plusieurs tâches d'inférence statistique et de vérification pour des CTMCs issus de systèmes biologiques, impliquant notamment des modèles oscillatoires et des problèmes d'atteignabilité bornés en temps. L'implémentation des méthodes présentées est rendue disponible sous la forme d'une bibliothèque dans le langage de programmation Julia.

Title: Statistical inference and verification of Chemical Reaction Networks

Keywords: Chemical Reaction Networks, inference, model checking, ABC, HASL, Julia

Abstract: Chemical Reaction Networks (CRN) constitute a formalism used to model biological processes. Under certain assumptions, a Continuous-Time Markov Chain describes its stochastic dynamics: the future state of the system only depends on the current state. Statistical inference of such CTMCs is complex: likelihood computations are generally intractable. Approximate Bayesian Computation is a recent class of likelihood-free methods for Bayesian inference that allows approximating the posterior distribution with Monte Carlo simulations. It has proven its efficiency in the case of CTMCs. There is a growing interest in the verification (model-checking) of models from Systems Biology to understand the complex molecular interactions within a biological system. Statistical model checking methods have been developed to apply these methods on more complex models. They simulate the model and compute the ratio of simulations that fulfils a property. Recently, Hybrid Automata Stochastic Logic (HASL) has been introduced for the statistical verification of stochastic models. This temporal logic inherently adopts the statistical point of view of model-checking. In this thesis, we focus on statistical inference and verification of CTMCs defined by CRNs. Our main contribution consists in the new formulation of an Approximate Bayesian Computation procedure combined with HASL called automaton-ABC. We apply this high-level method on several tasks of statistical inference and verification for biological CTMCs, including oscillatory models and time-bounded reachability problems. The implementation of our algorithms has led to a package in the Julia Programming language.