Contributions to formal modelling and analysis of stochastic models Mémoire d'habilitation à diriger des recherches

Paolo Ballarini

Contributions to formal modelling and analysis of stochastic models

Mémoire d'habilitation à diriger des recherches

Paolo Ballarini

Defended on June 6th 2023, before the committee composed of:

Eugenio Cinquemani, INRIA Grenoble Alpes	*reviewer*
Alain Denise, Université Paris Saclay	*president*
Giuliana Franceschinis, Università Piemonte Orientale	*examiner*
Stefan Haar, École Normale Supérieure Saclay	*examiner*
Dave Parker, University of Oxford	*reviewer*
Verena Wolf, Universität des Saarlandes	*reviewer*

Contributions to formal modelling and analysis of stochastic models

Mémoire d'habilitation à diriger des recherches

Paolo Ballarini

Abstract

This manuscript describes a number of contributions, I had the opportunity to work on, in the field of formal methods for stochastic models. Research in formal methods is concerned with defining mathematically rigorous formalisms that allows for specification, development and verification of the correctness of complex systems. In the case of stochastic systems modelling formalisms need to account for stochasticity and verification algorithms needs to address also *quantitative* properties other than *qualitative* ones.

The first contribution included in the manuscrit overviews a statistical model checking framework, named Hybrid Automata Specification Language (HASL), which targets a general class of stochastic processes and that, by employing hybrid automata as *machineries* to specify the behavior to be analysed, leads to a powerful property specification language which surpasses the expressiveness of classical stochastic temporal logic languages such as CSL.

The second contribution is concerned with the *parametrised* extension of the stochastic model checking problem, namely the realisation of a procedure to estimate the (probabilistic) dependency of the satisfaction of a property ϕ w.r.t. the parameters of a stochastic model. By introducing a *satisfiability distance* measure that quantifies how distant a property ϕ is from being satisfied by a model's instance and relying on the expressive of HASL property language, we adapted classical Approximated Bayesian Computation (ABC) schemes so to assess the *satisfaction probabilistic distribution* of ϕ against the parameter space Θ of the considered model.

The third contribution is about the application of model checking approaches to the analysis of stochastic oscillators, i.e. a prominent class of systems in systems biology. Classical temporal logic languages have proven limited success in detecting/assessing oscillations, whereas effective procedures can be obtained through automata-based model checking. The contribution we present is twofold: on one hand we show how sophisticated oscillation-related indicators can be assessed through HASL based model checking and on the other hand we show how to assess the dependency of such indicators (period and amplitude) against the parameters of a stochastic oscillator.

As a final contribution we illustrate a framework for building stochastic models that adequately reproduce periodic phenomenon from a dataset, with data about visits at a hospital Emergency Department (ED) as the reference case study. We considered different

families of stochastic processes, Markov renewal processes, Markov arrival processes (MAP) and hidden Markov models (HMM) and through fit them so that they match meaningful statistical indicators computed on the dataset.

Contents

Chapter 1

Introduction

Modern engineering is a wide, pervasive and multi-disciplinary field which revolves around the notion of *system*, where a system may be thought of as aggregation of interacting parts (i.e. objects, things, components) which are combined by nature or humans to form a *whole* that performs a certain *function*. Examples of systems can be found in just about any possible field. They may be distinguished according with their kind, as in, for example, *natural systems* (e.g. the system of biochemicals reactions which controls the Eukaryotic cell's reproduction cycle), or *human-made systems*, (e.g. embedded systems consisting of interacting electronic devices that allow an aircraft to be safely flown, or smart devices, as in modern *Internet of Things* context allowing remote control for home utilities), or *abstract systems* (e.g. an ecosystem of beings interacting with each other for preserving their species). The development of model-based methodologies to support the design, validation, performance analysis, testing and maintenance is fundamental in system engineering. A model is a mathematical, abstract representation of a system, that, depending on its nature, may serve diverse purposes the main of which is to reliably mimic the system's dynamics, while possibly allowing for performance analysis, and/or verification of requirements. As our society relies more and more on complex software controlled critical systems the need to employ formal methods to design, validate and certify them becomes paramount.

Model-based methods

My work is mainly focused on model-based methods, namely, the area of research whose goal is the development of formal modelling frameworks which allows one to unambiguously build a model of a real-life system and, based on rigorous mathematical foundations, assess the model's properties. The overall goal is to establish whether the modelled system functions correctly, which, depending on the nature of the model, may entail the verification of some *qualitative* requirement and/or the evaluation of some *quantitaive* performance indicator.

Model-checking. Amongst model-based methods *model checking*, quickly became a prominent one as it gives the modeller the possibility to automatically verify a system's requirements, formally expressed as temporal logic properties, against a system's model. The seminal work of Pnueli [Pnu77] on the verification of sequential and concurrent programs set the ground for the formalisation of algorithms for model-checking of non-probabilistic systems introduced by Emerson and Clarke [EC80] and also, independently, by Quelles and Sifakis [QS82] (resulting in the 2007 Turing award awarded to the authors) and eventually leading to many extensions targeting different classes of models including probabilistic ones [ASSB96, ASSB00, BHHK03a]. The attractiveness of model checking is mainly due to its practical simplicity as, while keeping the conception of the model separated from that of the requirements (i.e. *separation of concerns*), it provides the user with a *push-button* machinery for certifying that the system is correct (i.e. if the model *satisfies* the requirements).

Classes of models. The variety of real-life systems calls for models to account for different features including timing contraints, probabilities and unknown parameters.

Timing constraints are crucial for safety-critical systems, examples of which can be found in domains such as aircraft controlling, nuclear power plants and the automotive industry, to mention a few. Timed automata [AD94] and timed Petri, for instance, are formalisms suitable to account for time dependent behavior and where timing requirements are expressed through dense-time variables that evolve all at the same speed and that can be used to capture either the delay since the last occurrence of an event or how long a given transition has been enabled for.

Other kind of systems instead are characterised by a *probabilistic behaviour*, that is, a behaviour which can not be *certainly* predicted in advance. We may distinguish between systems that are *probabilistic by design* as, for example, WiFi protocols which employ probabilistic algorithms to reduce *packet collisions*, as opposed to systems that are *probabilistic by nature*, as the dynamics of the real-life phenomenons they depend upon can only be expressed through some probability law, like for example for failure rates of physical components.

Finally most often models depend on a number of parameters whose value is not necessarily known *a priori*. For example, for a model of queuing system the maximum number of clients standing in a queue for a given service might not necessarily be known, or, for a model of a manufacturing system the probability that a given workstation breaks down in the next time unit may not be known. Whenever some parameters are unknown the model becomes *parametric* and any analysis technique also needs to be adapted to account for the effect that parameters have on the behavior. In model checking terms this entails adapting verification algorithms so to assess the effect that parameters have on the satisfaction of the model's requirements.

Trading off expressiveness with tractability. The realm of model-based techniques is

affected by two opposing forces *i)* the *expressiveness* of the modelling framework and of the corresponding analysis techniques and *ii)* the tractability of the models. These two aspects are *in opposition* as, generally speaking, the more expressive the modelling framework, the more complex and computationally costly are the mathematical tools necessary to analyse its behaviour.

In model checking terms, for example, such an antagonism entails considering to what extent the relevant characteristic of a system's behaviour can be accounted for by the modeller through a formally expressed property. This depends both on the kind of model (e.g. transition-system, timed-automaton, Markov-chain) and of the mathematical means on top of which an algorithm for verifying the formulae of the corresponding property language can be defined.

Contents of this thesis

Over the last decade I have been focusing on research in the area of complex systems taking into account several amongst the above described aspects. My contributions are twofold: *methodological/theoretical* as well as *applicative*. From a methodological perspective the core of my work is in the area of probabilistic systems and revolves around the probabilistic model checking problem, that is, looking at how to improve on one hand the scalability and on the other the expressiveness of approaches for the verification of probabilistic systems. Although the probabilistic model checking has been at the core of my interests, progressively my activities diversified towards other areas, including that of parameter inference as well as that of introducing novel timed-automata based formalisms. From an applicative perspective the focus of my work has been mainly in demonstrating the benefit that the methodological novelties I have introduced brought in different application areas including that of biological systems, that of manufacturing systems and that of wireless communication networks.

Chapter 2 gives an overview of an important contribution in the field of probabilistic model checking. Specifically we introduce a *statistical model checking* (SMC) framework which relies on a novel automata-based formalism, namely the Hybrid Automata Stochastic Logic (HASL), for expressing properties of a probabilistic model. This set of works started through a collaboration with Serge Haddad, Nihal Pekergin and Marie Duflotwhich was then further extended to Benoit Barbotand resulted with the introduction of the HASL formalism [BDD+11c, BDD+11a, BBD+15a] as well as with the development of the COSMOS software tool [BDD+11a, BB22, COS].

In **Chapter 3** we report on a newly introduced approach for parametric verification of temporal properties against stochastic models, i.e., a framework which addresses the problem of estimating the satisfaction probability of a given temporal logic property in function of a model's parameters. This is achieved through adaptation of the Approximation

Bayesian Computation (ABC) method, a technique for calibrating a model's parameters according to observations of the actual system under study. The adaptation we introduced is based on the notion of *satisfiability distance*[1] used to quantify "how far" (the model instance corresponding to) a parameter set is from satisfying the considered property. In practical terms the framework uses stochastic simulation for sampling paths of a a model's instance and an HASL *monitor* automaton (of Chapter 2) for measuring the satisfiability distance of the sampled trajectories. This contributions have been developed as part of the PhD thesis work of Mahmoud Bentriou [Ben21] which I supervised with Paul-Henry Cournède and have led to the publications of papers [BBC19, BBC21].

In **Chapter 4** we present contributions addressing the problem of formal analysis of stochastic oscillators. Although established mathematical approaches, such as Fourier transform and autocorrelation analysis are commonly employed for the analysis of the periodic signals here we face the problem from a different perspective: we look for a *logical* characterisation of periodicity which can be used *i)* to assess oscillation related indicators against a discrete event stochastic model and *ii)* to search the parameter space of a stochastic oscillator so that the resulting model meets given periodicity constraints (e.g. the average duration of the period). Inspired by the PhD thesis work of David Spieler [Spi13] we extend the characterisation of *noisy periodicity* by introducing rich hybrid automaton specifications for assessing both the period and the amplitude of oscillations. Based on those automata we then further adapt the automata-based adaptation of ABC parameter inference outline in Chapter 3 to the case of oscillatory models. The contributions presented in this chapter are based on the following papers [BMM09, BMR12, BD15, Bal15a].

In **Chapter 5** we describe a number of contributions in the field of performance analysis of stochastic models in different domains, including that of biological systems, of telecommunication networks and of manufacturing systems. The overall goal of this chapter is to demonstrate, by taking into account diverse type of systems, to what extent one can take advantage of formal model-based approached to address relevant performance analysis questions.

Although conclusive remarks concerning research directions on each topic are given at the end of the respective chapters in **Chapter 7** a few general perspectives I wish to consider as directions to guide my future research work are discusssed.

Finally in **Chapter 6** we give a brief overview of other contributions I have worked on and that are not described in this manuscript. Those contributions span rather heterogenous domains, including that of *compositional approaches to stochastic model analysis*, that of *modelling formalisms for timed, stochastic, concurrent systems*, that of *improvements to the approximate Bayesian computation approach*, that of *formal performance modelling of manufacturing systems* and also of *wireless network protocols*. A brief description of each of these other contributions is given together pointers with the corresponding publications.

[1] The dual of *space-time robustness* used in approaches for assessing the robust satisfaction of MITL formulae over non-probabilistic models [DM10].

Chapter 2

Expressive statistical model checking

Since its introduction in the late-70s/early-80s model checking became a prominent technique for the automatic verification of a system's properties. Given a property formally stated through a temporal logic *property language*, algorithms are provided that allow one for establishing whether the property is *satisfied* by a given model of the considered system. The popularity of model checking is mainly due to three factors: (1) the ability to express specific properties by formulas of an appropriate logic, (2) the firm mathematical foundations based on automata theory and (3) the simplicity of the verification algorithms which has led to the development of numerous tools.

With its seminal work [Pnu77] Pnueli set the ground for model checking by introducing the verification of *linear-time temporal logic* (LTL) properties against discrete-state, untimed models of computer programs. LTL semantics gave rise to so-called *linear-time reasoning* as properties are evaluated w.r.t. *linear* sequences of states (i.e. *paths*) that represent possible executions of the system under analysis. The original work of Pnueli was further developed by both Emerson and Clarke [EC80] and Quelles and Sifakis [QS82], with the introduction of the *computation tree logic* (CTL), a temporal logic for *branching time* properties, i.e. properties that allows for taking into account possible *branching* along a model's execution, and hence are evaluated against *trees* (rather than *paths*) issued by a model. Both LTL and CTL properties are inherently *qualitative* as they evaluate to either *true* or *false* against a *transition system* model. Example of properties that can be verified through LTL or CTL model checking are 1) *"absence of concurrent access to a mutually exclusive shared resource"* (safety), or 2) *"possibility to access a shared resource infinitely often"* (liveness). Such kind of properties are called *qualitative* as, by definition, they cannot include any form of quantification. For example, one can state that *"every message that is sent will eventually be received"* but cannot quantify within which delay it will be received.

Notice that LTL and CTL are distinct property languages (some LTL properties have no CTL equivalent) with a non-empty intersection (some LTL properties have a CTL equivalent and vice versa) and that they are both subsumed by the CTL* [EH86] temporal logic.

Probabilistic model checking. As of the mid 90s researchers started to get interested in extending the model checking approach to the verification of probabilistic models. The seminal work of Hansson and Jonnson [HJ90] extended CTL model checking to the verification of discrete-time Markov chains (DTMC), through the introduction of the PCTL temporal logic, namely the probabilistic version of CTL. PCTL allows modellers to express properties which refer to the probability of (discretely) time-bounded events (e.g. *once a message is sent there is at least a 95% probability it will be delivered within 5 time units*). Later on Aziz [ASSB00] first, and Baier *et al.* [BHHK03a] then, further extended CTL model checking to continuous-time Markov chains (CTMCs) by introducing the Continuous Stochastic Logic (CSL) hence allowing for reasoning about the probability of continuously time-bounded events (e.g. *when the queue contains 5 jobs there is at most a 25% chance of a further job arrival within the next 2.75 seconds*). As CTL is a *branching time* logic, so are PCTL and CSL, which means that formulae from both PCTL and CSL are interpreted over the tree consisting of all trajectories unfolded from the considered Markov chain model (either a DTMC for PCTL or a CTMC for CSL). Verifying that a PCTL or CSL specification ϕ holds against a given Markov chain requires assessing the probability measures of the sub-tree that satisfies ϕ, which is well-defined based on the characterisation of the probability space for the trajectories of Markov chains [Kul16]. Specifically, formulae of a probabilistic temporal logic, require comparing the probability that a condition ϕ holds against a probability bound p, denoted $P_{\sim p}[\phi]$, with $\sim \in \{<, \leq, \geq, >\}$ and $p \in [0, 1]$. The *qualitative* fragment of PCTL/CSL is that resulting when the considered probability bound p is either 0 or 1, conversely the *quantitative* fragment is that in which $p \in]0, 1[$. , they are called CSL model checking has then been further extended by Kwiatkowska, Norman and Parker [KNP07a] giving the modeller the possibility to attach (state and/or transition) *rewards* to Markov chain models and to assess *reward based properties* all of which is supported by PRISM [PRI] arguably the most popular probabilistic model checking tool.

Numerical probabilistic model checking of Markovian models. Probabilistic model checking mainly targeted Markov chains models only and consisted of *exhaustive* procedures (i.e., algorithms requiring the complete storage of the Markov chain's *transition matrix*), which is in line with the actual notion of *verification* that inherently entails the truth of a formula must be established by considering a model in its entirety. The definition of *exhaustive* probabilistic model checking algorithms for Markov chain models is relatively straightforward thanks to two main factors: 1) the *memoryless property* of exponential distribution which characterises Markovian models and 2) the (somewhat) limited expres-

siveness of the considered temporal logics (e.g. PCTL and CSL). From an algorithmic point of view the verification of a temporal logic formula against a Markov chain model is obtained by a combination of mathematical methods that depend on the kind of formula. For example for DTMC models the verification of *untimed* PCTL properties boils down to solving of a system of linear equations whereas for *time-bounded* PCTL properties it amounts to matrix-vector multiplications. On the other hand for CTMC models the verification of *untimed* CSL formulae reduces to a PCTL problem on the corresponding *embedded DTMC*, whereas the verification of *time-bounded* CSL properties boils down to a *transient-state analysis* problem for CTMC which is numerically approximated through *uniformisation*.

Beyond Markovian models. Whenever the timing of stochastic events that characterise the considered system's cannot be reasonably assumed to follow an exponential distribution then the underlying stochastic process is no longer Markovian and hence the definition of probabilistic model checking algorithms becomes cumbersome. In this context the *transient stochastic state classes* method, by Horváth, Paolieri, Ridi and Vicario (i.e., a method that allows for keeping track of the evolution in time of the *transient-state* distribution for non-Markovian models) has been showed an effective means for adapting algorithms for the verification of time-bounded properties against Generalised Semi Markov Process (GSMP) models [HPRV11].

The curse of scalability. Independently of the targeted class of models and of the expressiveness of the property language the main limitation of *exhaustive approaches* to model checking is their scalability: as the space complexity of exhaustive algorithms depends on the size of the model they can only be applied to models of "reasonable" size, thus‘ ruling out many real-life applications.

Statistical model checking. In order to overcome the scalability issue of *exhaustive* algorithms, as of the mid 2000s, a novel kind of *non-exhaustive* approach, named *statistical model checking* (SMC), has been introduced by the pivotal work of Younes [YS06]. The main idea with SMC is to employ a finite number of model's *simulations* in order to obtain an approximated answer to the question of whether a probabilistic temporal logic formula $P_{\sim p}[\psi]$ (where $p \in [0,1]$ and $\sim \in \{<, \leq, \geq, >\}$) holds against a stochastic model $\mathcal{M}$ which boils down at deciding whether the probability $Pr(\mathcal{M}, \psi) \sim p$ of those paths satisfying ψ comply with $\sim p$. In this respect there exist two families of SMC approaches: those which aim at *deciding* whether $Pr(\mathcal{M}, \psi) \sim p$ without actually estimating $Pr(\mathcal{M}, \psi)$ (by application of *hypothesis testing*), and those which aim at computing an estimate p' of $Pr(\mathcal{M}, \psi)$ (by application of *confidence interval* methods) and hence establish whether $Pr(\mathcal{M}, \psi) \sim p$ by comparing the estimate $p' \sim p$.

The clear advantage of SMC is that one does not even need to build (hence store) a

system's model, it suffices that a syntactical representation, in terms of some modelling language (equipped with an operational semantics), is available so that a *stochastic simulation algorithm* can be devised for sampling (finite) *trajectories* from the model's state-space. Each trajectory sampled from $\mathcal{M}$ either satisfies (i.e. is *accepted*) or does not satisfy (i.e. is *rejected*) ψ and therefore it corresponds to a realisation of a Bernoulli experiment. As a consequence assessing the probability of ψ against $\mathcal{M}$ in the context of SMC corresponds to estimating the distribution of a Binomial random variable which, in turns, boils down to computing the ratio of sampled trajectories that satisfy ψ.

SMC approaches. Ever since its introduction a number of SMC formalisms, algorithms and tools have been proposed and their effectiveness demonstrated through analysis of large-scale systems in a variety of fields, including computer networks, manufacturing systems, and biological processes [AP18]. Younes opened the way by introducing an *hypothesis testing* based approach to decide, with arbitrary approximation, whether the probability that a CSL property ϕ is satisfied by a model $\mathcal{M}$ comply with a threshold p. An alternative solution to the same kind of problem has then been proposed by Peyrronet *et al.* [HLMP04] by introducing a *randomized approximation scheme* to establish whether the probability that an LTL property ϕ is satisfied by a DTMC match a given constraint p (which has then been integrated amongst the statistical engines of PRISM model checker). Alternatively confidence interval approaches have been proposed to solve the SMC *estimation* problem (i.e. the problem of estimating what is the probability that ϕ is satisfied by $\mathcal{M}$) with dedicated efficient sequential schemes discussed by Jegourel *et al.* [JSD18] which have been adopted in many SMC tools (PRISM included).

Expressiveness: Numerical versus Statistical model checking. The use of statistical methods instead of numerical ones allows for relieving the limitations inherent to stochastic model checking procedures based on numerical methods, both in terms of the family of stochastic models as well as of the type of properties that can be considered. If numerical model checking (NMC) is mainly targeting Markovian models (because of the nice mathematical properties of the Exponential distribution), statistical model checking (SMC) permit one to trespass the Markvovian perimiter, hence to use a very wide range of distributions, and to *synchronise* such a model with an automaton that includes linearly evolving variables, complex updates and constraints. In terms of the properties that one can target SMC allows for a much increased expressiveness. In fact since with NMC the verification of temporal logic properties boils down to either a *transient-state* or *steady-state* distribution computation, the kind of properties that one can formulate are limited: NMC algorithms can only cope with the evaluation of the probability of paths which are characterised by the temporal evolution of state conditions (something which goes under the name of *state-based reachability* problems). With SMC, on the other hand, one is completely freed from the *state-based* reachability perimeter which constrains NMC, therefore

one can easily combine state-conditions with other-conditions, e.g. constraints on given indicators evaluated along the path (e.g. such as waiting time, number of clients in a system, production cost of an item), for characterising the target random variable whose mean value is the target of the SMC verification process. As a result SMC is naturally suited for easily conceive and compute sophisticated performability parameters.

In quest for performance oriented property languages. Despite their success the vast majority of SMC frameworks are based on temporal logic languages (such as PCTL or CSL) that are limited to *state-based* properties, that is, properties that use *state-conditions* as criteria to characterise the behaviour (i.e. the executions) of interest. For example, in a systems consisting of 2 queues where different kind of jobs arrive and compete to access a shared resource an example of property based on *state-conditions* is *"what is the probability that the shared resource is busy when the number of jobs in the first queue does not exceed N?"* In standard performance evaluation problems, however, it makes sense to allow for a wider, *performance-oriented*, class of properties to be considered, i.e., properties that may use *quantitative statistics* (evaluated along a path) as criteria for selecting the executions of interest. For instance, the average length of a waiting queue during a busy period or the mean waiting time of a client are typical examples of criteria that cannot be expressed by the quantitative logics that use *state-conditions* as the only kind of criteria to characterise the executions whose probability shall be assessed.

Outline of contribution. This chapter reports on our contribution in the field of expressive SMC. We introduced a novel, hybrid-automata-based, property specification language named *Hybrid Automata Specification Language* (HASL) [BDD+10, BBD+15b] for expressing temporal-dependent performance indicators referred to a discrete-event stochastic model and we introduced the statistical model checking procedure to assess them together with an associated software tool named COSMOS [BDD+11a, COS] to use it. HASL is the evolution of the time-automata (TA) extension of CSL, namely the CSL^{TA} [DHS07] logic (by Donatelli, Haddad and Sproston), in that it replaces TA which, within CSL^{TA} are used as a machinery for specifying the behaviour of interest that one want to study on a system, with linear hybrid automata (LHA). Employing LHA in place of TA opens up to a much increased expressiveness, as more sophisticated criteria for specifying the behavior of interest can be formulated through an LHA. As it turns out the family of properties that can be taken into account through HASL go beyond those of most popular temporal logic languages rendering HASL a specification language more suited for performance analysis as demonstrated by several applications presented in Chapter 4 and Chapter 5.

2.1 Preliminaries

2.1.1 Discrete event stochastic processes

We target a very broad class of discrete-state, continuous-time stochastic processes which we generically refer to as Discrete Event Stochastic Processes (DESP), but which is often referred to as generalised semi- Markov processes (GSMPs) [Gly83, ACD91]. Differently from the most common probabilistic model checking frameworks such those supported by the PRISM [KNP11] and Storm [HJK+22] tools, with DESPs we get rid of any restriction about the type of probability distribution allowed for modelling events' delay, therefore we are free to leave the realm of *memorylessness* and consider also processes with memory. Syntactically speaking this comes to the price of a more cumbersome formalism as we need to 1) specify the probability distribution associated to each event of the process, 2) account for (time) memory in the specification of an event's occurence and 3) disambiguate between concurrently occurring events, which, within DESPs, may have a non null probability to occur.

Definition 2.1. A Discrete Event Stochastic Process (DESP) is a tuple $\mathcal{M} = \langle S,\pi_0,E,Ind, enabled,delay,choice,target\rangle$ where S is a countable set of states, $\pi_0 \in dist(S)$ is the initial state distribution, E is a finite set of events, $Ind : S \to \mathbb{R}$ is a set of *state indicator* functions, $enabled : S \to 2^E$ denotes the events enabled in each state, $delay : S \times E \to dist(\mathbb{R}^+)$ characterises the probability distribution of the delay of occurence of an event e in state s, $choice\colon S \times 2^E \times \mathbb{R}^+ \to dist(E)$ characterises the probability distribution over the set of concurrently occurring events in a given state and $target\colon S\times E \times \mathbb{R}^+ \to S$ describes which state is entered when from state s an enabled event $e \in enabled(s)$ occurs with a delay d.

Given a state $s \in S$ of a DESP $\mathcal{M}$, $enabled(s)$ is the set of events enabled in s. For an event $e \in enabled(s)$, $delay(s, e)$ is the distribution of the delay between the enabling of e and its possible occurrence. If $d \in \mathbb{R}^+$ is the earliest delay in some configuration of the process with state s, and $E' \subseteq enabled(s)$ the set of events with minimal delay, $choice(s, E', d)$ describes how the conflict is randomly resolved: for all $e' \in E'$, $choice(s, E', d)(e')$ is the probability that e' will be selected among E' after waiting for the delay d. The function $target(s, e, d)$ denotes the target state reached from s on occurrence of e after waiting for d time units.

DESP semantics. As the dynamics of a DESP is not necessarily *memoryless* we need to carry on extra information about a system's state along the unfolding of an execution. Therefore a DESP path amount to a sequences *configurations*, where a configuration consists of a triple $(s\ ,\tau\ ,\ sched)$ with $s \in S$ the current state, $\tau \in \mathbb{R}^+$ the current time and $sched : E \to \mathbb{R}^+ \cup \{+\infty\}$ the function that describes the occurrence time of each enabled event ($+\infty$ if the event is not enabled). From a given configuration $(s, \tau, sched)$ a path

is unfolded by iteratively selecting the next event $e \in enabled(s)$ to occur together with its occurence delay d leading to the next configuration $(s\ ,\ \tau\ ,\ sched) \xrightarrow[e]{d} (s'\ ,\ \tau'\ ,\ sched')$. The next transition to fire is determined through the following steps. First function *sched* determines $E^{min} = \{e \in enabled(s) \mid \forall e' \in enabled(s),\ sched(e) \leq sched(e')\}$, i.e., the set of enabled events with minimal delay $d_m = sched(e) - \tau$, $\forall e \in E'$. Then, if E^m contains more than 1 element, the probability distribution $choice(s, E^{min}, d_m)$ is used to randomly pick the actual next event $e_m \in E_m$ to fire. The next state s' is then determined by firing the previously selected next event to fire e_m, i.e. $s' = target(s, e_m, d_m)$ and also by updating the configuration's time accordingly, i.e., $\tau' = \tau + d_m$. Finally the schedules of currently enabled events is updated by application of function *sched* as follows: first for all events $e' \neq e_m$ that are still enabled, $sched(e')$ is maintained (i.e. $sched'(e') = sched(e)$, $\forall e' \in enabled(s') \cap enabled(s)$, such that $e' \neq e_m$) while for all other "newly" enabled event $e' \in enabled(s')$, a new delay d' is sampled according to the distribution $delay(s', e')$ and $sched(e')$ is set to $\tau + d + d'$. Finally for all disabled events $e' \notin enabled(s')$ the $sched'(e')$ is set to $+\infty$.

Operational semantics of a DESP. The characterisation of the operational semantics of a DESP reduces to showing that a sound definition of probability for a generic path $\sigma : (s_0\ ,\ 0\ ,\ sched_0) \xrightarrow[e_1]{d_1} (s_1\ ,\ \tau_1\ ,\ sched_1) \xrightarrow[e_2]{d_2} \dots$ of a DESP exists. This, in turns, boils down to formally characterising a number of relevant families of random variables, namely: the family $e_1, \dots, e_n, \dots$ (events associated to σ) with support the set of events E, the family $s_0, \dots, s_n, \dots$ (the states of σ) with support the set S of $\mathcal{M}$, the family $\tau_0 \leq \dots \leq \tau_n \leq \dots$ (the occurrence time of the events on σ) with support $\mathbb{R}^+$, the family $\{sched(e)_0, \dots sched(e)_n \dots\}$ with support $\mathbb{R}^+ \cup \{+\infty\}$ (the events' queue along σ) and the family $\{E_0^{min}, \dots E_n^{min} \dots\}$ (the set of events with minimal delay along σ). The characterisation of such families of random variables is inductive w.r.t. the length n of the σ and is formally given in [BBD+15b].

2.1.2 DESP as Non Markovian Generalised Stochastic Petri Nets

For the sake of practicality within HASL we adopt the non-Markovian extension of Generalised Stochastic Petri Net (NMGSPNs) as high level modelling language to represent DESP models. This has two main motivations: 1) to prevent the user from the tedious and error prone need to specifiy a DESP by explicit listing of all of its elements (as of Definition 2.1) and 2) to take advantage of the Petri nets semantics which is particularly well suited to yield highly efficient stochastic simulation algorithms. In essence a NMGSPNs is a generalised stochastic Petri nets [AMBC+95], that is a Stochastic Petri net formalism which combines stochastic timed transitions with probabilistic immediate ones, extended with the possibility to associate generic probability distributions (non necessarily Exponential ones) with stochastic timed transitions, and therefore, according to Definition 2.1,

must be equipped with necessary means to probabilistically disambiguate between events that might occur at the same time.

NMGSPN models and token game. A NMGSPN model is a bipartite graph consisting of *places* (circles) and *transitions* (bars) nodes. Places may contain *tokens* (representing the state of the modelled system) while transitions indicate how tokens "flow" within the net (encoding the model dynamics). The state of a NMGSPN (hence of the DESP it represents) consists of a *marking* indicating the distribution of tokens among the places (i.e. how many tokens each place contains). A transition t is enabled whenever every *input place* of t contains a number of tokens greater than or equal to the multiplicity of the corresponding (input) arc. An enabled transition may *fire*, consuming tokens (in a number indicated by the multiplicity of the corresponding input arcs) from its input places, and producing tokens (in a number indicated by the multiplicity of the corresponding output arcs) in its *output places*. Transitions can be either *timed* (denoted by thick bars) or *immediate* (denoted by thin filled-in bars, see Figure 2.1). Transitions are characterised by: (1) a *distribution* which randomly determines the delay before firing it; (2) a *priority* which deterministically selects among the transitions scheduled the soonest, the one to be fired; (3) a *weight*, that is used in the random choice between transitions scheduled the soonest with the same highest priority. With the original GSPN formalism [AMBC$^+$95] the delay of timed transitions is assumed *exponentially* distributed, whereas with our NMGSPN it can be given by any distribution. Thus a NMGSPN timed-transition is characterised by a tuple: $t \equiv (type, par, pri, w)$, where *type* indicates the type of distribution (e.g. uniform), *par* indicates the parameters of the distribution (e.g $[\alpha, \beta]$), $pri \in \mathbb{R}^+$ is a priority assigned to the transition and $w \in \mathbb{R}^+$ is used to probabilistically choose between transitions occurring with equal delay and equal priority. The information associated with a transition (i.e. $type, par, pri, w$) is exploited in different manners depending on the type of transition. For example for a transition with a continuous distribution the priority (pri) and weight (w) records are superfluous (hence ignored) since the probability that the schedule of the corresponding event is equal to the schedule of the event corresponding to another transition is null. Similarly, for an immediate transition (denoted by a filled-in bar) the specification of the distribution type (i.e. *type*) and associated parameters (*par*) is irrelevant (hence also ignored). Therefore these unnecessary informations are omitted in Figure 2.1.

Example 1 (Mutual exclusive shared resource)**.** The NMGSPN model of Figure 2.3 (top) describes the behaviour of an open system where two classes of clients (processes) (namely 1 and 2) compete to access a shared resource (e.g. the memory). Class i-clients ($i \in \{1, 2\}$) enter the system according to a Poisson process with parameter λ_i (corresponding to the exponentially distributed timed transition Arr_i with rate λ_i, represented by thick empty bars). On arrival, clients cumulate in places Req_i waiting for the resource to be free (a token in place $Free$ witnessing that the resource is available). The exclusive access to the shared resource is regulated either deterministically or probabilistically by the *priority*

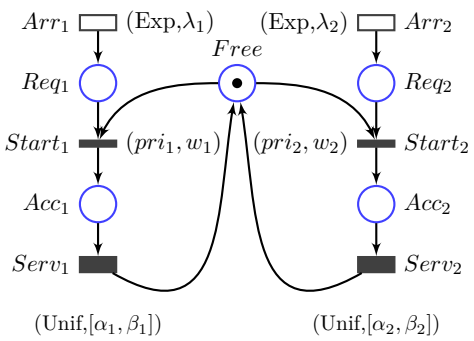


Figure 2.1: The NMGSPN description of a shared resource system.

(pri_i) and the *weight* (w_i) of immediate transitions $Start_1$ and $Start_2$. Thus in presence of a competition (i.e. one or more tokens in both Req_1 and Req_2) a class i client wins the competition with a class $j \neq i$ client with probability 1 if $pri_i > pri_j$, and with probability $w_i/(w_i+w_j)$ if $pri_i = pri_j$. The occupation time of the resource by a class i client is assumed to be uniformly distributed within the interval $[\alpha_i, \beta_i]$ (corresponding to the thick filled-in transitions $Serv_i$). Thus on firing transition $Serv_i$ the resource is released and a class i client leaves the system. Notice that graphically Exponential, respectively non-Markovian, timed transitions are represented by thick empty, respectively filled-in, bars).

2.2 Hybrid Automata Stochastic Language

Alur and Henzinger's hybrid automata [ACHH92, ACHH93] are a popular formalism for modelling linear hybrid systems, i.e. systems consisting of interacting digital and analog components. Although reachability problems for hybrid automata are, generally speaking, undecidable, Henzinger *et al.* proved that under certain conditions reachability decidability can be established [HKPV98] leading to the development of dedicated model checkers such as the HyTech tool [HHWT97]. Here we consider hybrid automata from a different perspective, that is, as a language to specify properties of a system rather than as a language to model a system. To this aim we introduce the Hybrid Automata Stochastic Language (HASL) [BDD$^+$11b, BBD$^+$15a], the basic idea it being to employ a tailored linear hybrid automaton (LHA) $\mathcal{A}$ as a *monitor* for selecting relevant paths of a DESP model $\mathcal{M}$, based on sophisticated selection criteria. This allows one to conceive very sophisticated performance indicators of a model $\mathcal{M}$, by means of a dedicated automaton $\mathcal{A}$ and to assess them through *synchronisation*, that is, through a process by which paths $\sigma \in Path(\mathcal{M})$ are processed *on-the-fly* and are either accepted or rejected by $\mathcal{A}$, while relevant statistics are maintained in the automaton's variables and are eventually used to produce estimates

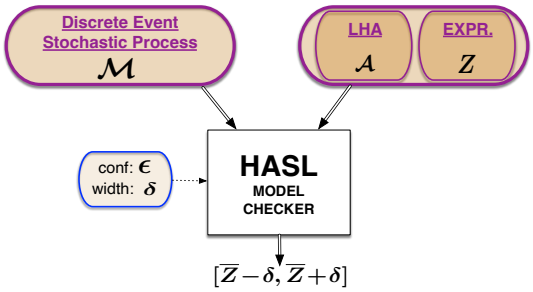


Figure 2.2: HASL-SMC schema: sampled paths are filtered by a LHA and the accepted ones are used for a confidence interval estimate of the target measure Z.

of the performance indicators of interest.

In practical terms a HASL formula ϕ consists of two components $\phi = (\mathcal{A}, Z)$, where $\mathcal{A}$ is the *monitor* LHA and Z is an expression that specifies the quantity to be evaluated (based on the $\mathcal{A}$'s data variables).

HASL model checking scheme. The synchronisation between a DESP model $\mathcal{M}$ and a LHA $\mathcal{A}$ naturally leads to definition of a statistical model checking procedure for the HASL formalism which is outlined in Figure 2.2. A HASL model checker takes a DESP model $\mathcal{M}$ and a HASL formula $(\mathcal{A}, Z)$ as inputs together with two auxiliaries parameters, i.e., the confidence level (ϵ) and the width (δ) of the confidence interval estimation process. The LHA part of a HASL formula is used to select, via synchronisation, the paths sampled on the DESP model and while so doing the (real-valued) variables of the LHA are updated with relevant statistics of the synchronising path. Such statistics are then employed for producing a confidence-interval estimation of the mean value of the considered expression Z (i.e. the second part of the HASL formula). Z is the quantity the modeller wants to assess w.r.t. the paths accepted by LHA.

2.2.1 Synchronised Linear Hybrid Automata

In the context of HASL model checking we introduce an adaptation of Alur and Henzinger's hybrid automata [ACHH92] which we refer to as synchronised LHA the main difference being that HASL automata are designed to interact with a DESP model, hence they must fulfil certain syntactical as well as semantical constraints. From a syntactical perspective there are two elements that impact the syntax of a HASL hybrid automaton definition: first, an automaton edges are now labeled either with events of the DESP model, indicating that the edge traversal takes place only through synchronised occurrence of a corresponding DESP event or, alternatively, through a special label $\sharp$, indicating that the edge traversal happens autonomously without synchronisation with any DESP event, and second, an automaton's variables can evolve at a rate which depends on the current state of the

DESP model (i.e. the rate of a variable is given by a DESP *indicator* function). From a semantical perspective we need to establish a number of *determinism-related* constraints so to guarantee the unicity of synchronisation, that is, the fact that there exists always a single execution of the synchronised product process ($\mathcal{M} \times \mathcal{A}$). All of these aspects are covered by the following Definition.

Definition 2.2. A *synchronised linear hybrid automaton*, referred to a DESP model $\mathcal{M}$ with states S, events E and indicator functions $Ind : S \to \mathbb{R}$, is defined by a tuple $\mathcal{A} = \langle E\ ,\ L\ ,\ \Lambda\ ,\ Init\ ,\ Final\ ,\ X\ ,\ flow,\ \rightarrow \rangle$ where: E is a finite alphabet of events (the events of $\mathcal{M}$); L is a finite set of locations; $\Lambda : L \to Prop$ denotes the locations' invariants[1]; $Init, Final \subset L$ denotes the initial, resp. final, locations; $X = (x_1, ...x_n)$ is a n-tuple of data variables, and:

- $flow : L \mapsto Ind^n$ associates with each location one indicator per data variable representing the evolution rate of the variable in the location.
- $\rightarrow \subseteq L \times \big((2^E \times \mathsf{Const}) \uplus (\{\sharp\} \times \mathsf{IConst})\big) \times \mathsf{Up} \times L$ is a set of edges, where Const, (resp. IConst), denotes the set of *linear constraints*, (resp. left-closed constraints) consisting of a boolean combination of inequalities of the form $\sum_{1 \leq i \leq n} \alpha_i x_i + c \prec 0$ where $\alpha_i, c \in Ind$ are DESP indicator functions and $\prec \in \{=, <, >, \leq, \geq\}$, $\uplus$ denotes the disjoint union and Up denotes the set of n-tuples of update functions with the following form $x_k = \sum_{1 \leq i \leq n} \alpha_i x_i + c$ where again $\alpha_i, c \in Ind$.

Furthermore $\mathcal{A}$ fulfils the following conditions.

- **Initial determinism:** $\forall l \neq l' \in Init$, $\Lambda(l) \wedge \Lambda(l') \Leftrightarrow \texttt{false}$. This must hold whatever the interpretation of the indicators occurring in $\Lambda(l)$ and $\Lambda(l')$.
- **Determinism on events:** $\forall E_1, E_2 \subseteq E\ s.t.\ E_1 \cap E_2 \neq \emptyset$, $\forall l, l', l'' \in L$, if $l'' \xrightarrow{E_1,\gamma,U} l$ and $l'' \xrightarrow{E_2,\gamma',U'} l'$ are two distinct transitions, then either $\Lambda(l) \wedge \Lambda(l') \Leftrightarrow \texttt{false}$ or $\gamma \wedge \gamma' \Leftrightarrow \texttt{false}$. Again this equivalence must hold whatever the interpretation of the indicators occurring in $\Lambda(l)$, $\Lambda(l')$, γ and γ'.
- **Determinism on $\sharp$:** $\forall l, l', l'' \in L$, if $l'' \xrightarrow{\sharp,\gamma,U} l$ and $l'' \xrightarrow{\sharp,\gamma',U'} l'$ are two distinct transitions, then either $\Lambda(l) \wedge \Lambda(l') \Leftrightarrow \texttt{false}$ or $\gamma \wedge \gamma' \Leftrightarrow \texttt{false}$.
- **No $\sharp$-labelled loops:** For all sequences $l_0 \xrightarrow{E_0,\gamma_0,U_0} l_1 \xrightarrow{E_1,\gamma_1,U_1} \cdots \xrightarrow{E_{n-1},\gamma_{n-1},U_{n-1}} l_n$ such that $l_0 = l_n$, there exists $i \leq n$ such that $E_i \neq \sharp$.

In the remainder we write $l \xrightarrow{E',\gamma,U} l'$ to denote an edge $(l, E', \gamma, U, l') \in \rightarrow$ of an LHA, furthermore we refer to an edge such as $l \xrightarrow{\gamma,E',U} l'$, where $E' \subseteq E$ is a subset of events,

[1] Where $Prop \subset Ind$ is the subset of DESP indicator functions that evaluates to $\{0, 1\}$

as a *synchronising edge* as opposed to an edge such as $l \xrightarrow{\gamma,\sharp,U} l'$ which we refer to as an *autonomous edge*.

Example 2. The two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ in Figure 2.3 are designed to be synchronised with the NMGSPN model of mutual exclusion of Example 1. Both $\mathcal{A}_1$ and $\mathcal{A}_2$ consists of

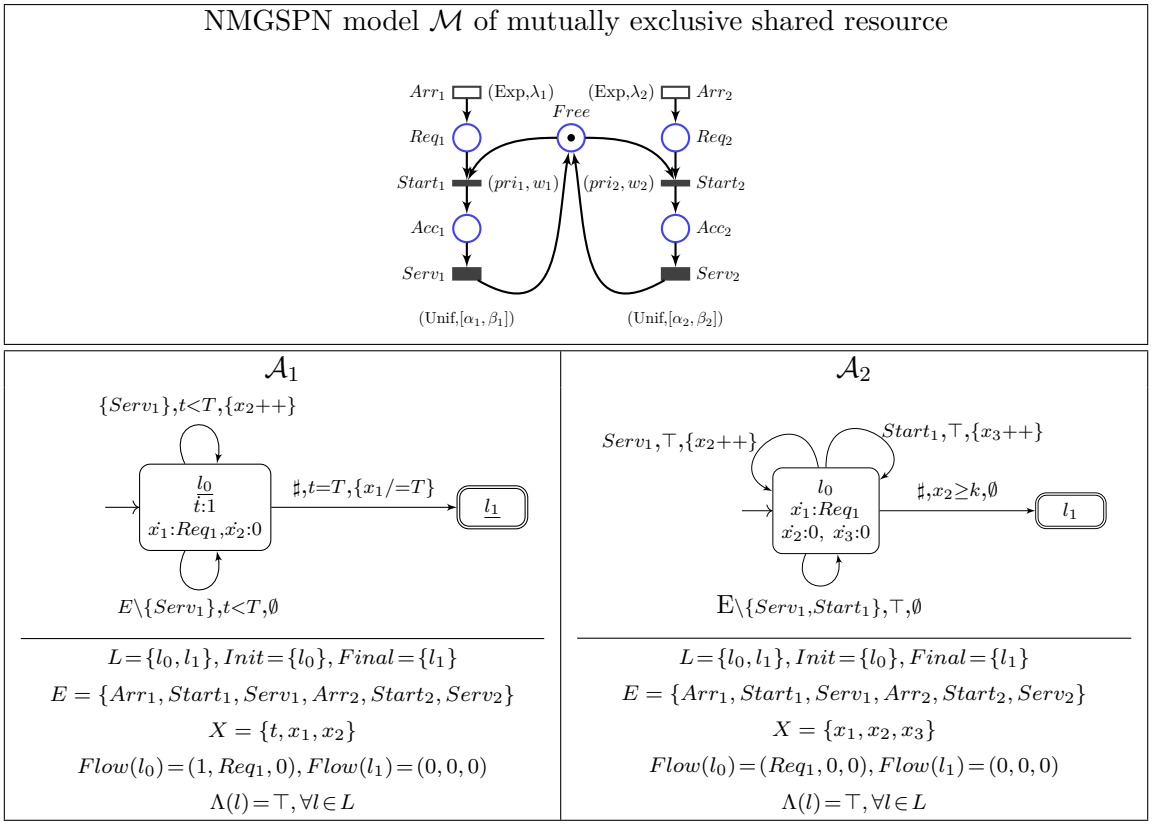


Figure 2.3: Examples of two LHA referred to mutual exclusion NMGSPN model

2 locations $L=\{l_0, l_1\}$, an initial one $Init=\{l_0\}$ and a final one $Final=\{l_1\}$. They both refer to the same event set $E = \{Arr_1, Start_1, Serv_1, Arr_2, Start_2, Serv_2\}$ i.e. that of the transitions of the NMGSPN model of mutual exclusion and both are equipped with 3 data variables, though $\mathcal{A}_1$ has a clock variable (t) whereas $\mathcal{A}_2$ is not equipped with any clock. The flow of variables is conventionally null in the final location ($Flow(l_1) = (0,0,0)$) as once in the final location the synchronisation ends[2], whereas is non-null (for certain variables)

[2]in fact the flow of variables in any final location could equivalently be set to an arbitrary value as it cannot affect the synchronisation process given the synchronisation stop precisely on entering a final location

in the initial location l_0: variable x_1 (for both $\mathcal{A}_1$ and $\mathcal{A}_2$) evolves at a rate given by the marking of place Req_1 of the NMGSPN model, whereas the clock t, for $\mathcal{A}_1$, naturally has constant rate of 1. As for the edges $\mathcal{A}_1$ has 2 synchronising self-loop edges on l_0, $l_0 \xrightarrow{\{Serv_1\},t<T,\{x_2++\}} l_0$, which synchronises with the occurrences of the $Serv_1$ transition of the NMGSPN model (and in so doing it stores the occurrences of $Serv_1$ in variable x_2, and $l_0 \xrightarrow{E\setminus\{Serv_1\},t<T,\emptyset} l_0$ which synchronises with any other transition (without registering any information in the variables of $\mathcal{A}_1$) plus an autonomous edge $l_0 \xrightarrow{\sharp,t=T,\{x_1/=T\}} l_1$ which establishes the end of the synchronisation as soon as the clock is $t = T$ (where T is a parameter of $\mathcal{A}_1$), therefore paths accepted by $\mathcal{A}_1$ are *time-bounded*. Similarly $\mathcal{A}_2$ has 3 synchronising self-loop edges on l_0, one for counting the occurrences of $Serv_1$ (stored in x_2), one for counting the occurrences of $Start_1$ (stored in x_3) and the remaining for ignoring any other transition occurring during the synchronisation. $\mathcal{A}_2$ autonomous edge establishes the ending of the synchronisation as soon as k (where k is a parameter of $calA_2$) occurrences of the $Serv_1$ transition have been observed, therefore paths accepted by $\mathcal{A}_2$ are *event-bounded*.

LHA semantics Intuitively

Real-valued variables with linear flow. An LHA is equipped with a n-tuple of real-valued variables $x_1, \ldots, x_n \in \mathbb{R}$ (called data variables) whose value evolves with a linear rate that depends both on the location of the automaton and on the current state of the DESP[3]. More precisely, a *flow* function associates with each location l of the automaton an n-tuple of real-valued *indicator* functions (one for each variable) which define how the value of each variable evolves, in function of the current state of the DESP, while the LHA spends time in l. For example $flow_i(l)(s)$ (where $flow_i(l)$ is the i^{th} component of $flow(l)$) denotes the flow of variable x_i while the LHA is in l and the DESP in s.

Autonomous versus synchronised transitions. An edge of a LHA is characterised by 3 components: $(eventSet, constraint, update)$, where $eventSet$ denotes either a subset of E, the event set of the DESP, or the special sympbol $\sharp$ (denoting a pseudo-event that is not included in the event set E), *constraint* is a boolean proposition composed of linear inequalities on the LHA variables x_i and *update* denotes a set of equations describing how the value of variables x_i is updated when the edge is traversed. Edges are enabled (depending on the LHA variables and possibly also on the state of the DESP) as soon as the *constraint* condition is satisfied. Depending on $eventSet$ we distinguish between two type of edges: *autonomous* and *synchronised*. When enabled an autonomous ($\sharp$) edge can be traversed *autonomously* (i.e. without synchronisation with the occurrence of a corresponding DESP event). On the contrary a synchronised edge is associated to a subset of

[3]the value of x_i evolves in a location of the LHA according to a rate defined by a real-valued linear function of some model's *indicator*

events $eventSet \subseteq E$, meaning that, when enabled, it can be traversed only through synchronisation with the occurrence of a DESP event $e \in eventSet$. Notice that autonomous edges have priority on synchronised ones: if both are enabled the autonomous edge will be traversed before the synchronous. Because of the co-existence of autonomous and synchronised edges an LHA is thus capable of taking into account, on one hand the system behaviour, through synchronised transitions, but also to autonomously move the automaton to another location, through autonomous transitions, as soon as specific conditions captured by the data variables x_i take place.

A *constraint* of an LHA edge is a boolean combination of inequalities of the form $\sum_{1\leq i\leq n} \alpha_i x_i + c \prec 0$ where $\alpha_i, c \in Ind$ are indicators, and $\prec \in \{=, <, >, \leq, \geq\}$. The set of constraints is denoted by Const. Given a location and a state, an expression of the form $\sum_{1\leq i\leq n} \alpha_i x_i + c$ linearly evolves with time. An inequality thus gives an interval of time during which the constraint is satisfied. We say that a constraint is left closed if, whatever the current state s of the DESP (defining the values of indicators), the time at which the constraint is satisfied is a union of left closed intervals. These special constraints are used for the "autonomous" edges, to ensure that the first time instant at which they are satisfied exists. We denote by IConst the set of left closed constraints.

An *update* is more general than the reset of timed automata. Here each data variable can be set to a linear function of the variables' values. An update U is then an n-tuple of functions $u_1, ..., u_n$ where each u_k is of the form $x_k = \sum_{1\leq i\leq n} \alpha_i x_i + c$ where the α_i and c are indicators. The set of updates is denoted by Up.

Finally each location of an LHA is also associated with an *invariant* (through a labelling function Λ), i.e. a boolean valued proposition built on top of DESP indicators. A location invariant plays also a role in the enabling of LHA edges (autonomous and synchronised): given in the current location l the constraint c of an edge $l \xrightarrow{-,c,u} l'$ is satisfied, the edge $l \xrightarrow{-,c,u} l'$ is actually enabled only if the invariant of $\Lambda(l')$ will also be satisfied after traversing $l \xrightarrow{-,c,u} l'$ (hence after applying the update u to the LHA variables).

Remark 1 (Unique synchronisation)**.** The four constraints outlined in Definition 2.2 let the automata we consider deterministic in the following (non usual) sense. Given a path σ of a DESP, there is at most one synchronisation with the linear hybrid automaton. The three first constraints ensure that the synchronised system is still a stochastic process. The fourth condition disables "divergence" of the synchronised product, i.e. the possibility of an infinity of consecutive autonomous events without synchronisation.

Remark 2 (Linearity)**.** We stress that the linearity restriction that concerns different elements of an LHA (i.e. the guards and the updates of an arcs as well as the flow of the variables) can be relaxed, as long as those elements do not concern autonomous transitions. Polynomial evolution of constraints could easily be allowed for synchronised edges for which we would just need to evaluate the expression at a given time instant. Since the best

algorithms solving polynomial equations operate in PSPACE [Can88], such an extension for autonomous transitions cannot be considered for obvious efficiency reasons.

Example 3. Figure 2.4 depicts two slight variants of an LHA for measures related to the time the shared resource of the mutual exclusion model is occupied by class 1 clients or by class 2 clients. Both automata employ 2 variables: x_0 that counts the number of completed occupation by either class of clients and x_1 dedicated to measuring the difference between the time the resource is occupied by clients of type 1 minus the time it is occupied by clients of type 2, and to this aim variable x_1 has flow 1 (*resp.* -1) when the resource is used by class 1 (*resp.* 2) clients, and 0 when the resource is not used. The termination condition, given in the constraint of arcs leading to the final location l_3, is defined so that synchronisation ends on observing the k-th occurrence of either $Serv_1$ or $Serv_2$ (corresponding to the completion of occupation of the shared resource by either type of client), i.e. k being the sum of completed occupation of the resource. The most peculiar characteristic of the automata in Figure 2.4 is that they both use *location invariants*, (i.e. state-dependent proposition, corresponding to function Λ of Definition 2.2) associated with the non-final locations: proposition $\mathbf{Acc_1} > \mathbf{0}$ (there is at least a token in place Acc_1) associated to l_1, $\mathbf{Acc_2} > \mathbf{0}$ (there is at least a token in place Acc_2) associated to l_2 and $\mathbf{Free} > \mathbf{0}$ (there is no token neither in Acc_1 nor in Acc_2) associated with the initial location l_0. Hence, starting from location l_0 (assuming initially the resource is free), no matter which precise event occurs in the system, the automaton will switch from l_0 to l_1 and l_2 depending on which class of clients has access to the resource. The fact that for example three different transitions labelled with E without any constraint are available in location l_0 does not induce non determinism as only one of these transitions is possible at a time thanks to the location invariants. The synchronisation terminates in location l_3 on observation of the k-th departure of (any) client from the system and on condition that type 1 clients have used the resource longer than type 2's (constraint $x_1 \geq 0$ for $\mathcal{A}_{3a}$) or viceversa on condition that type 2 clients have used the resource longer than type 1's (constraint $x_1 < 0$ for $\mathcal{A}_{3b}$). This allows to assess, through a corresponding HASL expression $Z \equiv PROB()$ (see Section 2.2.2) the probability that type 1 (resp. type 2) clients use the resource longer than type 2 (resp. type 1).

Operational Semantics: synchronised process $\mathcal{M} \times \mathcal{A}$. The formal characterisation of such *synchronisation process* boils down to showing that the synchronised process is itself a DESP (equipped with absorbing accept/reject states) and by providing its formal definition (Definition 2.3) in function of the synchronising components, that is the DESP model $\mathcal{M}$ and the LHA $\mathcal{A}$.

Notations. Given a DESP model $\mathcal{M} = \langle S, \pi_0, E, Ind, enabled, delay, choice, target \rangle$ and an LHA automaton $\mathcal{A} = \langle E, L, \Lambda, Init, Final, X, flow, \rightarrow \rangle$ in the remainder we adopt the

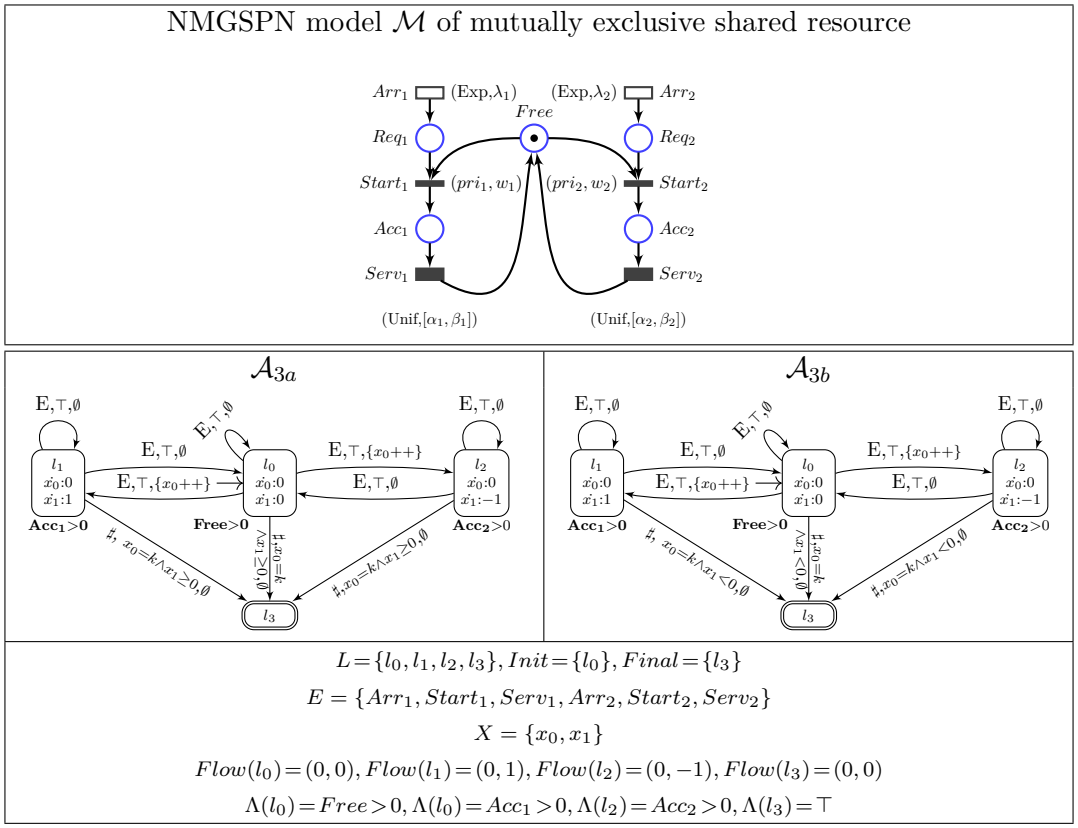


Figure 2.4: An LHA to assess the probability that class 1 (resp. class 2) clients use the shared resource longer than class 2's. (resp. class 1's).

following notations:

- LHA variables valuations: we denote $\nu : X \to \mathbb{R}$ a *valuation* of the LHA variables and Val the set of all possible valuations. For $\nu \in Val$ and $x_i \in X$ we denote $\nu(x_i)$ the value of x_i in ν. We denote $\nu = 0$ the valuation that assigns 0 to each variables $x_i \in X$.
- LHA constraints valuations: let $\nu \in Val$ be a valuation, $s \in S$ a state of $\mathcal{M}$ and $exp = \sum_{1 \leq i \leq n} \alpha_i x_i + c$ an expression related to variables $x_i \in X$ and indicators $\alpha_i, c \in Ind$ we denote $exp(s, \nu)$ the interpretation of exp w.r.t. ν and s where $exp(s, \nu)$ is defined by $exp(s, \nu) = \sum_{1 \leq i \leq n} \alpha_i(s)\nu(x_i) + c(s)$.
- LHA constraints satisfaction: let $\nu \in Val$ be a valuation, $s \in S$ a state of $\mathcal{M}$ and $\gamma \equiv exp \prec 0 \in (\mathsf{Const} \cup \mathsf{IConst})$ a constraint built on top of an expression $exp = \sum_{1 \leq i \leq n} \alpha_i x_i + c$, we write $(s, \nu) \models \gamma$ (i.e. (s, ν) satisfies γ) if $exp(s, \nu) \prec 0$.

- LHA updates valuation: let $\nu \in \mathit{Val}$ be a valuation, $s \in S$ a state of $\mathcal{M}$ and $U = (u_1, \ldots, u_n) \in \mathsf{Up}$ be an update (appearing on some edge) of the LHA we denote by $U(s,\nu)$ the valuation defined by $U(s,\nu)(x_k) = u_k(s,\nu)$ for $1 \leq k \leq n$.

- state proposition satisfaction: Let $s \in S$ be a state of $\mathcal{M}$ and $\varphi \equiv \sum_{1 \leq i \leq n} \alpha_i + c \prec 0$ be a state proposition (i.e. a constraint built on an expression that involves only indicators of $\mathcal{M}$ and no variables of the LHA). We write $s \models \varphi$ (i.e. φ is satisfied in s) if $\varphi(s) = \mathbf{true}$.

- Time elapsing: Given a state $s \in S$ of $\mathcal{M}$ a non final location $l \in L$ and a valuation ν of $\mathcal{A}$, we define the effect of time elapsing by: $\mathit{Elapse}(s,l,\nu,\delta) = \nu'$ where, for every variable x_k, $\nu'(x_k) = \nu(x_k) + \mathit{flow}_k(l)(s) \times \delta$.

- Earliest autonomous delay: Given a state $s \in S$ of $\mathcal{M}$ a non final location $l \in L$ and a valuation ν of $\mathcal{A}$ we denote $\mathit{Autdel}(s,l,\nu)$ the minimum delay which enables an outgoing autonomous edge originating in l,

$$\mathit{Autdel}(s,l,\nu) = \min(\delta \mid \exists l \xrightarrow{\sharp,\gamma,U} l' \wedge s \models \Lambda(l') \wedge (s, \mathit{Elapse}(s,l,\nu,\delta)) \models \gamma)$$

  Remark: Whenever $\mathit{Autdel}(s,l,\nu)$ is finite, we know that there is at least one executable transition with minimal delay and, thanks to the "determinism on $\sharp$" (Definition 2.2), we know that this transition is unique. We call the transition with such earliest delay the *earliest* and we denote $\mathit{Next}(s,l,\nu)$ its target location and $\mathit{Umin}(s,l,\nu)$ its update.

Definition 2.3 ($\mathcal{M} \times \mathcal{A}$ synchronised process)**.** Let $\mathcal{M} = \langle S, \pi_0, E, \mathit{Ind}, \mathit{enabled}, \mathit{delay}, \mathit{choice}, \mathit{target}\rangle$ be a DESP and $\mathcal{A} = \langle E, L, \Lambda, \mathit{Init}, \mathit{Final}, X, \mathit{flow}, \rightarrow\rangle$ a corresponding LHA we define $\mathcal{M}' = \langle S', \pi'_0, E', \mathit{Ind}', \mathit{enabled}', \mathit{target}', \mathit{delay}', \mathit{choice}'\rangle$ as the DESP corresponding to the synchronisation of $\mathcal{M}$ with $\mathcal{A}$ as follows:

- $S' = (S \times L \times \mathit{Val}) \uplus \{\bot\}$ among which $(S \times \mathit{Final} \times \mathit{Val}) \uplus \{\bot\}$ are the absorbing states.

- $\pi'_0(s,l,\nu) = \begin{cases} \pi_0(s) & \text{if } (l \in \mathit{Init} \wedge s \models \Lambda(l) \wedge \nu = 0) \\ 0 & \mathit{otherwise} \end{cases}$
  and $\pi'_0(\bot) = 1 - \Sigma_{s \in S, l \in L, \nu \in \mathit{Val}} \pi'_0(s,l,\nu)$.
  Note that this definition gives a distribution since, due to "initial determinism" (Definition 2.2), for every $s \in S$, there is at most one $l \in \mathit{Init}$ such that $s \models \Lambda(l)$.

- $E' = E \uplus \{\sharp\}$

- $\mathit{Ind}' = \emptyset$. In fact Ind' is useless since there is no more synchronisation to make.

- $enabled'(s,l,\nu) = \begin{cases} enabled(s) \cup \{\sharp\}, & \text{if } Autdel(s,l,\nu) \neq \infty \\ enabled(s), & \text{otherwise} \end{cases}$
- $delay'((s,l,\nu),e) = \begin{cases} delay(s,e), & \text{if } e \in enabled(s) \wedge \sharp \notin enabled'(s,l,\nu) \\ Dirac(Autdel(s,l,\nu)) & \text{if } \sharp \in enabled'(s,l,\nu) \end{cases}$
  Notice that the Dirac distribution centred on the earliest autonomous delays indicates the priority of autonomous transitions over synchronised ones.
- $choice'((s,l,\nu),E',d)(e) = \begin{cases} 1, & \text{if } (\sharp \in E' \wedge e = \sharp) \\ 0, & \text{if } (\sharp \in E' \wedge e \neq \sharp) \\ 0, & e \notin E' \\ choice(s,E',d)(e) & \text{otherwise} \end{cases}$
  Again this is coherent since, as soon as $\sharp \notin E'$, then E' is a subset of $enabled(s)$ on which *choice* is thus defined.
- target for occurrence of a synchronising event $e \in E$
  $target'((s,l,\nu),e,d) = \begin{cases} (target(s,e,d),l',\nu'), & \text{if } (\exists l \xrightarrow{E',\gamma,U} l' \in \mapsto) \wedge (e \in E' \cap enabled(s)) \\ & \quad \wedge Elapse(s,l,\nu,d) \models \gamma \\ & \quad \wedge target(s,e,d) \models \Lambda(l') \\ & \quad \wedge \nu' = U(Elapse(s,l,\nu,d)) \\ \bot & \text{otherwise} \end{cases}$
  Notice that due to the determinism constraints there is at most one such transition.
- target for occurrence of an autonomous event $\sharp$
  $target'((s,l,\nu),\sharp) = \begin{cases} (s,l',\nu'), & \text{if } (\exists l \xrightarrow{\sharp,\gamma,U} l' \in \mapsto) \wedge (\sharp \in enabled'(s,l,\nu)) \\ & \quad \wedge l' = Next(s,l,\nu) \\ & \quad \wedge \nu' = Umin(s,l,\nu)(Elapse(s,l,\nu,Autdel(s,l,\nu))) \\ \bot & \text{otherwise} \end{cases}$
- In order to get a DESP, for any absorbing state one adds a special event **tick** only enabled in absorbing states with Dirac distribution on value 1 (so that time diverges in these states).

Summary remarks. Roughly speaking the formal characterisation of synchronised ($\mathcal{M} \times \mathcal{A}$) process given above (Definition 2.3) tells us that as long as the automaton is not in a final state, the synchronisation of $\mathcal{M}$ with $\mathcal{A}$ waits for the first (synchronised or autonomous) transition to occur. If an autonomous transition occurs then only the location

of the automaton $\mathcal{A}$ and the valuation ν of its variables change, whereas, if a synchronising event of $\mathcal{M}$ occurs then, either the LHA can take a corresponding transition and the system goes on with the next transition or the system goes to a dedicated rejecting state $\perp$ implying the immediate end of the synchronisation. In case of a conflict between two transitions, an autonomous and a synchronised one, the autonomous transition is taken first. Furthermore note that, by initial determinism, for every $s \in S$ there is at most one $l \in \mathit{Init}$ such that s satisfies $\Lambda(l)$. In case there is no such l the synchronisation starts and immediately ends up in the additional state $\perp$. Determinism on events (*resp.* on $\sharp$) ensures that there is always at most one synchronised (*resp.* autonomous) transition fireable at a given instant.

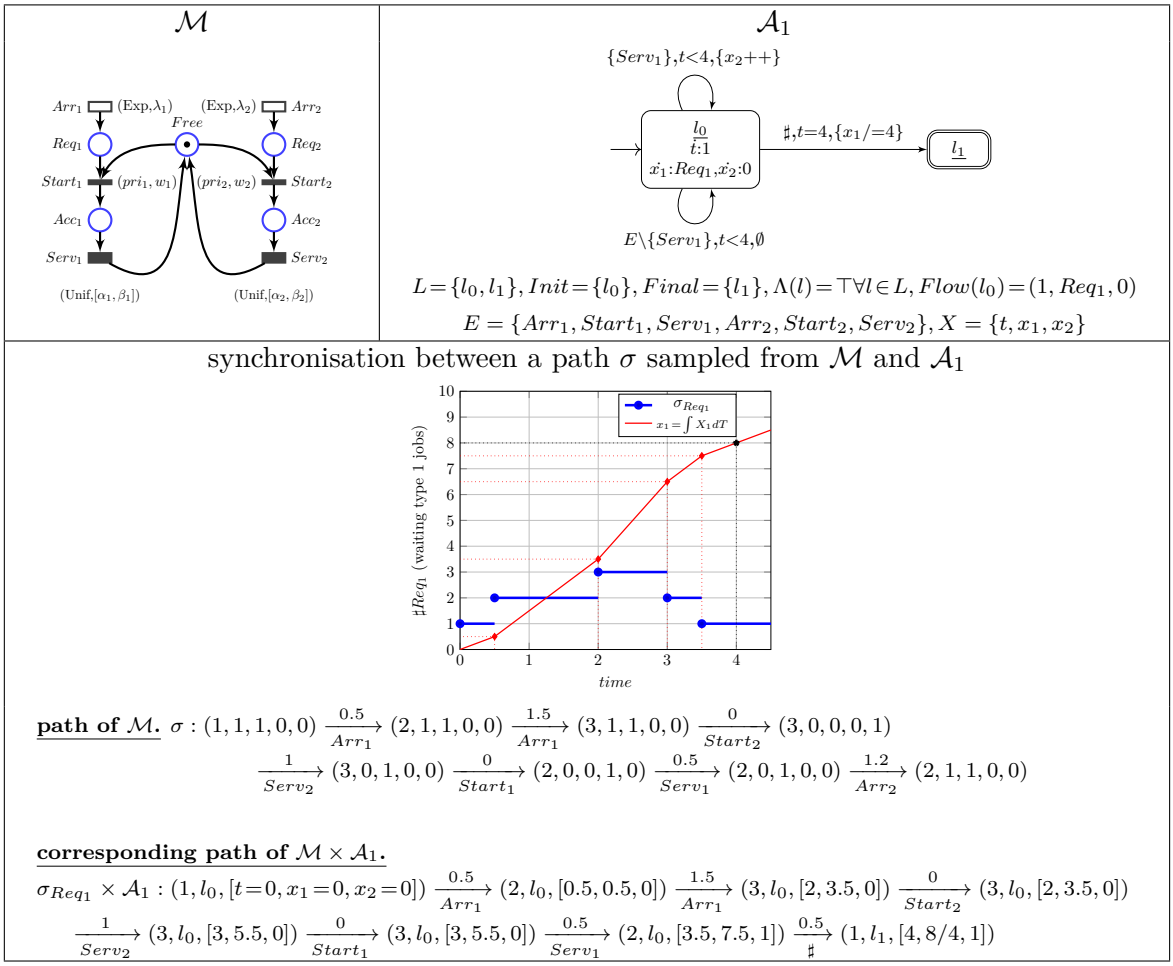


Figure 2.5: Example of synchronisation of a NGGSPN model with a LHA.

Example 4 (LHA and DESP synchronisation)**.** Figure 2.5 depicts an example of synchronisation between the NMGSPN model $\mathcal{M}$ (top left) of the mutual exclusion system (Example 1) with the LHA $\mathcal{A}_1$ of Example 2 (top right). The example shows a path σ (sampled from $\mathcal{M}$ and whose projection σ_{Req_1} over the Req_1 place is depicted as the blue step-like plot in the bottom picture) and the corresponding path $\sigma_{Req_1} \times \mathcal{A}_1$ induced by σ on the synchronised product process $\mathcal{M} \times \mathcal{A}_1$. The bottom picture includes also a plot (i.e. red linear plot) of the corresponding time evolution (induced by synchronisation with σ) of the x_1 variable of $\mathcal{A}_1$ that is the variable storing the value of the integral of the number of class 1 processes waiting for the shared resource.

We quickly recall that $\mathcal{A}_1$ uses a clock variable t to measure time elapsing and a variable x_1 which, while spending time in l_0, register the integral of class 1 processes waiting to get access to the shared resource ($Flow(l_0)(x_1) = Req_1$). The initial state of path σ is assumed to be $(Req_1 : 1, Req_2 : 1, Free : 1, Acc_1 : 0, Acc_2 : 0)$[4], i.e. we assume that initially 1 process of each class is waiting to access the shared resource and that the resource is free hence the initial corresponding configuration of the $\mathcal{M}_{Req_1} \times \mathcal{A}_1$ product process is $(1, l_0, [t=0, x_1=0, x_2=0])$ denoting that the marking of Req_1 is 1, the location of $\mathcal{A}_1$ is l_0 and that in the initial valuation all $\mathcal{A}_1$ variables are null $[t=0, x_1=0, x_2=0]$. The initial state of σ does not change until the first event (Arr_1) occurs after 0.5 time units yielding a change to state $(2, 1, 1, 0, 0)$, i.e. an increment in the tokens in Req_1 (the number of class 1 processes waiting increases by 1). While σ is sojourning in the initial state the automaton spends time in l_0 hence x_1 increased is value with rate $Req_1=1$ hence at $t = 0.5$, its value is $x_1 = 0.5$, which corresponds to the transition to the second configuration of $\mathcal{M}_{Req_1} \times \mathcal{A}_1$, that is $(2, l_0, [t = 0.5, x_1 = 0.5, x_2 = 0])$. Notice that as long as $t < 4$ the autonomous transition $l_0 \xrightarrow{\sharp, t=4, x_1/=4} l_1$ of $\mathcal{A}_1$ is not enabled hence the synchronisation cannot stop. The next transition in σ (again Arr_1) happens after further 1.5 time units, at that moment the clock is $t = 2$, while x_1 continued to grow but at a greater rate $\dot{x_1} = 2$ as for $t \in [0.5, 2]$ the marking of place is $Req_1 = 2$. Therefore the occurrence of the second Arr_1 events yields the third configuration of $\mathcal{M}_{Req_1} \times \mathcal{A}_1$, that is $(3, l_0, [t = 2, x_1 = 3.5, x_2 = 0])$. The following event in σ is the occurrence of the immediate transition $Start_2$ representing the fact that a class 2 process starts to occupy the shared resource: notice that the occurrence of $Start_2$ although yields a synchronisation with the automaton does not actually change the configuration of the product process as neither the location nor the variables of $\mathcal{A}_1$ are affected by the occurrence of $Start_2$. The synchronisation goes on in a similar fashion and ends with an autonomous transition through which the final location l_1 of $\mathcal{A}_1$ is reached from configuration $(2, l_0, [3.5, 7.5, 1])$ after a delay of 0.5 units, which corresponds with the clock reaching the value $t = 4$ hence fulfilling the constraint for the $l_0 \xrightarrow{\sharp, t=4, x_1/=4} l_1$ of $\mathcal{A}_1$ edge. By traversing such edge x_1 is updated to $x_1 = 8/4 = 2$ which indeed corresponds with the mean number of class 1 process that have been waiting to access the resource in

[4] we represent states of $\mathcal{M}$ as 5-tuples with the following correspondence with the places name $(Req_1, Req_2, Free, Serv_1, Serv_2)$

the time interval $[0, 4]$.

Remark 3 (Almost-sure termination of the synchronisation process)**.** A necessary condition for establishing a statistical model checking framework based on the HASL formalism is the guarantee that a stochastic simulation algorithm that (based on the operational semantics outlined in Definition 2.3) samples paths from the synchronised $\mathcal{M} \times \mathcal{A}$ product process always terminates. Generally speaking HASL model checking consists of an iterative procedure for samping infinite path of $\mathcal{M} \times \mathcal{A}$ however the quantity Z of interest (see Section 2.2.2) is evaluated on the finite prefix that ends when the path reaches an absorbing state. In the general case, it would be possible that an infinite path does not admit such a prefix. Here we assume that given a DESP $\mathcal{M}$ and a HASL formula $(\mathcal{A}, Z)$, with probability 1, the synchronising path generated by a random execution path of $\mathcal{M}$ reaches an absorbing configuration. This semantical assumption can be ensured by structural properties of $\mathcal{A}$ and/or $\mathcal{M}$. For instance, the time bounded `Until` of CSL guarantees this property. As a second example, the time unbounded `Until` of CSL also guarantees this property when applied on finite CTMCs where all terminal strongly connected components of the chain include a state that either fulfils the target sub-formula of the `Until` operator or falsifies the continuing sub formula. This (still open) issue is also addressed in [SVA05a, HJB+10].

2.2.2 HASL expressions

The second component of an HASL specification is an expression, denoted Z, given by grammar (2.1). Z is associated to a LHA $\mathcal{A}$ and expresses the target measure whose confidence interval should be estimated based on the paths accepted by $\mathcal{A}$.

$$\begin{aligned} Z &::= \mathbb{E}(Y) \mid Z + Z \mid Z \times Z \mid Pdist \\ Pdist &::= PDF(Y, step, start, stop)) \mid CDF(Y, step, start, stop) \mid PROB() \\ Y &::= c \mid Y + Y \mid Y \times Y \mid Y/Y \mid last(y) \mid min(y) \mid max(y) \\ y &::= c \mid x \mid y + y \mid y \times y \mid y/y \end{aligned} \tag{2.1}$$

There are two main types of expressions Z: $\mathbb{E}(Y)$ (where $\mathbb{E}$[5] indicates *mean value of*, i.e. the first moment of) and $Pdist$ (indicating a probability distribution or probability value expression). Y represent a random variable built on top of algebraic combination of some *path* operators applied to an algebraic expression composed of LHA variables y, i.e. $last(y)$ (i.e. the last value that y has at the end of an accepted path, $min(y)(resp.\max(y))$, the min (resp. max) value of y along an accepted path. Conversely Z expressions of $Pdist$ type include $PDF(Y, step, start, stop)$, which allows for estimating the PDF of random variable Y

[5]Notice that within the COSMOS tool $\mathbb{E}$ is actually denoted AVG, therefore HASL expressions may equivalently be formulated using either $\mathbb{E}(Y)$ or $AVG(Y)$.

computed by discretisation of the support set $[start, stop] \subset \mathbb{R}_{\geq 0}$ in $(stop - start)/step$ sub-intervals of size $step$ and similarly $CDF(Y, step, start, stop)$. Finally expression $PROB()$ allows for estimating the probability that a path is accepted, otherwise said $PROB()$ is used to estimating the probability of the paths event set represented by the considered automaton $\mathcal{A}$. Notice that one can then define sophisticated probability measures by considering specific conditions as constraints of the edge(s) leading to an accepting location of $\mathcal{A}$.

Notice that since Z can be a composed expression obtained by algebraic operators (e.g. $Z + Z$, $Z \times Z$) built on top of the first moment of Y ($AVG(Y)$) it is possible to take into account diverse significant characteristics of Y (apart from its expectation) as the quantity to be estimated, including, for example, the variance $Var[Y] \equiv \mathbb{E}[Y^2] - \mathbb{E}[Y]^2$ and the covariance $Covar[Y_1, Y_2] \equiv \mathbb{E}[Y_1 Y_2] - \mathbb{E}[Y_1]\mathbb{E}[Y_2]$. Furthermore we point out that for efficiency reasons, in the implementation of the COSMOS software tool (Section 2.4), we have considered a restricted version of grammar (2.1), where products and quotients of data variables (e.g. $x_1 \times x_2$ and x_1/x_2) are allowed only within the scope of the $LAST$ operator (*i.e.* not with MIN, MAX, INT). Indeed, allowing products and quotients as arguments of path operators such as MAX or MIN requires the solution of a linear programming problem during the generation of a synchronised $\mathcal{M} \times \mathcal{A}$ path which, although feasible, would considerably affect the computation time.

Example 5 (HASL expressions)**.** Referring to the automata $\mathcal{A}_1$ and $\mathcal{A}_2$ of Example 2.3 and $\mathcal{A}_{3a}$ and $\mathcal{A}_{3b}$ of Example 3 let us consider a few examples of complete HASL specifications together with their informal description.

- $\phi_{1a} \equiv (\mathcal{A}_1, \mathbb{E}(last(x_1)))$: the mean number of class 1 processes waiting to access the shared resource within the time interval $[0, 4]$
- $\phi_{1b} \equiv (\mathcal{A}_1, \mathbb{E}(max(x_2)))$: the max number of class 1 processes that got access the shared resource within the time interval $[0, 4]$
- $\phi_{2a} \equiv (\mathcal{A}_2, \mathbb{E}[LAST(x_1)/k)$: the upper bound on the waiting time of the first k instances of class 1 processes arrived in the system.
- $\phi_{2b} \equiv (\mathcal{A}_2, \mathbb{E}[LAST(x_1)/LAST(x_3))$: the upper bound on the waiting time of the first k instances of class 1 processes arrived in the system.
- $\phi_{3a} \equiv (\mathcal{A}_{3a}, \mathbb{E}[PROB())$: the probability that type 1 clients have used the shared resource longer than type 2's, given that k clients (of any kind) have used the shared resource.
- $\phi_{3b} \equiv (\mathcal{A}_{3b}, \mathbb{E}[PROB())$: the probability that type 2 clients have used the shared resource longer than type 1's, given that k clients (of any kind) have used the shared resource.

Remark 4 (Existence of expectation $\mathbb{E}(Y)$)**.** We emphasise that the (conditional) expectation of a path random variable is not always defined. There are two obvious necessary conditions on the synchronised product $(\mathcal{M} \times \mathcal{A})$: (1) almost surely the random execution ends either in a final state of the LHA or in the rejecting state, and (2) with positive probability the random execution ends in a final state of the LHA. However these conditions are not sufficient. Different restrictions on the path formula ensure the existence of expectations. For instance, when the formula only includes bounded data variables and the operator INT and the division are excluded, the expectation exists. Divisions may be allowed when the path expression is lower bounded by a positive value: for example w.r.t. formula ϕ_{2b}, $LAST(x_3)$ is lower bounded by $LAST(x_2)$ which is lower bounded by k. The existence of regeneration points of the synchronised product may also entail the existence of such expectations. We do not detail the numerous possible sufficient conditions for the expectation of a path expression Z to exist but for all applications discussed is this manuscrit the existence of such expectation can be easily shown.

2.3 Expressiveness of HASL

As arguably the most relevant feature of the HASL formalism is its expressiveness it is important to address how it compares with respect to related formalisms. In the remainder we briefly overview temporal logic based frameworks for the verification of stochastic models, referring to [BBD+15a] for a more in depth discussion.

2.3.1 Comparison with logics for numerical stochastic model checking

CSL and variants. A number of temporal logic formalisms have been introduced for model checking of CTMCs. These include the seminal Continuous Stochastic Logic (CSL), first introduced by Aziz *et al.* [ASSB00] and further extended by Baier *et al.* [BHHK03a]. CSL extended CTL *branching-time* state-based reachability reasoning to the realm of CTMCs, by 1) introducing densely time-bounded temporal operators, hence allowing for time-bounded reachability verification, and 2) by allowing for steady-state reasoning through the introduction of a dedicated steady-state modality. In an effort to extending model checking with performance analysis capabilities several variants of CSL have been introduced that operate on *reward enriched* Markov models, such as Baier *et al.* 's CSRL [BHHK00], that extends the CSL approach to Markov reward models, i.e. CTMCs with a single reward on states or Cloth *et al.* 's [CKKP05] that further extended that approach with impulse rewards on actions. Kwiatkowska *et al.* [KNP07b] then combined those advances by equipping the PRISM model checker [PRI] with complete state and transition reward-based analysis facilities which is achieved by giving the possibility to the modeler to enrich CTMCs with multiple state and/or transition rewards and to analyse them through dedicated *reward based operators* which allow for assessing their expected value either at given instant of time, at steady-state or cumulated over finite paths.

Another significant evolution of CSL is Baier *et al.* asCSL logic [BCH+07] through which the time-bounded `Until` of CSL is replaced by a regular expression with a time interval constraint. These path formulas can express elaborated functional requirements as in CTL* but the timing requirements are still limited to a single interval globally constraining the path execution.

Remark 5 (CSL expressiveness)**.** Despite the considerable effort to evolve CSL logic the expressiveness of all CSL-like formalisms is limited to state-based, singly time-bounded reachability reasoning, that is: with CSL we can only consider state conditions and at most a single time interval as constraining factors to characterise the paths whose probability measure we wish to account for.

Timed automata based CTMC model checking. In order to increase the expressiveness of CSL-like temporal reasoning timed-automata based model checking of CTMCs has been proposed. Haddad *et al.* introduced the overall idea with the logic CSL^{TA} [DHS09], whereby path formulae are defined by a single-clock deterministic timed automaton (DTA). This logic has been shown to be strictly more expressive than CSL and also more expressive than asCSL when restricted to path formulas. A generalisation of CSL^{TA}has been introduced by Chen *et al.* [CHKM09], deterministic timed automata with multiple clocks are considered and the probability for random paths of a CTMC to satisfy a formula is shown to be the least solution of a system of integral equations. The cost of this more expressive model is both a jump in the complexity as it requires to solve a system of partial differential equations, and a loss in guaranty on the error bound.

Remark 6 (Timed-automata temporal logic expressiveness)**.** The employment of (single or multiple-clock) timed-automata as paths specifiers increases the expressiveness of temporal reasoning as one can now not only combine state and event conditions but also consider multiple time-intervals as criteria to specify the paths of interest, therefore effectively moving beyond CSL expressiveness. However the proposed timed automata based model checking approaches do not provide any support for reward-based analysis as clocks being the only kind of variable supported by timed automata they do not allow to neither measure any reward nor consider cumulated rewards as criteria to characterise the paths of interest.

M(I)TL. Several logics based on linear temporal logic (LTL) have been introduced to consider timed properties, including Metric (Interval) Temporal logic in which the `Until` operator is equipped with a time interval. Chen *et al.* [CDM11] have designed procedures to approximately compute desired probabilities for time bounded verification, but with complexity issues. The question of stochastic model checking on (a sublogic of) M(I)TL properties, has also been tackled see e.g. [ZC13].

Remark 7 (Limited expressiveness for numerical model checking)**.** All of the above mentioned CSL and timed-automata logics have been designed so that numerical methods can be employed to assess the probability measure of a formula. This very constraint is at the basis of their limited expressive scope which has two aspects: first the targeted stochastic models are necessarily CTMCs; second the expressiveness of formulas is constrained by decidability/complexity issues. Furthermore the evolution of stochastic logics based on CTL seems to have followed two directions: one targeting *temporal reasoning* capability (in that respect the evolutionary pattern is: CSL $\rightarrow$ asCSL $\rightarrow$ CSL$^{\text{TA}}\rightarrow$ multi-clock DTA), the other targeting *performance evaluation* capability (evolutionary path: CSL $\rightarrow$ CSRL $\rightarrow$ CSRL+impulse rewards). A unifying approach is currently not available, thus, for example, one can calculate the probability for a CTMC to satisfy a sophisticated temporal condition expressed with a DTA, but cannot, assess performance evaluation queries at the same time (i.e. with the same formalism).

Hybrid automata based model checking. As HASL is inherently based on stochastic simulation, it naturally allows for releasing the constraints imposed by logics that rely on numerical solution of stochastic models and therefore broaden the class of target stochastic models to DESP which includes, but is not limited to, CTMCs. From an expressiveness point of view, the use of LHA, i.e. a generalisation of DTA, allows for generic variables, which include, but are not limited to, clock variables. That opens up to a much wider expressiveness as by combining real-valued, boolean, integer-valued variables one can now take into account sophisticated reward measure that can not only be estimated *per se* but, most importantly, can be used as selection criteria of the paths of interest. This means that sophisticated temporal conditions as well as elaborate performance measures of a model can be accounted for in a single HASL formula, rendering HASL a unified framework suitable for both model-checking and performance and dependability studies. Note that the nature of the (real-valued) expression Z (2.1) generalises the common approach of stochastic model checking where the outcome of verification is (an approximation of) the mean value of a certain measure (with CSL, asCSL, CSL$^{\text{TA}}$ and DTA a measure of probability).

More specifically it is also worth noting that the use of data variables and extended updates in the LHA enables to compute costs/rewards naturally. The rewards can be both on locations and on actions. First using an appropriate flow in each location of the LHA, possibly depending on the current state of the DESP we get "state rewards". Then, by considering the *update expressions* on the edges of the LHA, we can model sophisticated "action rewards" that can either be a constant, depend on the state of the DESP and/or depend on the values of the variables. Thus HASL extends the possibilities of CSRL (and extensions [LKKP05]). The extension does not only consist of the possibility to define multiple rewards (that can be handled, for example, through the reward-enriched version of CSL supported by the PRISM tool) but rather of their use. First several rewards can be used in the same formula, and last but not least these rewards have a more active role, as they can not only be evaluated at the end of the path, but they can also take an

important part in the selection of enabled transitions, hence of accepted paths. It is for example possible and easy to characterise the set of paths along which a reward reaches a given value and after that never goes below another value, a typical example that neither PRISM-CSL nor CSRL can handle.

Limitations of HASL. Finally, we briefly discuss two features that are available in the above mentioned stochastic logics but not in HASL. First HASL does not allow to properly model nesting of probabilistic operators. The key reason is that this nesting is meaningful only when an identification can be made between a state of the probabilistic system and a configuration (comprising the current time and the next scheduled events). While this identification was natural for Markov chains, it is not possible with DESP and general distributions that have no memoryless property, and therefore this operation has not been considered in HASL. Furthermore, even for Markovian systems, the complexity of the statistical method on formulas with nesting is quite high [YS06] as the verification time per state along a path is no longer constant.

A similar problem arises for the steady state operator. The existence of a steady state distribution raises theoretical problems, except for finite Markov chains. With HASL we allow for not only infinite state systems but also non Markovian behaviours. However, when the DESP has a regeneration point, various steady state properties can be computed considering the sub-execution between regeneration points.

In conclusion it is worth noting that these limitations are rather due to the verification method (statistical in our case) and to the expressiveness of the model (allowing non Markovian systems) than to a particular tool. All this information given, particularly concerning nested and steady state formulae, we can now state and prove our claim about the respective expressiveness of HASL, CSRL and CSL^{TA}:

Proposition 1. Given a non nested transient CSRL formula $P_{\bowtie q}\phi$ and a system described as a Markov Reward Model, it is possible to build an LHA to estimate the probability p for ϕ to hold, and then decide whether it fulfils the bound required (*i.e.* $p \bowtie q$ with $\bowtie \in \{<, >, \leq, \geq\}$).

Proof. See [BBD+15a]

Proposition 2. Given a non nested transient CSL^{TA} formula $P_{\bowtie q}\mathcal{A}(\phi_1, ..., \phi_n)$ and a system described as a Continuous Time Markov Chain, it is possible to build an LHA to estimate the probability p for an execution to be accepted by the DTA $\mathcal{A}(\phi_1, ..., \phi_n)$, and then decide whether it fulfills the bound required (*i.e.* $p \bowtie q$ with $\bowtie \in \{<, >, \leq, \geq\}$).

Proof. See [BBD+15a]

2.3.2 A couple of meaningful applications

The expressive power of the HASL formalism has been demonstrated through a number of applications in different domains, including that of biological modelling [BGGM14, BD15], that of performance analysis of manufacturing systems [BDD$^+$11d, BH21a, BH21b], that of verification of wireless network protocols [BBV19, MMB$^+$17], in medicine [BKMP15] and in autonomous vehicle safety analysis [BBDH17] (a list of publications related to HASL is available at [COS]). To illustrate HASL expressiveness below we briefly overview a couple of meaningful applications.

Analysis of gene expression models with stochastic delayed dynamics

Gene expression is a fundamental biological intra-cellular process by which proteins are synthesised from a *gene*, i.e., a sequence in the DNA. It consists of two main phases: *transcription*, i.e., the copying of a sequence in the DNA strand by an RNA polymerase (RNAp) into an RNA molecule, followed by *translation*, i.e., the process by which proteins are synthesised from the (transcribed) RNA sequence. The rate of expression of a gene is usually regulated at the stage of transcription, by repressor molecules that can bind to the operator sites (generally located at the promoter region of the gene) and then inhibit transcription initiation. Evidence suggests that this is a highly stochastic process since usually, the number of molecules involved, e.g. transcription factors and promoter regions, is very small, ranging from one to a few at a given moment, which motivates the application of stochastic modelling to the analysis of gene expression dynamics. Ribeiro *et al.* [RZK06] argued that the dynamics of chemical reactions involved in the gene expression process exhibits a *stochastic delayed* nature, meaning that the appearance of a reaction's product may be subject to a further (non Markovian) stochastic delay after the actual occurrence of the reaction which, in turns, is selected stochastically in a standard manner, i.e., through Gillespie's SSA [Gil77].

In [BMR12] we formally analysed the dynamics of a single gene reaction network taken from [RZK06] whose stochastic Petri net encoding is depicted in Figure 2.6. The six reactions are modelled by small subnet blocks consisting of few timed Exponential timed transitions (depicted as empty rectangles) and non-Markovian timed transitions (grey filled rectangles). Place *Pro* represent the single promoter area of the gene and a token in it indicates the promoter is not occupied. Expression begins with the initialisation reaction R_1, whereby an RNAp binds to a promoter (*Pro*), which remains unavailable for more reactions until reaction *transc* occurs. Following reaction *transc* both the promoter and the ribosome binding site *RBS* become available although the production of an RNA is not terminated until a further delay given by (the Gamma distributed) reaction *termin* (notice that the number of RNA molecules produced corresponds with the marking of place *RNA*). As soon as the *RBS* site of the RNA is completed (through reaction *transc*) the *translation process* begins by binding (reaction *transl*) of a ribosome molecule *Rib* to the *RBS*. This processes terminates only after further delays: a Gamma distributed timed

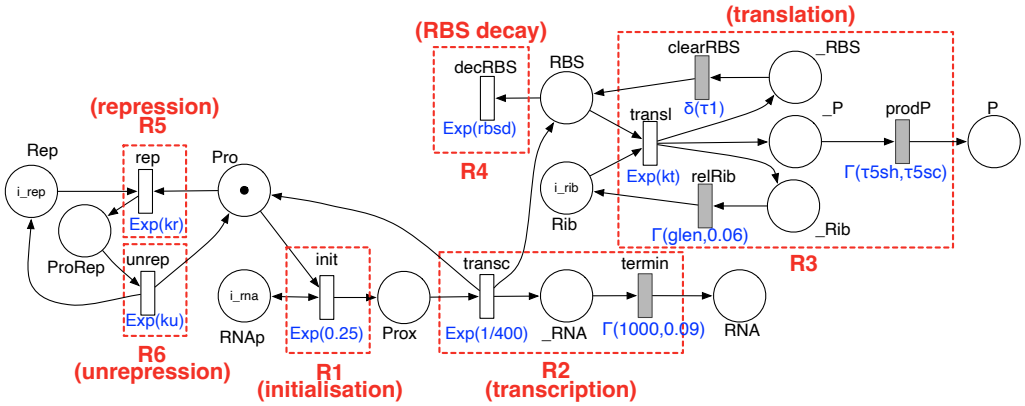


Figure 2.6: Stochastic Petri net model of the single-gene expression network

transition representing the release the *Rib* molecule from the the *RBS*, another Gamma distributed timed transition modelling the termination of translation hence the synthesis of the product proteins P and finally a Dirac distribution for freeing the *RBS*. The model is completed by few further Markovian transitions: one modelling the RBS decay (R_4) and a pair of transitions (R_5 and R_6) modeling the periodic occupation of the promoter by a repressor molecule which inhibits the whole expression process.

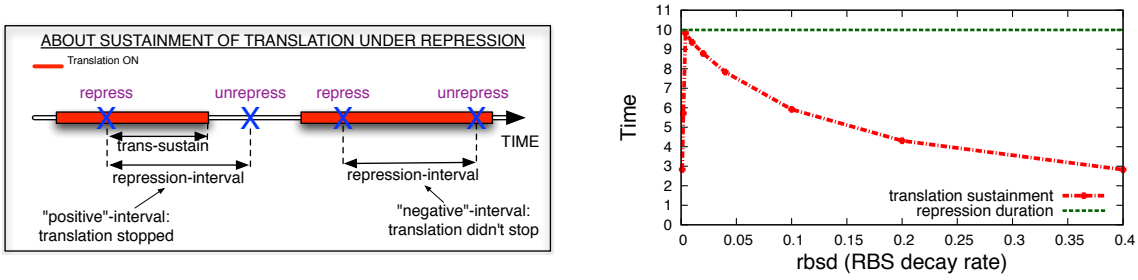


Figure 2.7: Sustainment of translation during repression intervals.

In order to analyse the dynamics of gene expression process we conceived a few properties which depend on non trivial random variables such as:

- ϕ_1: *"duration of translation process given that translation ends within a repression interval".*
- ϕ_2: *"probability of the number of translation events occurring in between two consecutive transcription events given at least N translations occur between two consecutive*

transcriptions".

Random variable ϕ_1 quantifies the inertia (sustainment) of the translation phase in terms of how long translation keep going since expression is turned off (through repression). The overall idea is illustrated in Figure 2.7 (left) where a *repression interval* is the time interval between the occurrence of a repression event (R_5) and the corresponding un-repression event (R_6) and a "positive" repression interval is one in which the translation process actually completely terminated (i.e. by releasing a P molecule and detaching the *Rib* molecule from the RBS) whereas a "negative" interval is one in which translation did not terminate. Figure 2.7 (right) shows the average value of ϕ_1 measured over a time-bounded horizon (through a dedicated HASL formula on COSMOS) in function of RBS decay rate.

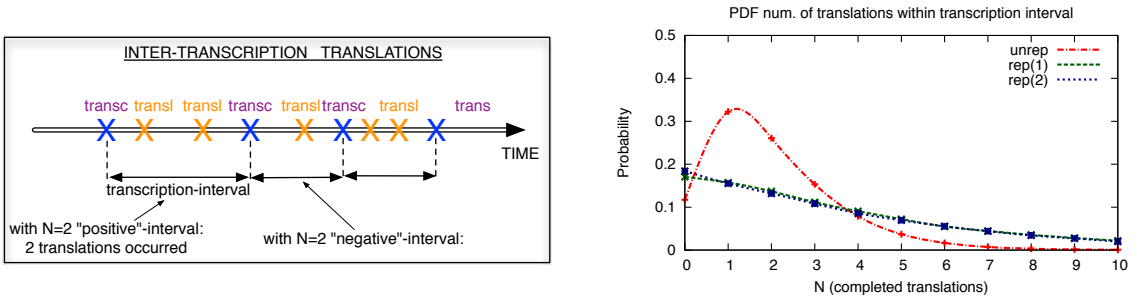


Figure 2.8: Completed translations between consecutive transcriptions.

On the other hand random variable ϕ_2 is meant as an indicator of the relative frequency of the transcription and translation processes. Figure 2.8 (left) illustrates intuitively the meaning of measures related to ϕ_2, where a "positive" interval is one in which at least $N = 2$ transcriptions event occurred as opposed to a "negative" interval is one in which less than N transcriptions occurred. Figure 2.7 (right) shows the value of ϕ_2 measured (through a dedicated HASL formula on COSMOS) in function of N and w.r.t. to unrepressed and differently repressed setting of the single-gene model.

We stress that ϕ_1 and ϕ_2 are meaningful witnesses of the expressive power of the HASL formalism: an equivalent formulation in terms of popular temporal logic formalisms such as, e.g., the reward-extension of CSL supported by the PRISM model checker, could hardly be obtained.

Analysis and computation of passage-time measures

In performance modelling the *passage time distribution* is a specific type of performance index particularly useful when reasoning about properties related with Service Level Agreements (SLA) or safety requirements. Generally speaking a passage time measure is con-

cerned with tracking the duration for a single instance of a process to complete, for example the time it takes for an order to be treated (archived) since it is placed on a selling platform (see model in Figure 2.9). In practice passage time measures can be formulated as the distribution of the time required to reach a *goal* state from any *entry* state possibly without hitting any *forbidden* or equivalently by matching the occurrence of a *start* with a corresponding *end* event. Notice that classical performance measures based on mean values, like the average response time, are not sufficient to estimate a passage time distribution, whose calculation requires dedicated methods and tools [Kul95, DHK04].

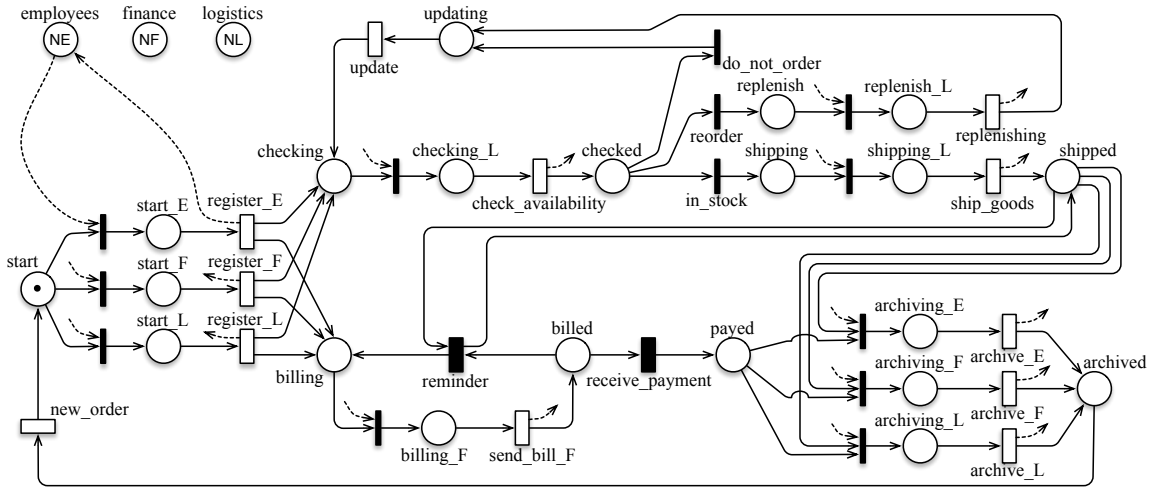


Figure 2.9: A NMGSPN model for an order-handling process.

From a modelling standpoint measuring of passage time distribution induces some difficulties, as it requires the ability to track the start and end of a specific process instance. In Petri nets terms this corresponds to the ability of tracking a specific token (e.g. the one corresponding to a specific order) as it flows through the Petri net up until it reaches the place that correspond to the end of the monitored process instance[6]. That lead to the extension of standard Petri net formalisms as is the case with the Tagged extension of GSPN [BDPF09] (TGSPN) where tokens can be *tagged* in order to follow their traversal of the net. The evaluation of passage time measures on TGSPN models boils down to a *transient analysis* problem of the underlying CTMC.

Alternatively passage times can be accounted for by combining a "standard" model with an automata-based *monitor* where the latter is used as a machinery to track the beginning and end of the activity of interest. Examples of this approach are the Extended Stochastic Probes (XSP [CG08], operating on PEPA models [Hil05]), Path Automata (PA,

[6]Notice that tracking tokens in a Petri net is not trivial as soon as several tokens, representing parallel instances of the monitored process, are permitted.

operating on Stochastic Activity Networks [OS99]) or Probe Automata (PrA [ABDF11] operating on GSPN models, where PrA are a kind of deterministic, untimed automata).

In [ABB+13] we introduced the use of the HASL formalism for the specification of passage time measures over NMGSPN models and showed that HASL allows for specifying and assessing more sophisticated passage time measures than those supported by Probe Automata, more specifically conditional passage times measures obtained by a combination of time, state and events constraints.

For example, referring to the order-handling model of Figure 2.9, we the introduced the following HASL-expressed conditional passage time measures which we assessed (see Figure 2.10) through a number of experiments run on the COSMOS tool (see Section 2.4):

- **Measure w1**: *the CDF of the passage-time for an ordered good to be delivered.*
- **Measure w2**: *the CDF of the passage-time for an ordered good to be delivered given that it was out-of-stock.*
- **Measure w3**: *the CDF of the passage-time for an ordered good to be delivered given that it is not out-of-stock and that the total delay for checking its availability and shipping it does not exceed K .*
- **Measure w4**: *the CDF of the passage-time for an ordered good to be delivered given that the total delay for reordering and updating the stock does not exceed K .*

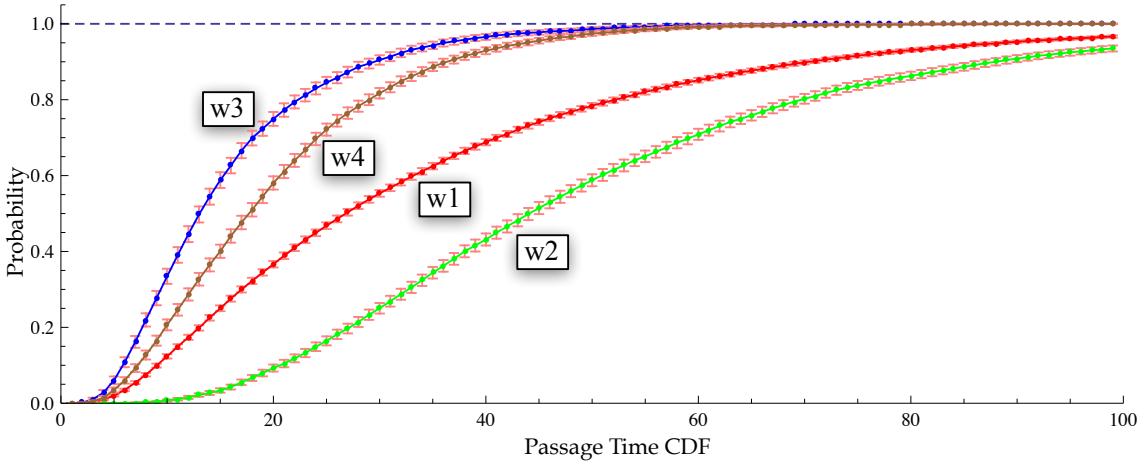


Figure 2.10: Passage times for the workflow properties w_1 to w_4 assessed over time interval $[0, 100]$.

It is worth stressing that apart from the increased expressiveness in [ABB+13] we also demonstrated the effectiveness of a general purpose performance analysis framework, such

as HASL model checking, in treating of a type of problem that otherwise is solved through customised formalisms. In fact assessment of passage time measures on a GSPN model through the Probe Automata approach is obtained through a quite cumbersome procedure which involves first constructing the tangible reachable graph of the GSPN, than composing it by considering the product with the Probe Automaton and finally solving the transient state distribution of the product model.

2.4 Software support: the Cosmos tool

In order to provide software support to the HASL formalism we developed Cosmos[7] [BDD+11a, COS, BB22] a software platform for statistical model checking of HASL formulae. Cosmos is based on the SMC scheme outlined in Figure 2.2 hence it implements the stochastic simulation algorithm for the synchronised product process ($\mathcal{M} \times \mathcal{A}$) and supports different statistical methods for estimating the target measure Z of a HASL formula $\phi \equiv (\mathcal{A}, Z)$, including confidence interval estimation as well as hypothesis testing.

2.4.1 A model driven code generation architecture

Cosmos is implemented in C++ and relies on the Boost libraries for random number generation functionalities. In an effort to obtain a performant tool Cosmos is designed according to a *model driven code generation* scheme (Figure 2.11): the inputs $\mathcal{M}$ (NMGSPN) and $\mathcal{A}$ (LHA) are parsed in order to generate an efficient customised C++ implementation of the simulation engine of the synchronized product $\mathcal{M} \times \mathcal{A}$. The generated code is linked

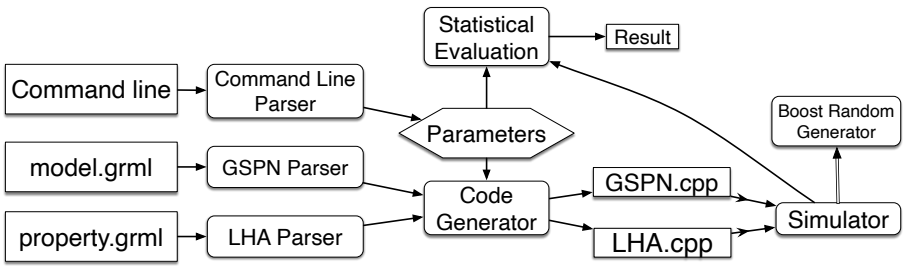


Figure 2.11: Cosmos's model-driven code generation scheme

with a library containing parts of the simulator that are independent of $\mathcal{M}$ and $\mathcal{A}$. This library contains the main function that determines the next event to occur by means of an event heap and the generated code. In addition a pseudo random generator computes delays for the new events that are put into the heap.

[7] COSMOS is an acronym of the french sentence "*Concept et Outils Statistiques pour le MOdèles Stochastiques*" whose english translation would sound like: "Tools and Concepts for Statistical analysis of Stochastic MOdels".

Parallel threads of execution. When an experiment is launched on a given model-automaton pair $(\mathcal{M}, \mathcal{A})$ COSMOS generate the C++ code for the corresponding SMC engine, compiles it and launches several threads of the resulting executable code in parallel that repeatedly generate trajectories and send back the evaluation of the formulas on these trajectories. COSMOS aggregates these evaluations and stops the simulation depending on the selected statistical method (see below). The code generation and compilation time is generally negligible compared to the simulation runtime.

2.4.2 COSMOS statistical engines

COSMOS is equipped with a number of statistical engines that are applied depending on the nature of the quantity to be estimated. Specifically *confidence interval* techniques, are used for *quantitative experiments*, i.e. experiments whose goal is to estimate the mean value of a given (real-valued) random variable, as opposed to *hypothesis testing* methods, which are applied for *qualitative experiments*, i.e. experiments whose goal is to establish how the mean value of random variable compares w.r.t. a given threshold.

Confidence interval methods. Confidence Interval (CI) methods works by collecting a number of samples $Z_1, \dots Z_n$ of an unknown random variable, in order to produce an estimate of an interval which is likely to contain the exact mean value μ_Z of the considered random variable (Z in the case of HASL). The accuracy of the estimate is controlled w.r.t. two complementary aspects: i) the *confidence level* ($\alpha \in (0, 1)$) which expresses how reliable the produced estimate is and ii) the *admitted error bound* (i.e. expressed as the width δ of the resulting interval). This means that if we repeatedly estimate the interval for a given θ we are guaranteed that the (possibly different) resulting intervals will contain the actual value of Z in a proportion corresponding to $(1 - \alpha)$. Given an n-sized sample $Z_1, \dots Z_n$ the general form of the $100(1 - \alpha)\%$ confidence-interval for the expected value μ_Z of Z, denoted $CI^{\alpha}_{\mu_Z}$, is:

$$CI^{\alpha}_{\mu_Z} = (PE_{\mu_Z}) \pm EB_Z$$

where PE_{μ_Z} is a Point Estimate of μ_Z and EB_α is the Error Bound, which corresponds to the semi-width of the CI interval, i.e. $EB_\alpha = \delta/2$. The sample mean $\overline{Z} = \frac{\sum_{i=1}^{n} Z_i}{n}$ is used as (*unbiased*) Point Estimator of μ_Z. The execution of CI experiment depends on three parameters: the sample size n, the confidence level α and the error bound δ. Generally speaking one can fix two of them and get the corresponding value of the remaining one in function of the previous two, however the actual choice of the two parameters to be fixed depend on the nature of variable to be estimated: in some case the sample size must be fixed a priori.

Static sample size estimation. With respect to CI estimation COSMOS distinguish between i) *indicator variables* (Bernoulli variables) used for evaluation of unknown prob-

abilities, ii) *bounded variables* used for evaluation of proportions, ratios, mean number of clients in a system with finite capacity and iii) *general variables* without any additional knowledge. For *indicator variables* and *bounded variables* COSMOS allows the user to chose 2 out of 3 of the CI parameters and automatically establishes the value of the remaining one in function of the fixed two parameters. For *general variables* however the sample size must be provided leaving free choice of one among the two remaining parameters. In general when the user fixes the sample size n and the confidence level α COSMOS compute the corresponding error bound based on the following statistical methods:

- Indicator variables: the error bound is obtained through the Clopper-Pearson method [CP34].
- Bounded variables the error bound is obtained through the Chernoff-Hoeffding method [Hoe63].
- General variables: an asymptotically correct error bound is obtained as approximation of a normal distribution with unknown mean and variance.

Dynamic sample size estimation. On the other hand, when estimating a general variable, the user may chose *a priori* the confidence α and the error bound δ and in this case cases COSMOS applies *sequntial CI scheme* through which the number of samples (i.e. trajectories) is established at runtime depending on a stopping condition (normally the convergence of the sample standard deviation). More specifically COSMOS supports the following CI sequential schemes: Chow and Robbins [CR65] Clopper-Pearson [CP34] and Chernoff-Hoeffdin [Hoe63].

Hypothesis testing. In case the user is interested in a qualitative formula, i.e. in establishing whether the average value of a given quantity Z fulfils a given contraint then COSMOS adopt a hypothesis testing approach. Specifically if the considered variable is a Bernoulli of unknown mean p, and given two probabilities $p_0 < p_1$ (the maximal probability of false positive and true negative results), COSMOS‘applies the Sequential Probability Ratio Test (SPRT) [Wal45] method (i.e. an optimal sequential test method) for deciding whether $p \geq p_0$ or $p \leq p_1$ holds.

Related Tools

Numerous tools that support statistical model checking have been developed over the years, some of which also support numerical model checking engines. These include COSMOS [BDD+11c], PLASMA [JLS12], PRISM [KNP11], UPPAAL [BDL+12], MARCIE [HRS13], STORM [HJK+22], APMC [HLP06], YMER [You05], MRMC [KZH+09] and VESTA [SVA05b]. An in depth comparison of these tools would certainly be an interesting task in which one would need to take into account several aspects including the

supported family of models, the solution engines, the expressive power of the property language but also how they perform w.r.t. some benchmark case studies. In an attempt to tackle such a goal in [BBD+15c] we provided a tentative comparative analysis limited to the YMER, PRISM, UPPAAL, PLASMA, MARCIE and COSMOS tools.

2.5 Perspectives

The HASL model checking framework effectively widen the ability of modellers to analyse the systems under study. By employing a rich automata-based formalism as a means to monitor a model's executions one may conceive sophisticated properties (by freely combining state-conditions, event-conditions and reward-based conditions) that allow one to capture complex aspects of a system's dynamics that could not be accounted for with "classical" temporal logic formalisms. A further positive consequence of such expressive power is that it yields a "full" *separation of concerns* meaning that modelling and properties conception are independent efforts in the sense that a model does not need to be enriched with complementary elements (rewards and/or dedicated monitor variables) which are otherwise necessary to overcome the deficits of less expressive formalisms and whose goal would be to register relevant statistics that one wants to account for during model checking.

If the expressive power is undoubtedly the strongest point in favour of the HASL formalism there are a number of factors worth considering for improving things. First the unsurprising cost of such an expressive power is the *unfriendliness* of the formalism: indeed the design of a hybrid automaton to capture a given temporal dynamics is a delicate and error prone task requiring expertise. In this respect it would be worth to come up with a workaround, in terms of a user friendly (syntax based) language to express a target property of the system under study (e.g. "what is the mean period of oscillation of species A?") which is suitable to automatic translation into a corresponding (HASL) hybrid automaton monitor. The conversion of temporal logic formulae into a corresponding (timed) automata formalism has been discussed by Donatelli, Sproston and Haddad [DHS07] as well as by ourselves [BBD+15a]. It is known that commonly used not-nested, (bounded/unbounded) reachability properties based on the `Until` temporal operator (and its variants) can straightforwardly be encoded in HASL terms while encoding of formulae involving nested temporal operators is more cumbersome. Generally speaking an in depth treatment of this matter is missing. It would therefore be worth figuring out to what extent one can outline an approach to automatically generate hybrid automaton monitors that represents relevant properties expressed through a more user friendly syntax.

Another aspect worth considering in order to evolve the HASL framework concerns the speed up of the statistical model checking engine. Although, differently from the memory constrained numerical model checking scheme, SMC allows for taking into account very large (even infinite state) models, the runtime of a SMC experiment can be huge given that sampling of a model's path through *exact* stochastic simulation can be very computational

intensive (depending on the stochasticity as well as of the size of the model). In recent time I got interested in investigating whether, within a SMC framework, one could take advantage of accelerated, *approximated* stochastic simulation algorithms, e.g., those based on the τ-leaping scheme [Gil01]. Roughly speaking that entails investigating what is the effect of the error induced by *approximated paths* on the (confidence-interval) estimation of the mean value of the property ϕ which is targeted by the SMC experiments. A preliminary prototype implementation developed as part of a student's research project seems to provide promising results in this respect.

Chapter 3

A Bayesian approach to parametric model checking

The automated verification of formally expressed properties by model checking have proved a successful achievement in theoretical computer science as demonstrated by numerous applications in heterogenous domains. One limitation of model checking though is that it does not come with natural means to analyse how the satisfaction of the considered property ϕ is affected by the model's parameters. Given that a system's model $\mathcal{M}$ normally depends on a set of parameters[1] $\theta = (\theta_1, \ldots, \theta_n) \in \mathbb{R}^n$ what model checking does is to establish whether a given model's instance $\mathcal{M}_\theta$ (i.e. the instance of $\mathcal{M}$ obtained by replacing the parameters with the values $\theta \in \mathbb{R}^n$) complies with ϕ, denoted $\mathcal{M}_\theta \models \phi$. A trivial approach to study how the satisfaction of ϕ is affected by parameters $\theta \in \mathbb{R}^n$ would be to systematically run a model checking experiment for every possible instance $\mathcal{M}_\theta$. Given the parameter space is normally dense such a *brute force* exhaustive approach would hardly be feasible other than being very inefficient given the computational cost for iterating single model checking experiments $\mathcal{M}_\theta \models \phi$, $\forall \theta \in \mathbb{R}^n$.

Here we are interested in *parametric model checking* of continuous-time probabilistic systems (particularly Markov chains) and therefore parameters correspond with the rates of the probability distributions that characterise the random delays of state transitions. The relevant question we wish to answer is: given a stochastic model with parametric rates for which sets of rate values a given temporal logic specification is satisfied? There are two families of approaches to answer such question: those that rely on *numerical methods* as opposed to those that rely on *statistical* methods.

Numerical approaches. Numerical approaches for model checking parametric CTMCs

[1] Example of which may be, the rate at which clients arrive in a given queue of a queueing system, or the maximum size of a mailbox in a message receiving server, or the probability at which a production machine in a manufacturing system breaks down in the next time unit, or yet the rate at which a given protein is synthesised when its corresponding gene is not inhibited.

rely on basic findings in CTMC model checking [BHHK03a], that is: the verification of time-bounded CSL specifications against a CTMC instance can be reduced to a *transient-state* distribution problem whose solution, in turn, is approximated through *uniformisation*. To deal with parametric CTMC a parametrised version of uniformisation has been proposed leading to algorithms for *parameter synthesis*. Initially limited to *qualitative* non-nested CSL specifications [HKM08], (i.e. boolean valued formulae of the form $P_{\sim p}[\phi]$ where ϕ is a non-nested path formulae and $\sim \in \{<, \leq, \geq, >\}$, $p \in [0, 1]$) they have then been extended to the entire CSL spectrum including *quantitative* formulae (i.e. formulae of the form $P_{=?}[\phi]$ whose verification requires assessing the probability that condition ϕ is met) [CDP+17]. The positive aspects of parametrised-uniformisation synthesis schemes is that they implement an exhaustive search of the parameter space yielding regions in which either a qualitative CSL formula is *guaranteed* to be satisfied or, alternatively, regions in which the probability of a quantitative CSL formula is ensured to fall within a bounded interval (of arbitrary width). The downside however is their computational cost[2] and hence their inability to scale up other than their limitation to the Markov chains realm.

Statistical approaches. Statistical approaches to parametric verification constitute a computationally cheaper alternative to numerical ones, the basic idea being to obtain an approximation of the *satisfaction probability* of ϕ through non-exhaustive exploration of the parameter space. The prominent contribution in this still relatively young area of research is that of smoothed model checking [BMS16] in which Bortolussi and Sanguinetti demonstrated that the satisfaction function of a Metric Interval Temporal Logic (MITL) formula against a parametric CTMC is (under mild conditions over the nature of the parameters) a smooth function of the CTMC parameters. This leads to algorithms where an arbitrarily precise approximation of the satisfaction function is derived via a Gaussian Process based on its smoothness.

Outline of contribution. In this chapter we present a statistical approach for approximating the *satisfaction probability* of temporal properties in function of a stochastic model's parameters. Differently from smoothed model checking our approach relies on Bayesian statistics, specifically on the Approximated Bayesian Computation (ABC) method, i.e., a well established method that given a set of *observations* of the modelled system and an initial *prior* distribution over the model's parameter space allows to approximate the *posterior* distribution by selecting parameters based on their *their distance* from the system's *observations*. Here we adapt ABC schemes by replacing the *observations* with a formal *machinery* (consisting in a linear hybrid automata of the HASL formalism) through which we assess *how far a set of parameters is from satisfying the considered temporal property*, hence letting ABC algorithms select only those parameters that match the desired behaviour. Since we use HASL linear hybrid automata as machinery to assess the distance of parameters we name Automaton-ABCthe parametric model checking adapta-

[2] in the general case parametrised uniformisation boils down to solving a nonlinear programming problem.

tion of ABC. Differently from standard ABC schemes, the automaton-ABC procedure is *observations free*, in that it does not require any dataset in order to drive the parameter selection process, however it relies on the existence of a notion of *distance* between the paths of a model instance $\mathcal{M}_\theta$ and the property ϕ which is the target of the parameter search. We demonstrate the effectiveness of the automaton-ABC scheme by introducing a formal definition of distance for a classical time-bounded reachability problems and we give the corresponding distance-monitor LHA automata. This contribution has been at the core of Mahmoud Bentriou PhD which I co-directed with Paul-Henry Cournède and whose results have been published in [BBC19, BBC21].

Structure of the chapter. The chapter is organised as follows: Section 3.1 gives an overview of necessary background material including the class of parametric Markov models which we consider, the property language we refer to, the classical ABC algorithms which we aim to extend and finally the Kernel estimation method which we resort to for reconstructing the satisfaction probability of a property φ from the posterior distribution issued by the Automaton-ABCalgorithm. In Section 3.2 we introduce the notion of satisfiability distance for reachability problems and introduce the novel Automaton-ABC framework based on such distance measure is developed while in Section 3.3 the effectiveness of the framework is demonstrated through a couple of case studies.

3.1 Preliminaries

3.1.1 Markov Population Models

Before looking at the basics of how the ABC method works we start by overviewing the class of models we are going to consider, namely that of (parametric) Markov population models (pMPM), a (parametrised) form of continuous-time Markov chain (CTMC) [Kul16, BC19] suitable for modelling of *population processes*, i.e. systems whose states represent the number of individuals of different *species* and whose transitions correspond to adding/removal of individuals. Since we target modelling of biological systems we also overview the Chemical Reaction Networks (CRNs) formalism as a means for expressing population models.

Definition 3.1 (Markov Population Model)**.** A Markov population model (MPM) for $n \in \mathbb{N}$ population species is a triple $\mathcal{M} = (S, Q, \pi_0)$ such that:

- $S \subseteq \mathbb{N}^n$ is a countable set of states
- $\mathbf{Q} : S \times S \to \mathbb{R}$: is the infinitesimal generator matrix (with $\mathbf{Q}(s_i, s_i) = -\sum_{j \neq i} \mathbf{Q}(s_i, s_j)$)
- $\pi_0 : S \to [0, 1]$ is the initial state probability distribution[3] (i.e. $\sum_{s \in S} \pi_0(s) = 1$).

[3] Whenever $\exists s_o \in S : \pi_0(s_0) = 1$ we use $\mathcal{M} = (S, Q, s_0)$ to denote an MPM.

Because of the *memoryless property* Markov models allows for a compact analytical expression of the probability for timed state transitions: given that the system is in state s at time t, the probability of observing a transition to state s' within time t' is $Pr((s,t) \to (s',t')) = \mathbf{P}(s,s') \cdot (1 - e^{-\mathbf{Q}(s,s')\cdot(t'-t)})$ where $E(s) = \sum_{s' \neq s} \mathbf{Q}(s,s')$ is the *exit rate* of state s, and $\mathbf{P}(s,s') = \mathbf{Q}(s,s')/E(s)$ is the (time-independent) probability of jumping from s to s'. Such basic property induces naturally a probability measure over the space of paths of a MPM model.

Whenever matrix Q is dependent on a d-dimensional vector of parameters $\theta \in \Theta \subseteq \mathbb{R}^d$ we talk of *parametric* MPM (pMPM).

Paths of a MPM. Since our final goal is to be able to estimate $Pr(\varphi|\mathcal{M}_\theta)$, i.e. the probability that a temporal property φ is satisfied in function of the parameters of a parametric MPM $\mathcal{M}_\theta$, we first recall that $Pr(\varphi|\mathcal{M}_\theta)$ corresponds with the probability of those *paths* of $\mathcal{M}_\theta$ that satisfy φ and then revise how, thanks to the memoryless property of MPMs, such probability enjoys a nice analytical expression leading to a compact a measure of probability[4] on the set of paths [BHHK03b].

A path of an MPM model $\mathcal{M}_\theta$ is a (possibly infinite) sequence $\sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{n-1}} s_n \ldots$, with $t_i \in \mathbb{R}_{>0}$ being the sojourn-time in state $s_i \in S$. We denote $Path_{\mathcal{M}_\theta}$ the set of all possible paths of $\mathcal{M}_\theta$ originating in the initial state s_0. For $\sigma \in Path_{\mathcal{M}_\theta}$, $i \in \mathbb{N}$ and $t \in \mathbb{R}_{>0}$, we denote $\sigma[i] = s_i$ the i-th state of σ, $\delta(\sigma,i) = t_i$ the sojourn-time of σ in the i-th state, σ_i the suffix of σ starting at state $\sigma[i]$, $T_k = \sum_{i=0}^{k} \delta(\sigma,i)$ the sum of the sojourn times up to state k, $\sigma@t$ the state of σ at time t and $\sigma[t\rangle$ the t-shifted suffix of σ, i.e. the suffix of σ that starts at time t. Formally $\sigma[t\rangle = \sigma[k+1] \xrightarrow{t-T_{k+1}} \sigma_k$ where k is the greatest index such that $T_k \leq t$. For example, for $\sigma = s_0 \xrightarrow{0.25} s_1 \xrightarrow{0.5} s_2 \xrightarrow{0.15} s_3 \xrightarrow{1} \ldots$ we have $\sigma[1] = s_1$, $\delta(\sigma,2) = 0.15$, $T_1 = 0.75$, $T_2 = 0.9$ and $\sigma[0.8\rangle = s_2 \xrightarrow{0.1} s_3 \xrightarrow{1} \ldots$ and $\sigma[1.5\rangle = s_3 \xrightarrow{0.4} \ldots$. Notice that trajectories of an MPM are *càdlàg* (i.e. step) functions of time. In order to indicate the reaction event that yielded a transition of the path of a CRN model, we sometimes adopt the following notation $\sigma = s_0 \xrightarrow[R_{1j}]{0.25} s_1 \xrightarrow[R_{2j}]{0.5} s_2 \xrightarrow[R_{3j}]{0.15} s_3 \xrightarrow[R_{4j}]{1} \ldots$, where R_{ij} indicates that reaction R_j occurred on the i-th transition of the path.

Paths probability space. For $s_0, s_1, \ldots, s_k$ a sequence of states of $\mathcal{M}_\theta$ such that $\mathbf{Q}(s_i, s_{i+1}) > 0$ $(0 \leq i < k)$ and $I_0, \ldots, I_{k-1}$ a sequence of non-empty time intervals in $\mathbb{R}_{\geq 0}$, we let $C(s_0, I_0, s_1, \ldots, I_{k-1}, s_k)$ be the *cylinder set* consisting of all paths $\sigma \in Path_{\mathcal{M}}(\mathcal{M})$ such that $\sigma[i] = s_i$ $(i \leq k)$ and $\delta(\sigma,i) \in I_i$ $(i < k)$. Furthermore we let $\mathcal{F}(Path_{\mathcal{M}})$ denote the smallest σ algebra containing all sets $C(s_0, I_0, s_1, \ldots, I_{k-1}, s_k)$. The probability measure on

[4] Notice that the notion of probability measure for paths of an MPM naturally extends to parametric MPMs which we target in the remainder.

$\mathcal{F}(Path_{\mathcal{M}})$ is inductively defined by:

$$\Pr_{\mathcal{M}}(C(s_0, I_0, \ldots, I_{k-1}, s_k)) = \begin{cases} 1, & \text{if } k = 0 \\ \mathbf{P}(s_{k-1}, s_k) \cdot (e^{-E(s_{k-1})\cdot t} - e^{-E(s_{k-1})\cdot t'}) \cdot & \\ \cdot \Pr_{\mathcal{M}}(C(s_0, I_0, \ldots, I_{k-2}, s_{k-1})), & \text{otherwise} \end{cases} \tag{3.1}$$

where $t = inf(I_{k-1})$ and $t' = sup(I_{k-1})$. In essence $\Pr_{\mathcal{M}}$ states that for a MPM $\mathcal{M}$ the probability of a path is given by the product of the probability to observe each constituent transitions $s_i \to s_{i+1}$ with a delay that falls in the corresponding binding interval I_i. In terms of vocabulary a subset of trajectories of $Path_{\mathcal{M}}(\mathcal{M})$ may be referred to as an event of $\mathcal{M}$ and its probability is given by $\Pr_{\mathcal{M}}$. In the realm of probabilistic model checking, temporal logic languages provide the modeller with a powerful language for characterising relevant events of an MPM model in terms of *formulae* (i.e. formal properties). The probability that a temporal logic property φ is satisfied by an MPM model $\mathcal{M}$ is defined in terms of the probability measure $\Pr_{\mathcal{M}}$ (see Definition 3.5).

In the remainder we consider Chemical Reaction Networks (CRNs) as a formalism for expressing population models.

Definition 3.2 (Chemical Reaction Network)**.** A (parametric) chemical reaction network (pCRN) with n species and m reaction channels is a triple $\mathcal{N}_\theta = (\mathcal{X}_n, \mathcal{R}_m, \theta, \mathbf{X^0})$ defined as follows:

- $\mathcal{X}_n = \{X_1, \ldots, X_n\}$ is a set of species
- $\mathcal{R}_m = \{R_1, \ldots, R_m\}$ is a set of reaction channels where each R_j ($j \in \{1, \ldots, m\}$) is characterised by an equation with the following form:

  $$R_j : \sum_{i=1}^{n} a_{ij}^{-} X_i \xrightarrow{k_j} \sum_{i=1}^{n} a_{ij}^{+} X_i$$

  where $a_{ij}^{-}, a_{ij}^{+} \in \mathbb{N}$ are the stoichiometric coefficients of the reaction's *reactants*, respectively, *products*. Furthermore R_j is characterised by a pair $R_j : (\nu_j, \eta_j)$ with

  - $\boldsymbol{\nu_j} = [\nu_{1j}, \ldots, \nu_{nj}]$ the change vector,
  - $\eta_j : \mathbb{N}^n \times \Theta \to \mathbb{R}_{\geq 0}$ is the propensity function of R_j.
- $\theta = [\theta_1, \ldots, \theta_d]$ is a d-dimensional vector of parameters affecting the kinetic rate of the reaction channels, with $\theta \in \Theta \subset \mathbb{R}^d$.
- $\mathbf{X^0} \in \mathbb{N}^n$ is the initial state

Although in the literature CRNs are often inherently mapped on their continuous-deterministic semantics (i.e., a system of differential equations) here we focus on their discrete-stochastic semantics, hence we assume the dynamics of a CRN to correspond with a (*càdlàg*) step-function governed by its reactions channels R_j. Assuming the system is in state $\mathbf{X} \in \mathbb{N}^n$ at time $t \in \mathbb{R}_{\geq 0}$ reaction $R_j : (\nu_j, \eta_j)$ may occur, moving the system to state $\mathbf{X}' = \mathbf{X} + \nu_j$, at time $t' > t$, with the delay $t' - t$ which is stochastically dependent on both the current state $\mathbf{X}$ and the actual value of the parameters θ.

Remark. For the sake of simplicity in our framework we assume reactions to obey the mass-action law. That means that the propensity functions are proportional to the product of the non-null stoichiometric coefficients of a reaction's reactants.

Definition 3.3 (pMPM semantics of a CRN)**.** A pMPM model $\mathcal{M}_\theta = (S, Q_\theta, \pi_0)$ of a CRN $\mathcal{N}_\theta = (\mathcal{X}_n, \mathcal{R}_m, \theta)$ is defined as follows:

- $S \subseteq \mathbb{N}^N$ is a countable set of states whose elements are vectors $\mathbf{X} = [X_1, \ldots, X_n] \in S$ where X_i is the population of the i-th species.
- $\mathbf{Q}_\theta : S \times S \to \mathbb{R}$: is the infinitesimal generator matrix whose entries are defined as:

$$\mathbf{Q}_\theta(\mathbf{X}, \mathbf{Y}) = \begin{cases} \sum_{\{R_j | \mathbf{X} + \nu_j = \mathbf{Y}\}} \eta_j(\mathbf{X}, \theta), & \text{if } \mathbf{X} \neq \mathbf{Y} \\ -\sum_{\mathbf{Z} \neq \mathbf{X}} \mathbf{Q}(\mathbf{X}, \mathbf{Z}), & \text{otherwise} \end{cases}$$

- $\pi_0 : S \to [0, 1]$ is defined as $\pi_0(\mathbf{X^0}) = 1$

Notice that by construction, the non-diagonal entries of $\mathbf{Q}_\theta$ are given by the sum of the propensities of those reactions whose occurrence leads the CRN to move from state $\mathbf{X}$ to state $\mathbf{Y}$. This is in line with the semantics of Markovian events, according to which the distribution of the minimum between a set of concurrent exponentially distributed reactions is itself exponentially distributed with rate given by the sum of the rates of the racing events.

Example 6 (CRN of infection spreading)**.** As a first example of CRN let us consider the SIR compartmental model [KM27], which describes the spread of infectious disease among a constant population. The CRN for the SIR is defined as $\mathcal{N}_{\text{SIR}} = (\{S, I, R\}, \{R_1, R_2\}, \{k_i, k_r\})$ where species S represents the susceptible individuals, I the infected and R the recovered ones. The system's dynamics is given by two reactions channels encoded by chemical equations (3.2).

$$R_1 : S + I \xrightarrow{k_i} 2I \qquad R_2 : I \xrightarrow{k_r} R \tag{3.2}$$

Reaction R_1 describes the infection step: a susceptible person meets an infected person and gets infected. Reaction R_2 models the recovering step: infected may become immune from the disease. The parameter vector of the model is $\theta = (k_i, k_r)$. The CRN of the

SIR yields a finite-state MPM with the following kinds of state-dependent transitions. For $\mathbf{X} = (\mathbf{X}_S, \mathbf{X}_I, \mathbf{X}_R)$ a state such that $\mathbf{X}_S > 0 \wedge \mathbf{X}_I > 0$ two kinds of transitions are possible, i.e. $\mathbf{Q}_\theta((\mathbf{X}_S, \mathbf{X}_I, \mathbf{X}_R), (\mathbf{X}_S - 1, \mathbf{X}_I + 1, \mathbf{X}_R)) = \mathbf{X}_S \cdot \mathbf{X}_I \cdot k_i$ and $\mathbf{Q}_\theta((\mathbf{X}_S, \mathbf{X}_I, \mathbf{X}_R), (\mathbf{X}_S, \mathbf{X}_I - 1, \mathbf{X}_R + 1)) = \mathbf{X}_I \cdot k_r$. For states such that $\mathbf{X}_S = 0 \wedge \mathbf{X}_I > 0$ only one transition is possible i.e. $\mathbf{Q}_\theta((\mathbf{X}_S, \mathbf{X}_I, \mathbf{X}_R), (\mathbf{X}_S, \mathbf{X}_I - 1, \mathbf{X}_R + 1)) = \mathbf{X}_I \cdot k_r$, whereas any state such that $\mathbf{X}_I = 0$ is absorbing.

Regions of an MPM. In the remainder, we refer to the notion of region associated with an MPM. A *region*, respectively a *time-bounded region*, of an n-dimensional MPM is any subset of $\mathbb{N}^n$, respectively $\mathbb{N}^n \times \mathbb{R}_{\geq 0}$, characterised by a collection of hyper-rectangles of dimension no larger than n. A region is *elementary* if it is characterised by a single hyper-rectangle. For example for a bi-dimensional MPM with state space $\mathbf{X} = \{X_1, X_2\}$, $R_1 \equiv [[1, 2]] \times \mathbb{N}$ is an elementary region (X_1 in $[[1, 2]]$ while X_2 is unbounded), $R_2 \equiv [[0, 3]] \cup [[5, 8]] \times [[5, \infty[[$ is a non-elementary region (X_1 is either in $[[0, 3]]$ or $[[5, 8]]$, X_2 is larger than 5), whereas $TR_1 \equiv ([[1, 2]] \times \mathbb{N}) \times [0.2, 1.41]$ is an elementary time-bounded region (similar to R_1, but with the supplemental condition that the time is in $[0.2, 1.41]$).

3.1.2 Linear-time temporal properties

To express properties of a MPM we refer to the Metric Interval Temporal Logic (MITL [MN04]), i.e. a linear-time temporal logic which allows for stating time-bounded reachability problems by combining state-conditions (expressed through inequalities on state variables) through classic time-bounded temporal operators. We choose MITL in order to formally set up reachability problems, however since we eventually employ hybrid automata as *meters* to measures the *satisfiability distance* of a MPM instance from a MITL formula, in practice this gives us the possibility to trespass the MITL expressiveness (e.g. by analysing oscillations properties).

MITL syntax. MITL formulae are terms of the following grammar:

$$\varphi \ ::= \ \top \mid \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \ \mathbf{U}^{[t_1, t_2]} \ \varphi_2$$

where $\top$ stands for the `true` formula, μ denotes an atomic proposition (i.e. an inequality built on top of model's state-variables), $\neg$ and $\wedge$ are the basic negation and conjunction connectives of propositional logic and $\mathbf{U}^{[t_1, t_2]}$ is the time-bounded *until* temporal operator with $[t_1, t_2] \subseteq \mathbb{R}_{\geq 0}$ being the bounding interval.

We consider the truth of an MITL formula to be established w.r.t. to a path σ of an MPM model through a so-called *satisfaction* relationship, denoted $\models$.

MITL semantics for temporal formulae. For $\sigma \in Path_{\mathcal{M}_\theta}$ a path of MPM model $\mathcal{M}_\theta$, $t \in \mathbb{R}_{\geq 0}$ a time instant the satisfaction relation $\models$ of MITL temporal formula is defined as follows:

$$
\begin{array}{lcl}
(\sigma,t) \models \top & \Leftrightarrow & true \\
(\sigma,t) \models \mu & \Leftrightarrow & \sigma@t \models \mu \\
(\sigma,t) \models \neg\varphi & \Leftrightarrow & (\sigma,t) \not\models \varphi \\
(\sigma,t) \models \varphi_1 \wedge \varphi_2 & \Leftrightarrow & (\sigma,t) \models \varphi_1 \text{ and } (\sigma[t,t) \models \varphi_2 \\
(\sigma,t) \models \varphi_1 \mathbf{U}^{[t_1,t_2]} \varphi_2 & \Leftrightarrow & \exists t' \in [t+t_1, t+t_2] : \\
 & & (\sigma,t')\rangle \models \varphi_2 \wedge \forall t'' \in [t,t'], \\
 & & (\sigma,t'') \models \varphi_1
\end{array}
$$

Intuitively MITL semantics states that an atomic proposition μ is satisfied by a path σ as of time t if the state condition μ is satisfied in the state in which σ is at time t ($\sigma@t \models \mu$). On the other hand, a time-bounded until formula $\varphi_1 \ \mathbf{U}^{[t_1,t_2]} \ \varphi_2$ is satisfied by σ as of time t if and only if φ_2 is satisfied by σ as of a future time instant t' which is no further than the time-bounding interval, (i.e. $t' \in [t+t_1, t+t_2]$) while φ_1 is sustainedly satisfied beforehand (i.e. $\forall t'' \in [t,t']$).

As usual, we consider two derivations of the time-bounded until operator: the time-bounded *eventuality* $\mathbf{F}^{[t_1,t_2]}\varphi \equiv \top \ \mathbf{U}^{[t_1,t_2]}\varphi$, which stands for *"at some point within* $[t_1,t_2]$ φ *is satisfied"* and the time-bounded *globally* $\mathbf{G}^{[t_1,t_2]}\varphi \equiv \neg\mathbf{F}^{[t_1,t_2]}\neg\varphi$ which stands for *"*φ *is always satisfied within* $[t_1,t_2]$*"*. In the remainder we assume that a path $\sigma \in Path_{\mathcal{M}_\theta}$ satisfies an MITL formula φ, denoted $\sigma \models \varphi$, if it does so starting from $t=0$, i.e. $\sigma \models \varphi \Longleftrightarrow \sigma[0\rangle\varphi$. Furthermore, we restrict our focus to the non-nested fragment of MITL, i.e., we consider only formulae such that the operands of a temporal modality are boolean combinations of atomic propositions μ. While bearing a definite limitation in terms of expressiveness, this constraint still allows us to treat the most common reachability problems.

MITL formulae and MPM regions. In the remainder we refer to the notion of untimed/time-bounded reachability regions associated with a MITL propositional formula. Specifically we point out that MITL propositional formulae induce *untimed* regions over the state space of an MPM. For example, for a bi-dimensional MPM the formula $\mu_1 \equiv x_1 \geq 1 \wedge x_1 \leq 2$ induces the region $R_1 \equiv [[1,2]] \times \mathbb{N}$ while $\mu_2 \equiv [(x_1 \leq 3) \vee (x_1 \geq 5 \wedge x_1 \leq 8)] \wedge x_2 > 4$ induces the region $R_2 \equiv [[0,3]] \cup [5,8] \times [[5,\infty[$. By a slight abuse of vocabulary, we say that two formulae μ_1, μ_2 are *disjoint* if the corresponding regions are. In the remainder, we assume regions are characterised by MITL propositional formulae in disjunctive normal form (DNF).

On the other hand we may associate a time-bounding interval to a MITL propositional formula and in this case we talk of time-bounded reachability region.

Definition 3.4 (Time-bounded reachability region)**.** Given an n-dimensional MPM population model $\mathcal{M}_\theta$ with state-space $S \subseteq \mathbb{N}^n$, a propositional formula μ and a time-interval $I \subseteq \mathbb{R}_{\geq 0}$ we define $S^I_\mu \subset S \times \mathbb{R}_{\geq 0}$ the time-bounded reachability as:

$$S^I_\mu = \{(s,t) \in S \times \mathbb{R}_{\geq 0} \mid s \models \mu \wedge t \in I\}$$

If $I = [0,\infty)$ then we omit the time interval and use $S_\mu = \cup_i^n \{s \in S \mid s \models \mu\}$

Note that depending on μ, we distinguish between *elementary reachability regions*, i.e. regions that consist of a single hyperrectangle (e.g. $\mu \equiv x_1 > 2 \wedge x_1 \leq 4$), as opposed to non-elementary regions, i.e. regions that consist of the union of several hyperrectangles (e.g. $\mu \equiv x_1 < 2 \vee x1 > 4$).

Definition 3.5 (Satisfaction probability of an MPM)**.** For $\mathcal{M}_\theta$ an MPM and φ an MITL formula, the satisfaction probability of φ w.r.t. $\mathcal{M}_\theta$ is defined as:

$$Pr(\varphi|\mathcal{M}_\theta) = \Pr_{\mathcal{M}_\theta}(\{(\sigma, 0) \models \varphi, \sigma \in Path_{\mathcal{M}_\theta}\})$$

where $\Pr_{\mathcal{M}_\theta}$ is the probability measure (3.1) induced by $\mathcal{M}_\theta$ over $Path_{\mathcal{M}_\theta}$.

Notice that assessing $Pr(\varphi|\mathcal{M}_\theta)$ is indeed the target of probabilistic model checking and that its value can be obtained either exactly through *numerical* model checkers [KNP02, DJKV17] or being approximated through *statistical* model checkers [You05, LST16, BDD+11a, SVA05a]. However since here we are interested in estimating how the satisfaction probability of φ changes with the model's parameters we introduce the notion of satisfaction probability function.

Definition 3.6 (Satisfaction probability function)**.** Let $(\mathcal{M}_\theta)_{\theta \in \Theta}$ be a parametric MPM and φ a 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.

We will see how through an adaptation of the ABC method we manage to obtain an approximation of f_φ for a parametric MPM model $\mathcal{M}_\theta$.

3.1.3 Bayesian inference: the ABC method

Given a parametric model $\mathcal{M}_\theta$ and an MITL formula φ, our goal is to estimate the satisfaction probability function f_θ, i.e. the function that characterises how the probability that φ is satisfied by $\mathcal{M}_\theta$ varies w.r.t. the parameter $\theta \in \Theta$.

Given a set of observations y_{obs} Bayesian inference is concerned with learning probability distributions over the parameters space Θ of a parametrised model $\mathcal{M}_\theta$. Relying on the Bayesian interpretation of probability, initial beliefs on the parameters expressed via a *prior distribution* π over Θ are progressively updated, via y_{obs}, based on the model's dynamics encoded by a *likelihood function* $p(y_{obs}|\theta)$ which expresses how probable y_{obs} is to be observed given the model's parameters $\theta \in \Theta$. The output of the inference procedure

consists of the *posterior distribution* $\pi(\theta|y_{obs})$ computed over Θ based on the observed data y_{obs} and which is defined by

$$\pi(\theta|y_{obs}) = \frac{p(y_{obs}|\theta)\pi(\theta)}{\int_{\theta'} p(y_{obs}|\theta')\pi(\theta')\, d\theta'}$$

For realistic models the likelihood function $p(y_{obs}|\theta)$, necessary to derive the posterior distribution, is too expensive to compute or even intractable. For this reason *likelihood-free* methods, such as Approximate Bayesian Computation [MPRR12, SFB18], have been introduced to obtain an approximation, denoted $\pi_{ABC,\epsilon}(\theta|y_{obs})$, of the posterior distribution $\pi(\theta|y_{obs})$. With ABC schemes the likelihood is approximated by matching trajectories y sampled, through simulation, from the likelihood $y \sim \pi(y|\theta)$ with the observations y_{obs} via a distance metric $\rho(y, y_{obs})$. This lead first to the simple ABC *rejection sampling* algorithm which has then been improved yielding the faster converging ABC-SMC algorithm.

ABC rejection sampling. The simplest form of ABC, known as *rejection sampling*, operates by iteratively sampling parameters θ' from the prior distribution $\theta' \sim \pi(.)$. For each θ' a trajectory $y' \sim p(.|\theta')$ is simulated from the corresponding model instance $\mathcal{M}_{\theta'}$ and θ' is accepted if $\rho(y', y_{obs}) \leq \epsilon$ (i.e. if it is sufficiently matching observations y_{obs}) or rejected if $\rho(y', y_{obs}) > \epsilon$, where $\epsilon \in \mathbb{R}_{\geq 0}$ represents a chose tolerance. The accepted parameters θ_i together with the corresponding traces y'_i give samples (θ_i, y_i) drawn from the joint distribution: $\pi_{ABC,\epsilon}(\theta, y \mid y_{obs}) \propto \mathbb{1}_{A_\epsilon(y,y_{obs})} p(y|\theta)\pi(\theta)$ where $A_\epsilon(y, y_{obs}) = \{y \in \mathcal{Y} \mid \rho(y, y_{obs}) \leq \epsilon\}$ (with $\mathbb{1}_{A_\epsilon(y,y_{obs})}$ denoting the indicator function representing the set of traces whose distance from y_{obs} is within the tolerance ϵ). $\pi_{ABC,\epsilon}$ approximates the posterior distribution: the smaller the ϵ, the closer the simulations y_i are to the observations y_{obs}, the better the approximation.

Algorithm 1 ABC rejection sampling

Require: N: number of particles, y_{obs} observations, tolerance ϵ, distance ρ, summary statistics η
Ensure: $(\theta_i)_{1 \leq i \leq N}$ drawn from $\pi_{ABC,\epsilon}$
  for $i = 1 : N$ **do**
    repeat
      $\theta' \sim \pi(.)$
      $y' \sim p(.|\theta')$
    until $\rho(\eta(y'), \eta(y_{obs})) \leq \epsilon$
    $\theta_i \leftarrow \theta'$
    $y_i \leftarrow y'$
  end for

Notice that in Algorithm 1, $\eta : \mathcal{Y} \to \mathcal{S} \subset \mathbb{R}^{k_1}$ represents summary statistics[5] computed on

[5]The choice of summary statistics is a crucial point in ABC (see for example [ANJB10]).

the observations y_{obs} and on the simulated trace y', while $\rho : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^+$ is a distance in the space of summary statistics.

ABC Sequential Monte Carlo method. The chosen value of ϵ is crucial for the performance of the simple ABC algorithm: a small ϵ is needed to achieve a good approximation. However, this may result in a high rejection rate leading to cumbersome computations. To overcome this issue, the more elaborate algorithm known as ABC Sequential Monte Carlo (ABC-SMC), has been proposed [BCMR09]. It is an SMC based approach [DMDJ06] through which a population of N particles is iteratively sampled with increasing accuracy until the targeted level of accuracy ϵ_M is obtained. At the first iteration, the particles are initialised through the simple ABC rejection sampling algorithm using a large enough ϵ_1 to limit the computation cost. ϵ_1 possibly equals infinity, which is equivalent to only sampling from the prior distribution. Then, at each step i, $i = 2, \ldots, M$, the particles are moved by a transition kernel $K(.|.)$ (for example, a Gaussian one [DMDJ06]) until they match the tighter, next level, approximation constraint ϵ_i. At iteration M, we finally get N particles that fulfil the desired approximation ϵ_M. Some ad-hoc strategies are proposed to find a proper sequence $(\epsilon_i)_{1 \leq i \leq M}$ ensuring an efficient convergence towards the posterior distribution.

Algorithm 2 ABC Sequential Monte Carlo

Require: N : number of particles, $y_{obs}, (\epsilon_i)_{1 \leq i \leq M}$, ρ, η
Ensure: $(\omega_j)_{1 \leq j \leq N}$, $(\theta_j)_{1 \leq j \leq N}$ weighted samples drawn from π_{ABC,ϵ_M}
  Iteration $i = 1$: find $(\theta_j^{(1)})_{1 \leq j \leq N}$ with ABC rejection sampling with tolerance ϵ_1
  $\omega_j \leftarrow \frac{1}{N}$
  for $i = 2 : M$ **do**
    for $j = 1 : N$ **do**
      repeat
        Take θ_j' from $(\theta_j^{(i-1)})_{1 \leq j \leq N}$ with probabilities $(\omega_j)_{1 \leq j \leq N}$
        $\theta_j^{(i)} \sim K(.|\theta_j')$
        $y' \sim p(.|\theta_j^{(i)})$
      until $\rho(\eta(y'), \eta(y_{obs})) \leq \epsilon_i$
      $\omega_j \leftarrow \frac{\pi\left(\theta_j^{(i)}\right)}{\sum_{j'=1}^{N} \omega_{j'}^{(i-1)} K(\theta_j^{(i)} | \theta_{j'}^{(i-1)})}$
    end for
    Normalize $(\omega_j)_j$
  end for

Remark: ABC a *distance* related class of methods. A basic point about ABC algorithms is that, by definition, they all rely on a *distance metric* (between the simulations y' and observations y_{obs}). It is precisely based on this characteristic that we extend ABC

statistical inference to the model checking problem. In Section 3.2, we first introduce the notion of *satisfiability distance*, which quantifies *how far a model instance $\mathcal{M}_\theta$ is from satisfying a temporal logic property φ* and then we exploit such a *novel distance* for adapting ABC schemes to the estimation of f_θ, i.e. the satisfaction probability function for property φ. In practice we do so by plugging a *distance automaton* in the ABC procedure and use it as a machinery to assess how far trajectories issued by an MPM model with parameter θ are from satisfying an MITL formula φ. Furthermore, as we will demonstrate, distance automata yield a null distance for any simulation that satisfy the considered formula φ therefore with the HASL-based extension of ABC, we actually estimate the ABC-posterior using a zero tolerance, i.e. with $\epsilon = 0$.

3.1.4 Kernel density estimation

Given that ABC methods only deliver samples $(\theta_i)_i$ drawn from the ABC posterior distribution $(\pi_{ABC,\epsilon}(\theta|y_{obs}))$ and that our end goal, instead, is to obtain an approximation of a continuous probability density function, namely the probability satisfaction f_φ, we need to resort to methods for reconstructing a density function given a set of samples. Specifically we resort to kernel density estimation (KDE), which allow one to derive an approximation $\widehat{\pi}$ of an unknown probability density function π given a finite number of samples of a random variable. The approximation $\widehat{\pi}$ is obtained as the sum of the application of a *kernel function* K to the samples, i.e. a continuous function which quantify the contribution, in terms of probability mass, brought by a sample to the density π to be estimated (i.e. essentially, a kernel is a manner to weight data samples).

Definition 3.7 (Kernel function)**.** A function $K : \mathbb{R} \to \mathbb{R}_{\geq 0}$ is called a kernel if:

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

Based on kernel functions, one can define a kernel density estimator [Sil86].

Definition 3.8 (Kernel density estimator)**.** Let $\theta^{(1)}, \ldots, \theta^{(N)}$ be N i.i.d samples from an unknown density π on Θ. The kernel density estimator $\widehat{\pi}$ associated with a kernel function K on $\mathcal{X}$ is:

$$\forall \theta \in \Theta, \widehat{\pi}(\theta) = \frac{1}{Nh} \sum_{i=1}^{N} K\Big(\frac{\theta - \theta^{(i)}}{h}\Big)$$

K is a kernel function, h is a *smoothing* parameter called the *bandwidth*[6].

[6]Where h is either a scalar, a vector or a matrix, depending on the dimension of Θ and the choice of K.

The rationale behind estimator $\widehat{\pi}(\theta)$ is that each value θ contributes to the probability mass over the whole set Θ, with the idea that the further θ is from the observation $\theta^{(i)}$, the lower is its contribution to the probability mass. The selection of the kernel K and the calibration of the bandwidth h are crucial points in KDE and several possibilities exist [Sil86], [CD18]. In Section 3.2.4 we adapt the KDE approach to the Automaton-ABC method more specifically by employing the minimisation of the Least Squares Cross-Validation as criterion for selecting the bandwidth h jointly with Gaussian and Beta kernels [Che99].

3.2 Automaton-ABC

We introduce an adaptation of the ABC methodology, named automaton-ABC, aimed at approximating the *satisfaction probability function* f_φ (Definition 3.6) of a MITL formula φ. It relies on the following four aspects: *i)* the formalisation of the notion of *satisfiability distance* expressing how far a model's path is from satisfying the considered formula φ *ii)* the introduction of the corresponding HASL meter automaton to measure such distance, *iii)* the definition of a novel ABC algorithm whose convergence is driven by assessment of the satisfiability distance and, finally, *iv)* the derivation of the normalisation constant through which the KDE approximation of the probability satisfaction function f_φ is obtained from the *ABC posterior density*. In the remainder we present the automaton-ABC approach taking into account simple, non-nested, time-bounded reachability problems only. Its extension to the full MITL spectrum could intuitively be straightforwardly obtained as long as the notion of satisfiability distance can be reasonably extended to the entire MITL spectrum.

3.2.1 Satisfiability distances for reachability problems

The basic idea behind the notion of satisfiability distance is that a time-bounded temporal logic specification φ, which depends on atomic propositions μ_i and time intervals $I_j \subseteq \mathbb{R}_{\geq 0}$, identifies a set of spatio-temporal regions $S_{\mu_i}^{I_j} \subseteq S \times \mathbb{R}_{\geq 0}$ (see Definition 3.4) that a path $\sigma \in Path_{\mathcal{M}_\theta}$ should (partially or totally) traverse in order to satisfy φ. Therefore one can sensibly introduce a *satisfiability distance*, denoted $d : Path_{\mathcal{M}_\theta} \times \Phi \to \mathbb{R}_{\geq 0}$, which quantifies how far a path σ is from satisfying φ by taking into account some form of Euclidean distance of σ from the relevant regions $S_{\mu_i}^{I_i}$. To characterise $d(\sigma, \varphi)$ we consider the following guidelines: *i)* $d(\sigma, \varphi)$ should evaluate to zero whenever $\sigma \models \varphi$ and *ii)* $d(\sigma, \varphi)$ should favour the convergence of the ABC algorithm.

Before formally introducing it (Definition 3.9), we illustrate, in Figure 3.1, the notion of distance for the three kind of (non-nested) temporal MITL formulae, namely $\mathbf{F}^I \mu$ (eventual reachability), $\mathbf{G}^I \mu$ (global reachability) and $\mu_1 \mathbf{U}^I \mu_2$ (conditional reachability), assuming, for simplicity, that μ, μ_1 and μ_2 corresponds each to a single elementary region, hence that

both $\mathbf{F}^I\mu$ and $\mathbf{G}^I\mu$ induce a *simple* satisfiability region S^I_μ, whereas $\mu_1\mathbf{U}^{[t_1,t_2]}\mu_2$ induce a *compound* satisfiability region given by $S^{[0,t_1]}_{\mu_1} \cup S^{[t_1,t_2]}_{\mu_2} \cup S^{[t_1,t_2]}_{\mu_1}$.

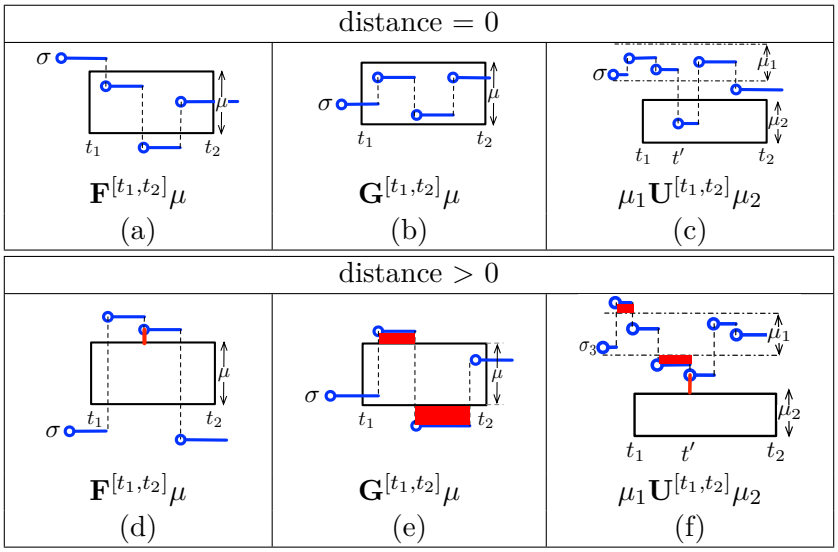


Figure 3.1: Examples of trajectories with zero-distance (left) and positive distance (right) from an F, a G and a U region (positive distances are depicted in red).

Plots in the top row of Figure 3.1 depict cases in which a path σ has to have a null satisfiability distance from a given formula. For $\mathbf{F}^I\mu$ distance has to be 0 if and only if σ has at least one point traversing region $S^{[t_1,t_2]}_\mu$ (Figure 3.1a), for $\mathbf{G}^{[t_1,t_2]}\mu$ if and only if within $t \in [t_1, t_2]$ all points $\sigma@t$ fall in $S^{[t_1,t_2]}_\mu$ (Figure 3.1b) while for $\mu_1\mathbf{U}^{[t_1,t_2]}\mu_2$ the distance is 0 if and only if there exists $t' \in [t_1, t_2]$ such that $\sigma@t'$ is in $S^{[t_1,t_2]}_{\mu_2}$ while is consistently in $S^{[0,t']}_{\mu_1}$ beforehand (Figure 3.1c)[7].

The bottom row of Figure 3.1 instead depicts cases in which a path σ has to have a positive distance, illustrating the rationale we adopted in the characterisation of the metric $d(\sigma, \varphi)$ given in Definition 3.9. Specifically for $\mathbf{F}^{[t_1,t_2]}\mu$ (Figure 3.1d), assuming σ does not enter $S^{[t_1,t_2]}_\mu$, the distance corresponds with the minimal Euclidian distance between σ and $S^{[t_1,t_2]}_\mu$ (Equation (3.3)), whereas for $\mathbf{G}^{[t_1,t_2]}\mu$ (Figure 3.1e), assuming σ leaves $S^{[t_1,t_2]}_\mu$, the distance corresponds with the volume of the hyperrectangles delimited by the segments of σ that lie outside $S^{[t_1,t_2]}_\mu$ (Equation (3.4)). Finally, for $\mu_1\mathbf{U}^{[t_1,t_2]}\mu_2$ (Figure 3.1f) the distance of σ (Equation (3.5)) bears 3 components: $d(\sigma, \mathbf{G}^{[0,t_1[}\mu_1)$ which accounts for the fact that a path satisfying $(\mu_1\mathbf{U}^{[t_1,t_2]}\mu_2)$ must never leave region S_{μ_1} before t_1, $d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu_2)$ that accounts for the fact that σ must enter region S_{μ_2} within $[t_1, t_2]$ and $d(\sigma, \mathbf{G}^{[t_1,t']}(\mu_1 \vee \mu_2))$

[7]Note that for some $t'' > t'$ $\sigma@t'' \notin S_{\mu_1}$ but this is uninfluential, as the satisfaction of $\mu_1\mathbf{U}^{[t_1,t_2]}\mu_2$ by σ is established as of what happens before t' not what happens after.

that accounts for the fact that there must be a time t' where σ switch from region μ_1 to region μ_2 directly, i.e. without spending time in any intermediate $S_{\neg(\mu_1 \vee \mu_2)}$ region: if that is not the case (i.e. if σ within $[t_1, t_2]$ has points in the complementary region $S^{[t_1,t_2]}_{\neg(\mu_1 \vee \mu_2)}$ then Equation (3.5) yields a positive value which accounts for the sum of the minimal distances of each such point from either regions S_{μ_1} or S_{μ_2} .

Definition 3.9 (Satisfiability metrics)**.** Given a path $\sigma \in Path_{\mathcal{M}}$ of a n-dimensional MPM $\mathcal{M}$ a closed time-bounding interval $[t_1, t_2] \subset \mathbb{R}_{\geq 0}$, an elementary propositional formulae μ, μ_1, μ_2, we define the distance $d(\sigma, \varphi)$ from the satisfiability region for the following kinds of temporal formulae:

1) $\varphi \equiv \mathbf{F}^{[t_1,t_2]}\mu$

$$d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu) = \begin{cases} d_e((\sigma@t_l(t_2), t_l(t_2)), S^{[t_1,t_2]}_{\mu})) \\ \qquad\qquad \textbf{if } t_l(t_2) < t_1 \wedge \sigma@t_l(t_2) \not\models \mu \\ min(d_e(\sigma@t_l(t_1), S^{[t_1,t_2]}_{\mu}), \min\limits_{t \in [t_1,t_2]} d_e((\sigma@t, t), S^{[t_1,t_2]}_{\mu}))) \\ \qquad\qquad \textbf{otherwise} \end{cases} \tag{3.3}$$

where $t_l(t^*) = min\{t \in [0, t^*] : \forall t' \in [t, t^*], \sigma@t' = \sigma@t\}$ is the time instant of the last jump occurred on σ before t^* and $d_e((s,t), S_1 \times T_1) = \min\limits_{t' \in T_1, s' \in S_1} \sqrt{(t-t')^2 + \sum_{i=1}^{d}(s[i] - s'[i])^2}$ denotes the euclidean distance of a point $(s,t) \in S \times \mathbb{R}_{\geq 0}$ from the closest point of a time-bounded region $S_1 \times T_1$.

For non-elementary propositional formulae $\mu \equiv \bigvee \mu_i$, 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.

2) $\varphi \equiv \mathbf{G}^{[t_1,t_2]}\mu$

$$d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu) = \int_{t_1}^{t_2} d_e(\sigma@t, S_\mu)dt \tag{3.4}$$

where $d_e(s, S_1) = \min\limits_{s' \in S_1} \sqrt{\sum_{i=1}^{d}(s[i] - s'[i])^2}$ denotes the euclidean distance of a point $s \in S$ from the closest state of a state space subset $S_1 \subseteq S \subseteq \mathbb{N}^d$. $d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu)$ is the integral of the Euclidean distance from S_μ of any point of σ that occurs within $[t_1, t_2]$,

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.

3) $\varphi \equiv \mu_1 \mathbf{U}^{[t_1,t_2]} \mu_2$

$$\begin{aligned} d(\sigma, \mu_1 \mathbf{U}^{[t_1,t_2]}\mu_2) = & d(\sigma, \mathbf{G}^{[0,t_1[}\mu_1) + d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu_2) + \\ & d(\sigma, \mathbf{G}^{[t_1,t_{min}]}(\mu_1 \vee \mu_2)) \end{aligned} \tag{3.5}$$

where $t_{min} = \min(\arg\min_{t\in[t_1,t_2]} d_e(\sigma@t, S_{\mu_2}))$ is the earliest time corresponding to the closest point between σ and region μ_2.

Satisfiability metric and ABC convergence aspects. In order to be employed in ABC frameworks, satisfiability distances shall account for the convergence of the ABC algorithms, in the first place by ensuring that each path is ranked with an as large a value of distance as the path is further from satisfying the considered formula, which, indeed (3.3), (3.4) and (3.5) do. Experimental evidence showed that the convergence of ABC algorithms for eventual formulae $\mathbf{F}^{[t_1,t_2]}\mu$ is affected by a peculiar aspect of a model's path σ, that is, the presence/lack of jumps within the bounding interval $[t_1, t_2]$. Specifically if, within $[t_1, t_2]$, σ does not contain any jump (and lies outside region $S_\mu^{[t_1,t_2]}$) then it is more convenient (from a convergence standpoint) that the distance $d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu)$, i.e. (3.3), is set to the Euclidian distance between region $S_\mu^{[t_1,t_2]}$ and the point entered by σ at the last jump occurred before entering $[t_1, t_2]$ even if within $[t_1, t_2]$ the path is actually closer to $S_\mu^{[t_1,t_2]}$ (e.g. Figure 3.2a). Such an aspect ensures that the ABC-driven parameter search is not mislead by anomalous situations such as, e.g. parameters $\theta \in \Theta$ that yield a model $\mathcal{M}_\theta$ for which there is a non-null probability of reaching an absorbing state in $S_{\neg\mu}$ before t_1. On the other hand, if σ contains jumps in $[t_1, t_2]$, then it is more convenient that $d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu)$ is set to the Euclidian distance of the closest point amongst the point corresponding to the last jump before t_1 and those corresponding to jumps occurring in $[t_1, t_2]$ (e.g. Figure 3.2c).

An essential aspect of the satisfiability metrics introduced in Definition 3.9 is their soundness, i.e., to prove that they yield a null distance $d(\sigma, \varphi) = 0$ if and only if $\sigma \models \varphi$.

Proposition 3 (Soundness of satisfiability metric for elementary **F**, **G** and **U**)**.** For $\sigma \in Path_{\mathcal{M}}$ a path of an MPM $\mathcal{M}$ and μ, μ_1, μ_2 are MITL propositional formulae in DNF then

$$\begin{aligned} \sigma \models \mathbf{F}^{[t_1,t_2]}\mu &\iff d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu) = 0 \\ \sigma \models \mathbf{G}^{[t_1,t_2]}\mu &\iff d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu) = 0 \\ \sigma \models \mu_1 \mathbf{U}^{[t_1,t_2]}\mu_2 &\iff d(\sigma, \mu_1 \mathbf{U}^{[t_1,t_2]}\mu_2) = 0 \end{aligned}$$

where d is as per Definition 3.9.

Proof. see [BBC21] □

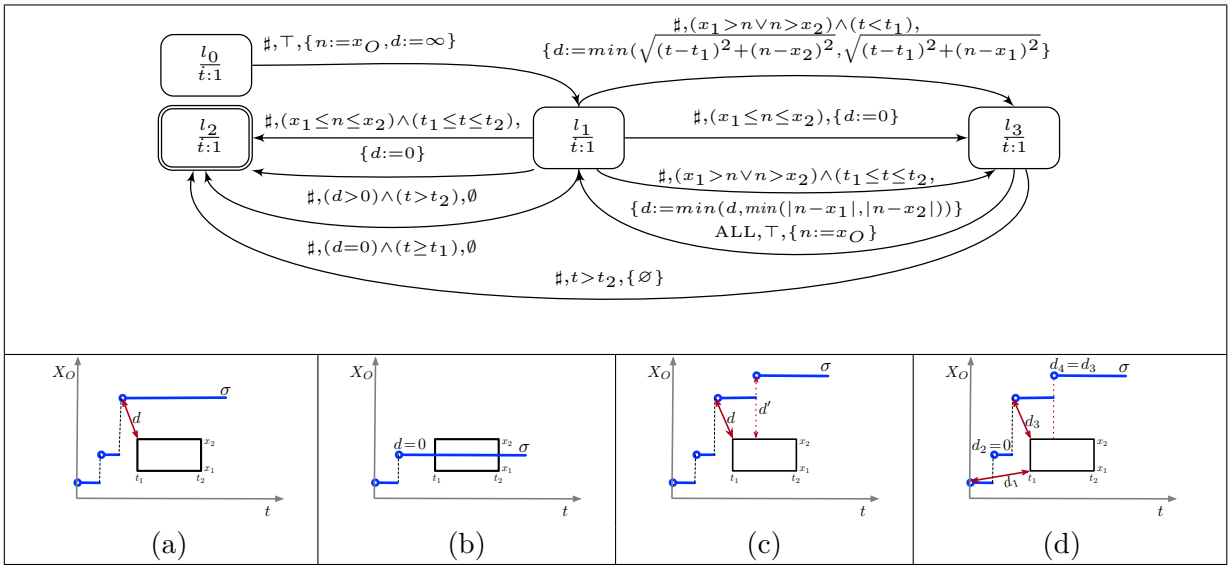


Figure 3.2: Automaton $\mathcal{A}_F$ (top) for measuring the distance of a path σ for an eventual property concerning observed species X_O and examples (bottom) of measured distance d: positive distance (a), null distance (b), selection of the minimum distance (c) in case of presence of jumps in $[t_1, t_2]$ and evolution of the computed distance d along a path (d).

3.2.2 HASL specifications for satisfiability distances

Having established the notion of satisfiability metric for non-nested MITL reachability properties we introduce hybrid automata *meters* for assessing them. For simplicity in the remainder we present automata for mono-dimensional reachability properties (i.e. temporal formulae built on top of elementary conditions such as $\mu \equiv x_1 \leq x_O \leq x_2$, where x_O denotes the population of an observable quantity O and $x_1 < x_2 \in \mathbb{N}$), bearing in mind that distance automata for formulae based on n-dimensional regions are straightforward adaptations of those discussed below.

Meter automaton $\mathcal{A}_F$. Automaton $\mathcal{A}_F$ (Figure 3.2) is designed to measure the distance (3.3) of a path σ from the region associated with $\mathbf{F}^{[t_1,t_2]}(x_1 \leq x_O \leq x_2)$, i.e. the region corresponding to the observed species $x_O \in [x_1, x_2]$ within time $t \in [t_1, t_2]$. It uses 4 variables: d (computed distance), t (current time along the path), n (population of the observed species O after the most recent occurrence of a reaction) and n' (population of O before the most recent occurrence of a reaction). The synchronisation of σ with $\mathcal{A}_F$ is managed through a number of mutually exclusive *autonomous* transitions (from l_1 to l_3), plus a single *synchronised* transition (from l_3 to l_1), which results in the automaton looping between l_1 and l_3 up until a termination condition is fulfilled. It is straightforward to show that $\mathcal{A}_F$ complies with the HASL determinism constraints (Chapter 2) and

therefore the synchronisation of an arbitrary path σ yields a unique path in the product process $\mathcal{M} \times \mathcal{A}_F$. Specifically, synchronisation of σ with $\mathcal{A}_F$ works as follows. At the start ($l_0 \to l_1$) the distance is initialised to $d := \infty$ and the initial value of the observed species are stored in $n{:=}x_O$. Once in l_1, the analysis of σ begins and is driven by seven mutually exclusive autonomous transitions. If initially σ is inside the region (and this include even initially with $t = 0$ in case $t_1 = 0$ too), then transition $l_1 \xrightarrow{\sharp, t_1 \leq t \geq t_2 \wedge (x_1 \leq n \leq x_2), \{d:=0\}} l_2$ occurs immediately and the synchronisation stops with distance $d = 0$. On the other hand if, while in l_1, the path has not entered $[x_1, x_2]$, distance d must be computed depending on different conditions (that correspond to 4 mutually exclusive autonomous transitions linking $l_1 \to l_3$). Specifically: in case $t < t_1$ (i.e. σ has not yet temporally reached the time interval $[t_1, t_2]$) then either σ has entered $[x_1, x_2]$, in which case d is correctly set to $d := 0$[8] through firing of $l_1 \xrightarrow[\{d:=0, n':=n\}]{\sharp, (t<=t_1) \wedge (x_1 > n \vee n > x_2)} l_3$ or, σ has not entered $[x_1, x_2]$ and then d is set to the Euclidian distance of the current point of the path from the nearest corner of the region (either (t_1, x_1) or (t_1, x_2)) through firing of $l_1 \xrightarrow[\{d:=\min(\sqrt{(t-t_1)^2+(n-x_2)^2}, \sqrt{(t-t_1)^2+(n-x_1)^2})\}]{\sharp, (t<=t_1) \wedge (x_1 > n \vee n > x_2)} l_3$. On the other hand if $t \geq t_1$, in accordance with (3.3), the distance of the current point of the path is either: i) left unchanged (by firing of $l_1 \xrightarrow[\emptyset]{\sharp, (t \geq t_1) \wedge (n = n') \wedge (x_1 > n \vee n > x_2)} l_3$), if the last occurred reaction had not produced a jump w.r.t. the observed species (i.e. $n' = n$), or, conversely, ii) to the minimum between the previous value of d and the distance of the current point from $[x_1, x_2]$ (by firing of $l_1 \xrightarrow[\{d:=\min(d, \min(|n-x_1|, |n-x_2|))\}]{\sharp, t \geq t_1 \wedge (n \neq n') \wedge (x_1 > n \vee n > x_2)} l_3$) if the last occurred reaction did produce a jump w.r.t. O.

Meter automaton $\mathcal{A}_G$. Automaton $\mathcal{A}_G$ (Figure 3.3) is designed to measure the distance of a path σ w.r.t. to a formula $\mathbf{G}^{[t_1,t_2]}(x_1 \leq x_O \leq x_2)$, based on (3.4). It uses the same variables as $\mathcal{A}_F$ (hence d stores the measured distance corresponding with the integral of the segments that, within $[t_1, t_2]$, fall outside the region) plus an extra timer t', to measure the duration of a segment falling outside the region within $[t_1, t_2]$, and a boolean flag *in*, which is set to true if the last segment of the path originates in $[t_1, t_2]$ 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 after a time $t'' > t_1$, in order 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 variables ($l_0 \to l_1$), analysis begins in l_1: for events 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 $[x_1, x_2]$ otherwise ($l_1 \to l_3$ 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 events occurring at $t \in [t_1, t_2]$, if $\sigma@t \notin [x_1, x_2]$ (sequence $l_1 \to l_4 \to l_1$), the distance is incremented

[8] this is because MPM paths are *càdlàg* functions of time, if the next reaction occurs at time $t \geq t_1$ then it is certain that the current path has at least one point within the considered region hence the path will be then accepted ($l_1 \xrightarrow{\sharp, (d=0) \wedge (t \geq t_1)} l_2$) and the finally measured distance will be $d = 0$).

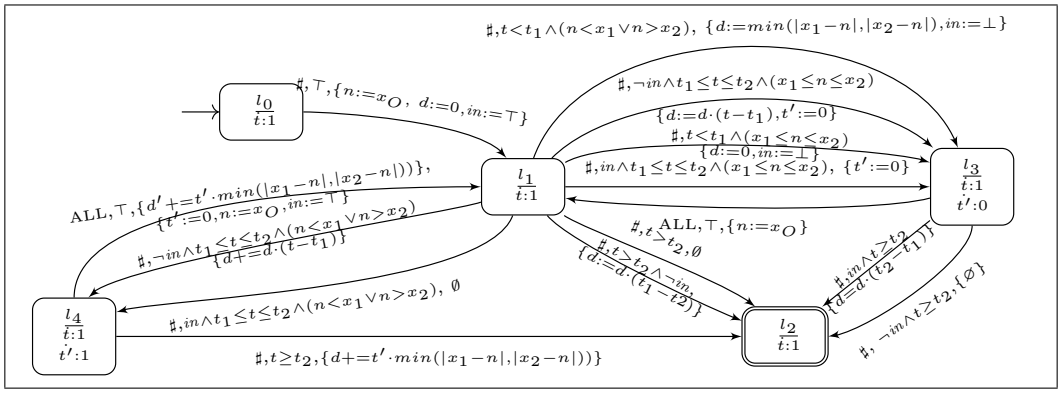


Figure 3.3: Automaton $\mathcal{A}_G$ for global property.

by the surface defined by the path 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 $l_1 \to l_3 \to l_1$).

Having introduced the meter automata for formulae $\varphi \equiv \mathbf{F}$ and $\varphi \equiv \mathbf{G}$ we prove their soundness. i.e. we demonstrate that the value stored in the automaton's variable d at the end of the synchronisation with a path σ corresponds with the distance $d(\sigma, \varphi)$ given in Definition 3.9.

Proposition 4. Let X_O be a species of an MPM model $\mathcal{M}$, $\mathcal{A}_F$ be the distance LHA corresponding to the MITL reachability formula $\varphi \equiv \mathbf{F}^{[t_1,t_2]} x_1 \leq x_O \leq x_2$ and $\sigma \in Path_{\mathcal{M}}$ be a path of $\mathcal{M}$, then:

$$last(\mathcal{A}_F(\sigma).d) = d(\sigma, \mathbf{F}^{[t_1,t_2]} x_1 \leq x_O \leq x_2)$$
$$last(\mathcal{A}_G(\sigma).d) = d(\sigma, \mathbf{G}^{[t_1,t_2]} x_1 \leq x_O \leq x_2)$$

where $last(\mathcal{A}_F(\sigma).d)$ (resp. $last(\mathcal{A}_G(\sigma).d)$) is the value stored in variable d of automata $\mathcal{A}_F$ (resp $\mathcal{A}_G$) at the end of the synchronisation between $\mathcal{A}_F$ (resp. $last(\mathcal{A}_G(\sigma).d)$) and σ while $d(\sigma, \mathbf{F}^{[t_1,t_2]} x_1 \leq x_O \leq x_2)$ (resp. $d(\sigma, \mathbf{G}^{[t_1,t_2]} x_1 \leq x_O \leq x_2)$) is the distance of σ from $\mathbf{F}^{[t_1,t_2]} x_1 \leq x_O \leq x_2$ ($\mathbf{G}^{[t_1,t_2]} x_1 \leq x_O \leq x_2$) as per the metric in Definition 3.9.

Proof. See [BBC21]. □

Meter automaton for other formulae. We stress that automata for other kind of temporal formulae such as $\mathbf{U}$ and the conjunction of a $\mathbf{G} \wedge \mathbf{F}$ have been obtained [Ben21] following a similar approach to that outlined for $\mathbf{F}$ and $\mathbf{G}$ formulae. Specifically the automaton for MITL formula: $\mathbf{G}^{[t_1,t_2]}(x_1 \leq x_O \leq x_2) \wedge \mathbf{F}^{[t_3,t_4]}(x_3 \leq x_{O'} \leq x_4)$ with $t_2 \leq t_3$ (i.e. the G region precedes the F 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' is shown to correspond to a combination of automata $\mathcal{A}_G$ and $\mathcal{A}_F$ [Ben21].

3.2.3 Automaton-ABC: ABC with satisfiability distance

Based on the notion of satisfiability distances and on the corresponding automata meters to measure them, we introduce the Automaton-ABC algorithms, i.e. the adaptation of the ABC-algorithms to estimate the satisfaction probability of a MITL formula φ by a parametric MPM $(\mathcal{M}_\theta)_\theta$. The basic idea behind Automaton-ABC schemes is to inject a model-automaton synchronisation procedure through which trajectories of the product process $\mathcal{M}_\theta \times \mathcal{A}_\varphi$ are sampled and rejected if the distance measured by $\mathcal{A}_\varphi$ through synchronisation is non-null. The accepted parameter samples θ_i constitute a sample to be used for estimating Automaton-ABC posterior distribution which we denote $\pi_{\varphi-ABC}$. We point out the estimation of the Automaton-ABC posterior distribution, differently from classical ABC schemes, is no longer computed as a limit approximation (i.e. $\lim_{\epsilon\to 0} \hat{\pi}_{ABC,\epsilon}(.|y_{obs})$), but rather as an estimation of the exact distribution, since paths are accepted exclusively if their distance to the satisfiability regions for the considered formula φ is zero.

Automaton-ABC rejection sampling. Algorithm 3 introduces a modified version of the rejection sampling ABC Algorithm 1 adapted to satisfiability distances. The algorithm takes as inputs a parametric MPM $(\mathcal{M}_\theta)_{\theta\in\Theta}$, a prior distribution π over Θ and a satisfiability distance automaton $\mathcal{A}_\varphi$ corresponding to a MITL formula φ. The workflow is as for classical ABC rejection sampling hence at each iteration, a parameter θ' is drawn from the prior $\pi(.)$, however now a path σ' is sampled from the product process $\mathcal{M}_{\theta'} \times \mathcal{A}_\varphi$ and the corresponding satisfiability distance $d(\sigma', \varphi)$ is stored in one variable of $\mathcal{A}_\varphi$; θ' is accepted only if the distance from φ is $d(\sigma', \varphi)=0$. (i.e. if $\sigma' \models \varphi$ by Proposition 3).

Algorithm 3 Automaton-ABC rejection sampling

Require: $(\mathcal{M})_{\theta\in\Theta}$ a pMPM, $\pi(.)$ prior, $\mathcal{A}_\varphi$ satisfiability distance automaton 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
      $\theta' \sim \pi(.)$
      $d(\sigma', \varphi) \sim (\mathcal{M}_{\theta'} \times \mathcal{A}_\varphi)$
    until $d(\sigma', \varphi) = 0$
    $\theta_i \leftarrow \theta'$
  end for

Automaton-ABC sequential montecarlo. The Automaton-ABC rejection sampling (Algorithm 3) suffers from slow convergence in an even stronger fashion than its standard counterpart given the strict acceptance condition (i.e. only parameters that yield a null satisfiability distance are accepted). To improve convergence we introduce the sequential Monte Carlo version of Automaton-ABC (Algorithm 4), through which we insert

a decreasing (distance) tolerance $\epsilon \in \mathbb{R}_{>0}$ for accepting parameters. In essence we rank parameters according to the satisfiability distance of the corresponding paths and accept only parameters whose corresponding satisfiability distance is within tolerance ϵ (even if the corresponding paths do not necessarily satisfy φ). This can lead to a faster convergence of the parameter inference process hence resulting in an algorithm with a reduced *runtime* (w.r.t. Algorithm 3) but that still yields samples from the posterior satisfiability density function $\pi_{\varphi-ABC}$ (as per Proposition 1).

Algorithm 4 Automaton-ABC Sequential Monte Carlo

Require: $(\mathcal{M})_{\theta\in\Theta}$ a pMPM$_,$ $\pi(.)$ prior, $\mathcal{A}_\varphi$ satisfiability distance automaton for MITL formula φ, N number of particles, $\alpha \in (0,1)$, K kernel distribution
Ensure: $(\omega_j, \theta_j)_{1\leq j\leq N}$ weighted samples drawn from $\pi_{\varphi-ABC}$

```
(θ_j^(1))_{1≤j≤N} ~ π(.)
∀j ∈ 1,...,N, d_j ~ (M_{θ_j^(1)} × A_φ)
ε ← quantile(α, (d_j)_{1≤j≤N})
(ω_j)^(1)_{1≤j≤N} ← 1/N
i ← 2
while ε > 0 do
    for j = 1 : N do
        repeat
            Take θ'_j from (θ_j^(i−1))_{1≤j≤N} with probabilities (ω_j^(i−1))_{1≤j≤N}
            θ_j^(i) ~ K(.|θ'_j)
            d_j ~ (M_{θ_j^(i)} × A_φ)
        until d_j ≤ ε
        ω_j ← π(θ_j^(i)) / Σ_{j'=1}^{N} ω_{j'}^(i−1) K(θ_j^(i) | θ_{j'}^(i−1))
    end for
    Normalise (ω_j^(i))_j
    ε ← quantile(α, (d_j)_{1≤j≤N})
    i ← i + 1
end while
```

The Automaton-ABC SMC algorithm (Algorithm 4), takes the same inputs as the Automaton-ABC rejection sampling (Algorithm 3), i.e. a parametric MPM $\mathcal{M}_\theta$, a satisfiability distance automaton $\mathcal{A}_\varphi$, the size $N \in \mathbb{N}$ of parameters' sample to be drawn at each iteration and a prior distribitution $\pi(.)$, plus two additional ones: a kernel distribution K and a hyperparameter $\alpha \in]0,1[$ used to control how fast the tolerance ϵ decreases along with the iterations. The algorithm operates iteratively following the ABC-SMC schemes of Algorithm 2 yet with a few peculiarities. Initially, N parameters $(\theta_j^{(1)})_{1\leq j\leq N}$ are drawn from the prior $\pi(.)$, their corresponding satisfiability distances $(d_j)^{(1)}_{1\leq j\leq N}$ are computed (through synchronised simulations $\mathcal{M}_{\theta_j^{(1)}} \times \mathcal{A}_\varphi$). Parameters $\theta_j^{(1)}$ of this initial sample

are all accepted (regardless of their distance $(d_j)^{(1)}$) and coupled with weights purposely uniformly set to $(\omega_j)^{(1)}_{1\leq j\leq N} \leftarrow \frac{1}{N}$ to form the first draw from the posterior distribution $\pi_{\varphi-ABC}$. Furthermore $\theta^{(1)}_j$ are the used to establish the first tolerance level ϵ which is set to the α-quantile of the distances $(d_j)^{(1)}_{1\leq j\leq N}$ (i.e. $\epsilon \leftarrow quantile(\alpha, (d_j)^{(1)}_{1\leq j\leq N})$)[9]. Notice that the first tolerance level ϵ may actually result in a quite large value of distance $d^{(1)}_j$ as no control is imposed on the selection of the initial sample $\theta^{(1)}_j$. The convergence of the parameter search process is then driven through the iterative loop. At each iteration $i \geq 2$ a new sample of parameters $\theta^{(i)}_j$ is obtained by moving parameters $\theta^{(i-1)}_j$ accepted at the previous iteration through the kernel distribution K. Each newly sampled parameters $\theta^{(i)}_j$ is accepted only if its corresponding distance $d^{(i)}_j$ (assessed through synchronised simulation $\mathcal{M}_{\theta^{(i)}_j} \times \mathcal{A}_\varphi$) is below the tolerance level ϵ established at previous iteration. The accepted N parameters $\theta^{(i)}_j$ are then used to establish the next tolerance level ϵ as the α-quantile of the distances $(d^{(i)}_j)_{1\leq j\leq N}$. The iterations end as soon as 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 $\mathcal{A}_\varphi$.

Theorem 1. For $(\mathcal{M}_\theta)_{\theta\in\Theta}$ a parametric MPM, φ an MITL formula and π a prior distribution over the parameter set Θ, the $(\theta_i)_{1\leq i\leq N}$ issued by either Algorithm 3 or Algorithm 4 are drawn from a density function $\pi_{\varphi-ABC}$:

$$\pi_{\varphi-ABC}(\theta_i) = f_\varphi(\theta_i) \cdot \frac{\pi(\theta_i)}{K}$$

where $f_\varphi : \Theta \to [0,1]$ (i.e. $f_\varphi(\theta) = Pr(\varphi \mid \mathcal{M}_\theta)$) is the probability satisfaction function of φ (Definition 3.6) and $K \in \mathbb{R}_{\geq 0}$ is a positive constant.

Proof. See [BBC21]. □

Theorem 1 links the Automaton-ABC algorithms (Algorithm 3 and 4) with the satisfaction probability function f_φ (Definition 3.6). It establishes that with Automaton-ABC the regression of a smooth function, i.e. the likelihood function targeted by classical ABC schemes, is transformed into the regression of a probability density function, i.e. the probability satisfaction function f_φ. In this respect we observe that each parameter $(\theta_i)_{1\leq i\leq N}$ sampled (through the Automaton-ABC method) from the $\pi_{\varphi-ABC}$ is informative as it corresponds to a path that verifies φ. Furthermore, as $\pi_{\varphi-ABC}$ is a probability density function, the relative position of accepted parameters is highly informative w.r.t. the estimation of the satisfaction probability function: the denser a subset of accepted parameters, the higher the satisfaction probability function over the subset.

[9] Therefore ϵ is set to the distance $\epsilon = d_k$, with $1 \leq k \leq N$ such that the empirical cumulative distribution computed over $(d_j)^{(1)}_{1\leq j\leq N}$ is $\geq \alpha$.

3.2.4 Estimation of the satisfaction probability function

Following Theorem 1 we have that:

$$f_{\varphi}(\theta_i) = K \cdot \frac{\pi_{\varphi-ABC}(\theta_i)}{\pi(\theta_i)}$$

This implies that if from the parameter's samples $(\theta^{(i)})_{1\leq i\leq M}$ we obtain by application of either Automaton-ABC algorithm we can derive an estimate of the ABC posterior density $\pi_{\varphi-ABC}(.)$ and if we can then further estimate the constant K then we inherently obtain an estimate of the satisfaction probability function $f_{\varphi}(.)$ as the product $K \cdot \pi_{\varphi-ABC}(.)$ divided by the (known) prior $\pi(.)$ (the one which has been employed in the application of the Automaton-ABC algorithm).

Estimation of the $\varphi - ABC$ posterior distribution. To obtain an estimate of the Automaton-ABC posterior distribution $\pi_{\varphi-ABC}(.)$ we apply kernel density estimation (Section 3.1.4) to the samples $(\theta_i)_{1\leq i\leq N}$ issued by an Automaton-ABC algorithm. Specifically we considered two different kind of kernels: Gaussian and Beta [Che99]. Beta kernels are useful when we have to estimate densities over bounded supports with positive probabilities on the boundaries, but are more computationally expensive for the calibration of the bandwidth. The optimal bandwidth is obtained by Least Squares Cross-Validation minimisation [Sil86].

Estimation of K. An estimate of K can be obtained directly as the ratio between a single-point estimation of $Pr(\varphi|\mathcal{M}_{\theta^*})$ and $\pi_{\varphi-ABC}(\theta^*)$ where $\theta^* \in \Theta$ is a specifically chosen value in the parameter space. Since we propose to estimate $Pr(\varphi|\mathcal{M}_{\theta^*})$ through (standard) statistical model checking, θ^* should be chosen so that the runtime to statistically model check $\mathcal{M}_{\theta^*} \models \varphi$ (hence to estimate $Pr(\varphi|\mathcal{M}_{\theta^*})$) is not eccesively high, i.e. which is the case as long as $Pr(\varphi|\mathcal{M}_{\theta^*}) \gg 0$. Therefore, based on the estimate of the posterior distribution $\pi_{\varphi-ABC}$ one should chose θ^* wisely in a high probability region of $\pi_{\varphi-ABC}$. Alternatively, in order to improve kernel density estimation stability, one could select several $\theta*$, estimate the corresponding constants K^* and compute their mean value.

3.3 Experiments

We demonstrate the effectiveness of parametric model checking via the automaton-ABC method through a few examples of biological systems .

3.3.1 Enzymatic reaction system

We consider a simple model of enzymatic reaction (ER) system in which a *substrate* species S is converted into a *product* P through mediation of an *enzyme* E. The dynamics is given

by chemical equations (3.6) which depend on the parameters $\theta = \{k_1, k_2, k_3\}$, i.e. the kinetic rate constants of reactions R_1, R_2, R_3.

$$R_1 : E + S \xrightarrow{k_1} ES \quad R_2 : ES \xrightarrow{k_2} E + S \quad R_3 : ES \xrightarrow{k_3} E + P \tag{3.6}$$

The dynamics of the ER system is such that the totality of the substrate (initially $S_0 = 100$) is converted into the product at a speed dependent on parameters θ. For example with $\theta = (1, 1, 1)$, the totality of S is converted to P before $time = 5$ (Figure 3.4 left), whereas with a tenfold speed reduction in the formation of the ES complex and synthesis of P (i.e. $\theta = (0.1, 1, 0.1)$), only about 30% of S has been converted at $time = 5$ (Figure 3.4 right).

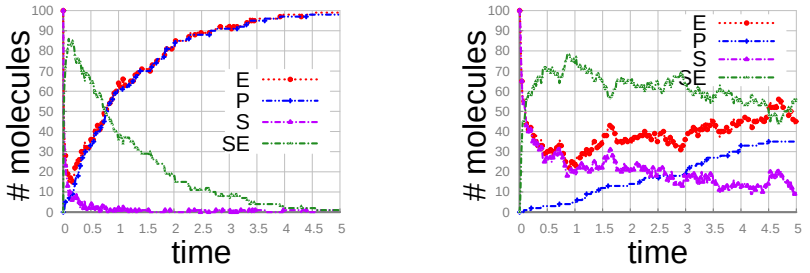
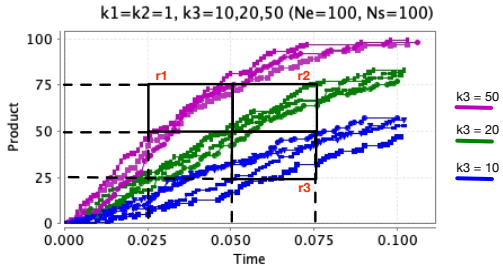


Figure 3.4: Trajectories of the ER system with $\theta_{\text{left}} = (1, 1, 1)$, $\theta_{\text{right}} = (0.1, 1, 0.1)$.

Satisfaction probability function estimation. As a case study we consider the estimation of the satisfaction probability function f_{φ_i} for two sets of time-bounded reachability formulae φ_1, φ_2, φ_3 (Figure 3.6) that we use for mono-dimensional experiments and φ_4, φ_5, φ_6 (Figure 3.7) which we use for bi-dimensional experiments. While φ_1, φ_2, φ_3, φ_4 and φ_5 refer to "simple" time-bounded reachability conditions, i.e. they are concerned with the probability that the population of a single species (i.e. species P for φ_1, φ_2, φ_3, φ_4 and E for φ_5) enter a specific elementary spatio-temporal regions (i.e. R_1, R_2, R_3, R_4 and R_5), ϕ_6 consists of a conjunction of time-bounded conditions involving two different species, namely E and P, thus corresponds with a combination of 2 elementary regions one constraining the E-projection the other concerning P-projection of paths issued by the ER model. Before discussing the outcome of f_{φ_i} estimation we point out the effect that parameter k_3 has on the dynamics of the P species by showing few paths (Figure 3.5 left) sampled from the ER model by varying parameter $k_3 \in \{10, 20, 50\}$ while keeping $k_1 = k_2 = 1$ (i.e. equally coloured plots correspond to the same parameter set). Intuitively one can point out that parameter set $\theta_1 : (1, 1, 50)$ induces a high probability that trajectories enter region R_1 (which corresponds to formula φ_1), while $\theta_2 = (1, 1, 20)$ induces a high probability on region R_2 and $\theta_3 = (1, 1, 10)$ on region R_3.



$$\varphi_1 : \mathbf{F}^{[0.025,0.05]}(50 \leq x_P \leq 75) \equiv R_1$$
$$\varphi_2 : \mathbf{F}^{[0.05,0.075]}(50 \leq x_P \leq 75) \equiv R_2$$
$$\varphi_3 : \mathbf{F}^{[0.05,0.075]}(25 \leq x_P \leq 50) \equiv R_3$$

Figure 3.5: Projection of trajectories of ER system along P-dimension, and three spatial-temporal regions with corresponding MITL formulae encoding.

Mono-dimensional experiments. Figure 3.6 illustrates the estimated posterior distribution $\pi_{\varphi_i - ABC}$ and corresponding satisfaction probability function f_{φ_i} in function of parameter k_3 (1D experiments) obtained by application of the automaton-ABC method (Algorithm 3 and 4) to formulae φ_1, φ_2 and φ_3.

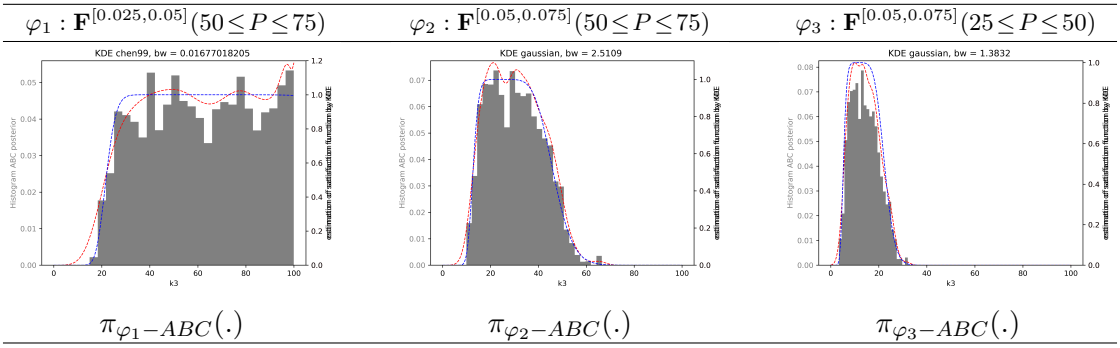


Figure 3.6: Weighted histograms of posterior distribution $\pi_{\varphi_i - ABC}$ for ER system obtained with Automaton-ABC-SMC using constant $k_1 = k_2 = 1$, prior distributions $\pi_{k_3}(.) \sim U(0, 100)$ and using 1000 particles. The red plot depicts the estaimated satisfaction probability function obtained through kernel density estimation method, while the blue plot depicts the "true" satisfaction probability function estimated on a selection of points through statistical model checking (at 95% confidence semi-interval width 0.01). In red: the satisfaction function estimated through kernel density estimation method.

The marginal for $\pi_{\varphi_1 - ABC}$ exhibits a rather uniform profile with the 95% credibility interval that φ_1 is satisfied for $k_3 \in [20, 100]$ (approximately) whereas the marginals for $\pi_{\varphi_2 - ABC}$ and $\pi_{\varphi_3 - ABC}$, instead, result in narrower 95% credibility intervals with $k_3 \in [15, 50]$ (φ_2) resp. $k_3 \in [5, 25]$ (φ_3). The estimation of the satisfaction probability function (red plot in Figure 3.6) from the posterior $\pi_{\varphi_i - ABC}$ has been obtained using Beta kernels for φ_1 (to handle boundary constraints) and Gaussian kernels for φ_2, φ_3. **Bi-dimensional**

experiments. Figure 3.7 depicts the results of the 2D experiments on the ER system for formulae $\varphi_4, \varphi_5, \varphi_6$. The triangular profile of the joint posterior in experiments for φ_4 and φ_5 (computed with $k_3 = 1$ and $\pi_{k_1}(.), \pi_{k_2}(.) \sim U(0, 100)$) indicates that only very low values of k_1 ($k_1 \leq 0.015$ for φ_4, $k_1 \leq 1$ for φ_5) combined with rather high-values of k_2 (i.e. $k_2 \in [40, 100]$ for φ_4, $k_2 \in [50, 100]$ for φ_5) result in trajectories entering $R4$, resp. never leaving $R5$, which means that the algorithm managed to catch the correlation between the parameters. This is intuitively correct in both cases, in fact $R4$ 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 trajectories to stay in $R4$. Similarly, $R5$ limits the speed of the initial decrease of E (which initially is $E_0 = 100$), to 50 within $t \leq 0.8$, which again is compatible only with slow ES complexation and cannot be compensated by fast decomplexation. The R6 experiment caught an even more important correlation between parameters, in fact the posterior for φ_6 is contained in that for φ_5 (which is expected because if a trajectory verifies φ_6 then it also verifies φ_5).

Algorithm selection remarks. The selection of either version of the Automaton-ABC algorithm, i.e. the Automaton-ABC rejection sampling or Automaton-ABC-SMC algorithm, depends very much on the computational complexity of the estimation problem, which in turns depends on the proportion between the support of the posterior distribution $\pi_{\varphi_i - ABC}(.)$ to be estimated and the support for the considered prior distribution $\pi(.)$, i.e. $|\pi_{\varphi_i - ABC}(.)|/|\pi(.)|$, where we denote $|\pi_{\varphi_i - ABC}(.)|$ (resp. $|\pi(.)|$) the size of the support of $\pi_{\varphi_i - ABC}(.)$ (resp. $\pi(.)$). Since there is no way to predict $|\pi_{\varphi_i - ABC}|$ the only way to proceed is by empirically monitoring the runtime of the considered experiment settings. For the mono-dimensional experiments, i.e. those concerning formulae φ_1, φ_2 and φ_3, we noticed no notable differences of performance between Automaton-ABC rejection sampling (Algorithm 3) and Automaton-ABC-SMC (Algorithm 4): both can be applied and results in comparable runtime. This is because of the fairly large ratio $|\pi_{\varphi_i - ABC}(.)|/|\pi(.)|$ between the support of the obtained posterior distributions, and the size of the sampling support given the prior for the single parameter is $k_3 \sim U(0, 100)$. Conversely for the bi-dimensional experiments (φ_4, φ_5 and φ_6), we point out that Automaton-ABC rejection sampling algorithm is not worth using because of its extremely large runtime. In fact for those experiments the support for the prior distribution is $[0, 100]\times[0, 100]$ while the region with non-null probability of satisfying the considered formula corresponds with, in the worst case (i.e. the support of $\pi_{\varphi_4 - ABC}$ in Figure 3.7 top left), the area of a triangle with sides roughly 90 and 0.03. Therefore the probability of sampling a bi-dimensional parameter (k_1, k_2) in the resulting posterior-distribution is roughly $|\pi_{\varphi_i - ABC}(.)|/|\pi(.)| \approx \frac{90 \cdot 0.03}{2} \cdot \frac{1}{100 \cdot 100} \approx 10^{-4}$. This results in an infinitesimal probability of drawing from the prior $N = 1000$ particles that fall in such a narrow distribution let alone the fact that even a parameter sampled in the obtained distribution could produce paths that don't satisfy φ. By adding several transitional steps with Algorithm 4, the problem becomes treatable. The results for φ_4 required about 2×10^5 simulations of the model (i.e. the highest number of simulations

for experiments on the ER model).

Further case studies. We tested the Automaton-ABC method on other systems including the popular SIR model of infection spreading and a model of intracellular viral infection (see [BBC21]).

3.4 Perspectives

We introduced a novel approach for parametric verification of temporal logic specifications against probabilistic models based on an adaptation of the ABC parameter inference scheme. As the original ABC scheme is based on a notion of *distance* (between the system's observations and an instance of the model's parameters) the parametric verification extension relies on the adaptation of the distance notion to the *satisfiability distance*, which quantifies how far a model's instance is from satisfying a given temporal logic condition. We have then shown that dedicated hybrid automaton monitor can be defined and plugged in within an adapted ABC scheme in order to assess on-the-fly the satisfiability distance of a temporal property, yielding an approximation of the satisfaction probability distribution for the considered property. The presented approach suffers from lack of generality as, in its current formulation, the satisfiability distance is custom-defined and restrained to non-nested time-bounded temporal specifications, rather than covering the entire syntactical spectrum of a temporal logic formalism as done by Donzé and Maler in similar approaches that target the robustness of MITL formulae against deterministic (ODEs) models [DM10]. On the other hand the fact we employ hybrid automata to monitor the paths issued by a model's instance allows us to widen the realm of parametric verification problems by taking into account much more sophisticated behaviours such as e.g. that of tuning of stochastic oscillators, as discussed in Chapter 4.

One aspect worth investigating in order to evolve this contribution is how to generalise the satisfiability distance adaptation of the ABC scheme to the entire MITL spectrum. Intuitively that would entail working out a reformulation of the satisfiability distance definition based on the recursive syntax of MITL temporal formulae, similar to Donzè and Maler's approach to tackle robustness of MITL properties [DM10]. Given a recursive definition of satisfiability distance can be obtained then it'd also necessary to figure out how it can be conveniently encoded in hybrid automata terms.

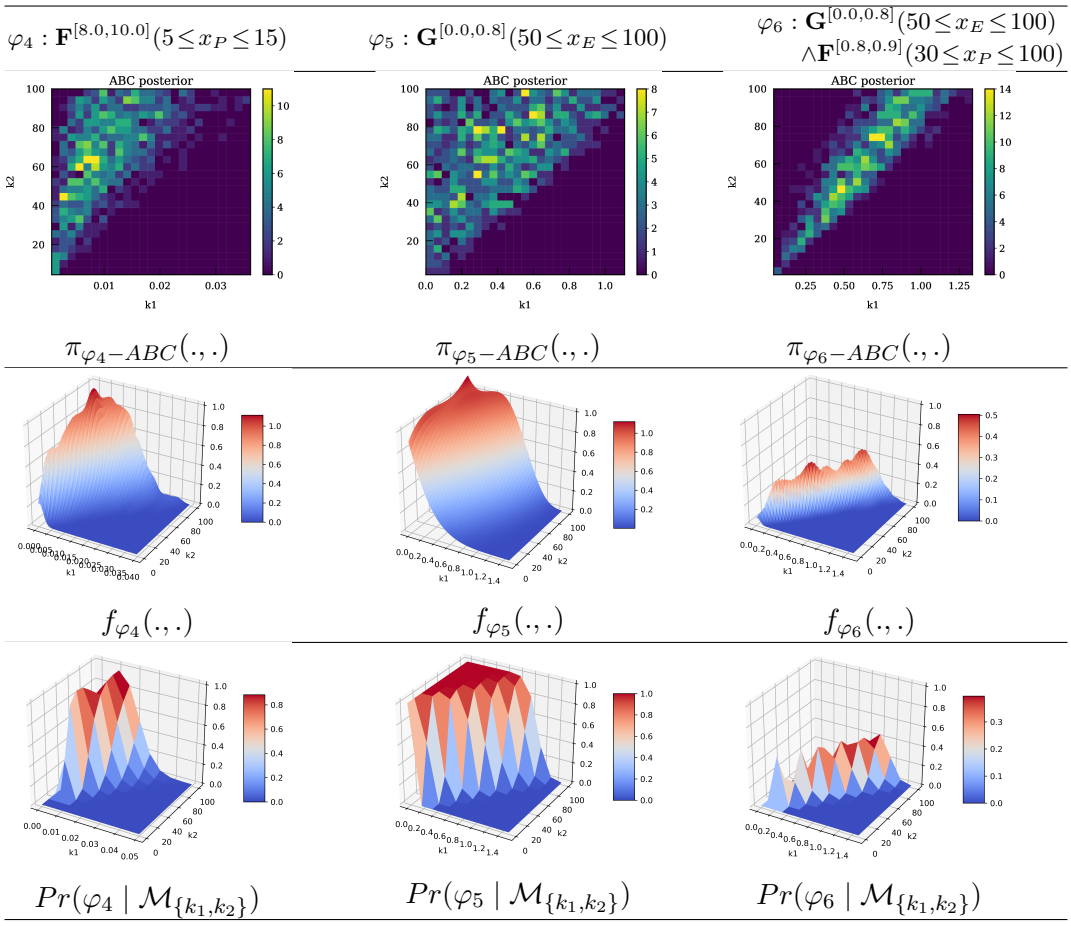


Figure 3.7: Top: bi-dimensional weighted histograms of posterior distribution π_{φ_i-ABC} for ER system obtained with Automaton-ABC-SMC using constant $k_3 = 1$, prior distributions $\pi_{k_1}(.), \pi_{k_2}(.) \sim U(0, 100)$ and using 1000 particles. Mid: kernel density estimation of the satisfaction probability function. Bottom: validation through estimation of the satisfaction probability function by statistical model checking (95% confidence semi-interval with half-width 0.01).

Chapter 4

Formal analysis of stochastic oscillators

4.1 Introduction

Many real life systems are characterised by some form of oscillatory dynamics examples of which can be found in many domains, ranging, from ecology, e.g. with the early Lotka-Volterra *prey-predator* model [Lot09, Vol28], to social systems, where periodic behaviour can be observed in many social context for example with the arrivals of patients at hospital's emergency department [BDHA20] and, specially, in biology where oscillations are prominent dynamics at the core of many fundamental biological processes [Gol02].

Here we focus on two complementary aspects related to modelling oscillators: the assessment of meaningful indicators (e.g. the period and amplitude of oscillations) from an oscillator model and the *reverse engineering* of an oscillator (i.e. how to tune an oscillator model so that it meets a given a target oscillatory behaviour). As for the first aspect, the analysis of mathematical models of oscillators is a well established problem in the literature, which typically boils down to *limit-cycle analysis* for *deterministic oscillators* (i.e., oscillators modelled as a systems of Ordinary Differential Equations), or, alternatively is achieved by application of signal processing methods such as Fast Fourier Transformation (FFT) or autocorrelation analysis. In this contribution we focus on discrete-state stochastic oscillators and propose an alternative approach, which differently from classical signal processing, is based on finding a *temporal logic* characterisation of periodicity for noisy signals issued by a stochastic model. With such a formal characterisation of what we name *noisy periodicity* we can then take advantage of the model checking framework so to automatically assess the period and amplitude of oscillations.

As for the second aspect we look at analysing the relationship between the model's parameters and the character of the oscillations. More specifically we tackle the problem of how to *reverse engineer* an oscillator, that is, given a target oscillatory behavior, e.g.

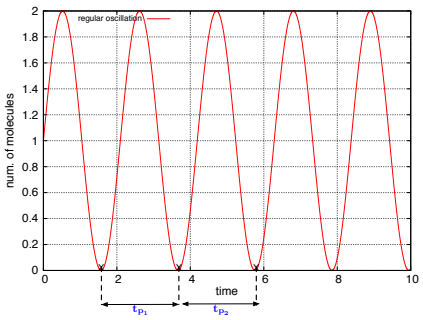
a desired period of oscillation, we aim at inferring the model's parameters that *match* the desired behaviour. We obtain such a framework based on the automaton-ABC approach outlined in Chapter 3.

Outline of the contribution. This chapter report on the contribution in the matter of formal analysis of stochastic oscillators. We demonstrate the effectiveness of the HASL formalism as a means for specifying and automatically estimate oscillation related measures, such as the period and amplitude of noisy oscillators. We do so by introducing two alternative characterisations of noisy periodicity, one relying on a pair of selected observational thresholds, the other based on the detection of (noisy) local maxima and minima. We then provide a formal encoding in terms of linear hybrid automata (LHA) of the HASL logic for both characterisations of periodicity, which we demonstrate being capable of detecting the periods, respectively the peaks, of the stochastic oscillator they target and to estimate *on-the-fly* "classical" indicators such as the average duration and the average amplitude of the oscillations as well as more sophisticated ones like the period variance, which allow for assessing the *regularity* of the oscillator. Differently from Spieler's approach [Spi13], which allows one for qualitatively establishing whether a CTMC model is a sustained oscillator and, if so, to assessing some quantitative characteristics (e.g. the average period duration), the methodology we introduce here is limited to assessing quantitative characteristics of a stochastic model which is known (or believed to) oscillates.

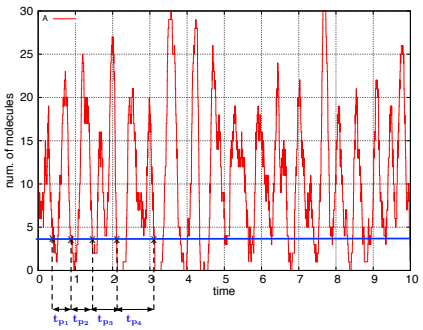
4.2 Formal analysis of stochastic oscillators

Intuitively an oscillation is the periodic variation of a quantity around a given value. In mathematical terms this is associated with the definition of (non-constant) periodic function. i.e. a function $f : \mathbb{R}^+ \to \mathbb{R}$ for which $\exists t_p \in \mathbb{R}^+$ such that $\forall t \in \mathbb{R}^+$, $f(t) = f(t + t_p)$, where t_p is called the period as in e.g., Figure 4.1(a). In the context of stochastic models such a "deterministic" characterisation of periodicity is of little relevance, as the trajectories of a stochastic oscillator being strictly periodic (as in $f(t) = f(t + t_p)$), will have (unless in degenerative cases) zero probability. More generally the paths of (discrete-state) stochastic oscillators are characterised by a remarkable level of noise: period occurrences differ in duration as well as in amplitude as in e.g., Figure 4.1(b).

The difficulty we face is that of finding a suitable characterisation of what should be considered as a *period realisation* which agrees with the intuitive notion of periodicity, while copying with the noisy nature of stochastic oscillator paths. The relevant question in this respect is "what shall one consider as a period realisation"? We propose two complementary approaches to determine what is a noisy period realisation. With the first, inspired by Spieler [Spi13] and further developed in [BD15], one has to consider a partition of the domain of the oscillating quantity in three region: two "extreme" regions (*low*, respectively, *high*) separated by an intermediate region (*mid*). A period realisation is then considered as the delay between the beginning of two consecutive sojourns in either of the



(a) deterministic oscillation: period occurrences and peaks are deterministic (period equal to 2, maxima at 2 and minima at 0)



(b) noisy oscillation: the period occurrences depend on the "observation point" (blue line) and are noisy.

Figure 4.1: Deterministic versus stochastic oscillations

extreme regions, interleaved by a sojourn in the opposite extreme region (see Figure 4.2(b)). With the second one, introduced in [BD15, Bal15a], a period realisation is considered as the delay between two consecutive *local maxima* (or equivalently local minima), bearing in mind that the characterisation of local maxima and minima in a noisy oscillator is not trivial as one has to be able to distinguish between actual local maxima/minima points and *critical points* that correspond to noisy spikes.

4.2.1 Temporal logic characterisation of oscillations

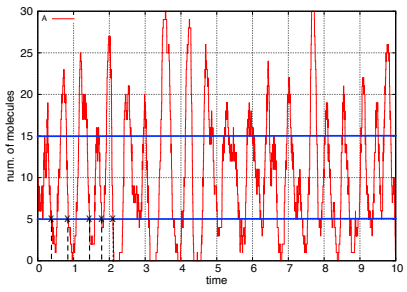
The possibility to automatically assess the periodic character of a system has attracted the attention of researchers in model checking community in recent times, the basic idea being to identify suitable temporal logic formulae that can be fed in to a model checker to *qualitatively* establish the existence of periodicity and (possibly) to *quantitatively* assess relevant indicators (e.g. period duration, amplitude of oscillations).

In an early work Fages, *et al.* [CRCD$^+$04] tackled the problem w.r.t. transition system models of cell cycle and proposed the following (branching-time) CTL formula to capture the existence of periodicity:

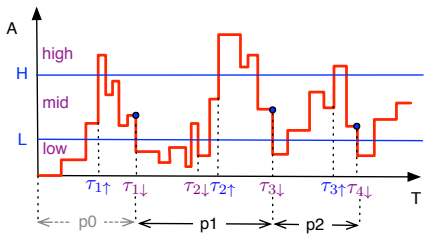
$$EG((\mathtt{p} \Rightarrow EF(\neg \mathtt{p})) \wedge (\neg \mathtt{p} \Rightarrow EF(\mathtt{p}))) \tag{4.1}$$

Formula (4.1) essentially states that there should exist an execution of the model such that *infinitely often* if a condition p is satisfied then it will be not satisfied and vice-versa if it is not satisfied it will be satisfied

In a seminal work with Mardare and Mura [BMM09] we considered a similar CTL-like reasoning to query whether a CTMC population model of a simple *3-way* synthetic



(a) a regular oscillation centred at 1 with maxima at 2, minima at 0, and period equal to 2



(b) Example of trace σ_A which is *noisy periodic* w.r.t to species A and a given ($L < H$ induced) partition of a DESP state space.

Figure 4.2: Noisy periodic trajectories of stochastic oscillators

biochemical oscillator exhibits *sustained oscillations* consisting of the following formula:

$$AG(((X_i = k) \Rightarrow EF(X_i \neq k)) \wedge (((X_i \neq k) \Rightarrow EF(X_i = k)))) \tag{4.2}$$

Formula (4.2) represents a stricter version of (4.1) and establishes, by replacing the outermost existential path quantifier E with the stricter universal quantifier A and by instantiating $\mathtt{p}$ with $(X_i = k)$, that if the system's evolution reaches a state where the i-th species is $X_i = k$ ($k \in \mathbb{N}$ being a constant) then it has to be possible to reach a future state such that $(X_i \neq k)$ and vice versa. Through basics equivalences we then derived a *probabilistic* CSL counterpart of (4.2), that is:

$$P_{\leq 0}[F(((X_i = k) \wedge P_{\leq 0}[F(X_i \neq k)]) \vee (((X_i \neq k) \wedge P_{\leq 0}[F(X_i = k)])))] \tag{4.3}$$

which can be used to rule out states which allow for converging evolutions, that is states for which there's a positive probability to reach a state $X_i = k$ and never leave it[1].

The problem with formulae for capturing sustained oscillations of population models, such as (4.2) and its CSL counterpart (4.3), is that they account also for those trajectories whose infinite oscillations around k corresponds to *stochastic noise spikes* (e.g. trajectories such as $\sigma_i : (x_i{=}k, x_i{=}k+1)^\omega$) rather than to actual oscillations therefore we proposed a slight variant formula:

$$AG(((X_i = k) \Rightarrow EF(X_i \leq k{-}n \wedge X_i \geq k{+}n)) \wedge (((X_i \leq k{-}n \wedge X_i \geq k{+}n) \Rightarrow EF(X_i = k)))) \tag{4.4}$$

which is meant to rule out oscillations corresponding to an arbitrary level of noise n.

[1]Notice that (4.3) relies on the assumption that all trajectories hit the level of exactly k molecules, that is, the maximum change of the molecule number is one.

In a follow up work [BG10] we proposed an evolved qualitative characterisation of periodicity by considering notions of paths' monotonicity, specifically by identifying the monotonic increase, respectively decrease, of the i-th species along a path by enriching the CTMC model with dedicated boolean variables named inc_i, respectively dec_i. That resulted in the following probabilistic LTL formula

$$P_{=?}[inc\_i\ U(X_i = k \wedge (dec\_i\ U\ (X_i = j))] \tag{4.5}$$

which can be used to measure the probability of paths whose initial state $X_i = j$ and that exhibit a one-period prefix of amplitude $k-j$ consisting of monotonic increase up to $X_i = k$ followed by a monotonic decrease back to the initial level $X_i = j$.

$$P_{=?}[inc\_i\ U(X_i = k \wedge ((k-n \leq X_i \leq k+n)\ U\ (dec\_i\ U\ (X_i = j))))] \tag{4.6}$$

To relax the quite strict characterisation given by (4.5) we introduced the variant formula (4.6) that allows the fluctuation to stay at the peak, that is, around a molecule level of k (module noise band n), for an unlimited amount of time, before beginning the monotonic descent to $X_i = j$.

Unfortunately all previously discussed attempts to come up with a temporal logical characterisation of oscillations only partially comply with the start goal of this investigation. For example qualitative CSL formulae like (4.3) although will be unsatisfied by damped oscillators, will also be satisfied by any *ergodic* CTMC model no matter if the model actually oscillates. On the other hand quantitative CSL formulae like (4.5) and (4.6) help us quantifying how likely oscillations of given amplitude are, but they do not actually help to disambiguate between sustained and damped oscillations as they inherently account for a finite number periods.

A major progress in this respect is given by the PhD work of David Spieler which lead to the publication of paper [Spi13]. The main idea of Spieler was to characterise noisy oscillations by partitioning the state space of the oscillating quantity in 3 regions, a *high region* where the maxima points of the oscillations are supposedly located, a *low region* where the minima are located and an intermediate *mid region* which is traversed in between each consecutive pair of minimum and maximum. Given such characterisation Spieler introduced a corresponding *period detector* deterministic timed automaton (DTA), i.e. a DTA capable of accepting noisy periodic paths corresponding to the considered *low-mid-high* state-space partition. Spieler then demonstrated that deciding whether a Markov population model (MPM) admits sustainable oscillations corresponds to check whether a specific steady-state condition holds over the corresponding, so-called, *period detector Markov population model* (PDMPM), i.e. the (DTA $\times$ MPM) product process. Undoubtably a major step forward Spieler's approach suffers nonetheless of some drawback: firstly being based on numerical model checking approaches it is affected by state-space explosion, even more so given the size of the product process PDMPM is at least thrice[2]

[2]The period detector DTA consists of 3 states.

the size of the MPM alleged oscillator. Secondly it does not naturally comply with the separation of concern modelling philosophy, as it requires the integration of the detector DTA within the model, something which probabilistic model checking tools (e.g. PRISM) that do not support automata based property languages suffer of. Finally the choice of relying on numerical model checking inherently limits the expressiveness of the oscillation related indicators that one may want to consider: Spieler's outline a transient-analysis based algorithm to approximate the PDF/CDF of the oscillation's period duration however extending such an approach so that other indicators such as, e.g., the variance of the period and/or the PDF/CDF of the oscillation's amplitude looks non trivial if feasible at all.

In the remainder we outline our contribution in this context, specifically, we describe how, by taking advantage of the more expressive hybrid automata based HASL model checking approach, we extended Spieler's DTA period detection analysis so that more sophisticated oscillation related indicators can be assessed.

4.2.2 Partition based noisy periodicity

In the remainder we consider n-dimensional DESP population models, that is, models whose state space is contained in $\mathbb{N}^n$ and we introduce the notion of *noisy periodicity* w.r.t. to one dimension of the model which, for the sake of simplicity, we often refer to as as the A dimension, where A is the name of a generic biochemical species. Therefore given a species A whose periodicity we want to assess the idea behind such characterisation is to establish a partition (w.r.t. to the A dimension) of the model state-space induced by two threshold levels $L, H \in \mathbb{N}$ with $L < H$ and we say that, a path of the model oscillates (w.r.t. A) or, equivalently is *noisy periodic*, if it the projection of path over the A-dimension infinitely often alternate between a traversal of the $[0, L]$ and of the $[H, \infty)$ regions. This is formalised by Definition 4.1.

Definition 4.1 (noisy periodic trajectory)**.** For $\mathcal{M}_\theta$ an n-dimensional DESP population model, let $dom(\mathcal{M}_\theta)$ be its state space, and $dom_i(\mathcal{M}_\theta)$ the projection of $dom(\mathcal{M}_\theta)$ along the i^{th} dimension $1 \leq i \leq n$ of $\mathcal{M}_\theta$. Let $L, H \in \mathbb{N}$, $L < H$, be two levels establishing the partition $dom_i(\mathcal{M}_\theta) = low \cup mid \cup high$ with $low = [0, L)$, $mid = [L, H)$ and $high = [H, \infty)$. A path $\sigma \in Path(\mathcal{M}_\theta)$ is said *noisy periodic* w.r.t the i^{th} dimension, and the considered L, H induced partition of $dom_i(\mathcal{M}_\theta)$ if the projection σ_i visits the intervals low, mid and $high$ infinitely often.

H/L-crossing points. Given a noisy periodic trace σ_A we denote $\tau_{j\downarrow}$ ($\tau_{j\uparrow}$), the instant of time when σ_A enters for the j-th time the low ($high$) region. $T_\downarrow = \cup_j \tau_{j\downarrow}$ (resp. $T_\uparrow = \cup_j \tau_{j\uparrow}$) is the set of all *low-crossing points* (reps. *high-crossing points*). Observe that $T_\downarrow$ and $T_\uparrow$ reciprocally induce a partition on each other. Specifically $T_\downarrow = \cup_k T_{k\downarrow}$ where $T_{k\downarrow}$ is the subset of $T_\downarrow$ containing the k-th sequence of contiguous *low-crossing points* not interleaved by any *high-crossing point*. Formally $T_{k\downarrow} = \{\tau_{i\downarrow}, \ldots, \tau_{(i+h)\downarrow} | \exists k', \tau_{(i-1)\downarrow} < \tau_{k'\uparrow} < \tau_{i\downarrow}, \tau_{(i+h)\downarrow} < \tau_{(k'+1)\uparrow}\}$. Similarly $T_\uparrow$ is partitioned $T_\uparrow = \cup_k T_{k\uparrow}$ where $T_{k\uparrow}$ is the subset of $T_\uparrow$ containing the k-th

sequence of contiguous *high-crossing points* not interleaved by any *low-crossing point*. For path σ_A in Figure 4.2(b) we have that $T_{\downarrow} = T_{1\downarrow} \cup T_{2\downarrow} \cup T_{3\downarrow} \ldots$ with $T_{1\downarrow} = \{\tau_{1\downarrow}, \tau_{2\downarrow}\}$, $T_{2\downarrow} = \{\tau_{3\downarrow}\}$, $T_{3\downarrow} = \{\tau_{4\downarrow}\}$, while $T_{\uparrow} = T_{1\uparrow} \cup T_{2\uparrow} \cup T_{3\uparrow} \ldots$ with $T_{1\uparrow} = \{\tau_{1\uparrow}\}$, $T_{2\uparrow} = \{\tau_{2\uparrow}\}$, $T_{3\uparrow} = \{\tau_{3\uparrow}\}$. Based on H/L crossing points we formalise the notion of *period realisation* for a noisy period path.

Definition 4.2 (k^{th}noisy period realisation)**.** For σ_A a noisy periodic trajectory with crossing point times $T_{\downarrow} = \cup_{k \geq 1} T_{k\downarrow}$, respectively $T_{\uparrow} = \cup_{k \geq 1} T_{k\uparrow}$, the realisation of the k^{th} noisy period, denoted t_{p_k}, is defined as $t_{p_k} = min(T_{(k+1)\downarrow}) - min(T_{k\downarrow})$[3].

Figure 4.2(b) shows an example of *period realisations*: the first two period realisations, denoted $p1$ and $p2$, are delimited by the *mid*-to-*low* crossing points corresponding to the first entering of the *low* region which follows a previous sojourn in the *high* region and their duration (as per Definition 4.2) is $t_{p_1} = \tau_{3\downarrow} - \tau_{1\downarrow}$ respectively $t_{p_2} = \tau_{4\downarrow} - \tau_{3\downarrow}$. Notice that the time interval denoted as $p0$ does not represent a complete period realisation as there's no guarantee that $t = 0$ corresponds with the actual entering into the *low* region. Definition 4.2 correctly does not account for the first *spurious* period $p0$. Relying on the notion of period realisation we characterise the *period average* and *period variance* of a noisy periodic trace. Observe that the *period variance* allows us to analyse the regularity of the observed oscillator, that is, a "regular" ('irregular") oscillator is one whose traces exhibit little (large) period variance.

Definition 4.3 (period mean)**.** For σ_A a noisy periodic trajectory the period average of the first $n \in \mathbb{N}$ period realisations, denoted $\bar{t}_p(n)$, is defined as $\bar{t}_p(n) = \frac{1}{n} \sum_{k=1}^{n} t_{p_k}$, where t_{p_k} is the k-th period realisation.

Observe that for a sustained oscillator, the average value of the noisy-period, in the long run, corresponds to the limit $\bar{t}_p = \lim_{n \to \infty} \bar{t}_p(n)$.

Definition 4.4 (period variance)**.** For σ_A a noisy periodic trajectory the period variance of the first $n \in \mathbb{N}$ period realisations, denoted $s^2_{t_p}(n)$, is defined as $s^2_{t_p}(n) = \frac{1}{n} \sum_{k=1}^{n} (t_{p_k} - \bar{t}_p(n))^2$, where t_{p_k} is the k-th period realisation and $\bar{t}_p(n)$ is the period average for the first n period realisations.

We introduce an HASL specification which consists of an LHA automaton, named $\mathcal{A}_{per}$, which allows for estimating *on-the-fly*[4] the period mean and variance, that is, as the noisy periodic traces are generated and scanned by $\mathcal{A}_{per}$.

[3] t_{p_k} could alternatively be defined as $t_{p_k} = min(T_{(k+1)\uparrow}) - min(T_{k\uparrow})$, that is, w.r.t. crossing into the *high* region, rather than into the *low* region. It is straightforward to show that both definitions are semantically equivalent, i.e., the average value of t_{p_k} measured along a trace is equivalent with both definitions.

[4] Based on an adaptation of the so-called *online algorithm* [Knu97] for computing the mean and variance out of a sample of observations.

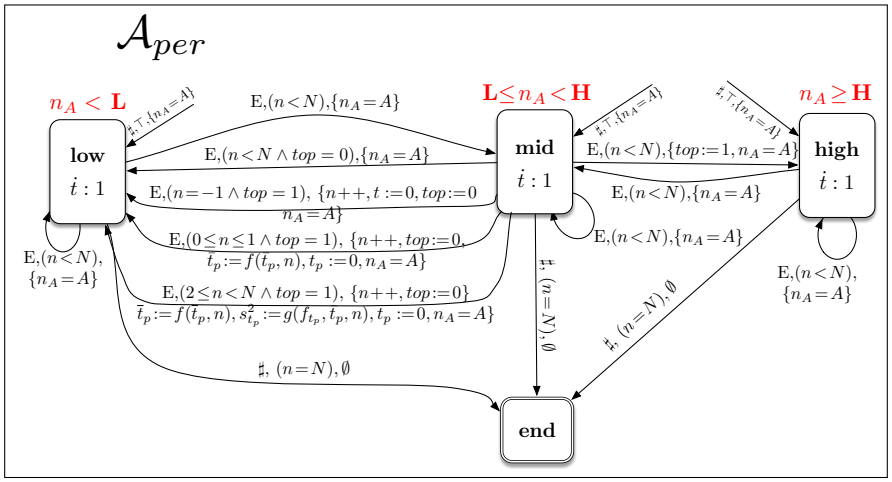


Figure 4.3: The $\mathcal{A}_{per}$ LHA for selecting noisy periodic traces (with respect to an observed species A) related to partition $low = (-\infty, L]$, $mid = (L, H)$ and $high = [H, +\infty)$.

The $\mathcal{A}_{per}$ automaton. We introduce a LHA named $\mathcal{A}_{per}$ (Figure 4.3) for assessing the mean value (as per Definition 4.3) and the variance (as per Definition 4.4) of the first N noisy period realisations (as per Definition 4.2) measured on paths of a DESP oscillator $\mathcal{M}_\theta$. The automaton consists of three main locations **low**, **mid** and **high** corresponding to the regions of the partition of A's domain induced by thresholds $L < H$: location **low** corresponds to region $low = (-\infty, L]$, location **mid** to region $mid = (L, H)$ and location **high** to region $high = [H, +\infty)$. The functioning of $\mathcal{A}_{per}$ is as follows: processing starts in either of the 3 initial locations (low,mid,high) depending on the initial state of the oscillator (variable n_A registers the population of oscillating species A and is initialised through autonomous transitions before unfolding of a path begins). While a path of the considered oscillator is simulated the $\mathcal{A}_{per}$ automaton follows the oscillation by moving between locations **low** and **high**, passing through **mid**, following the profile of the observed species A. The completion of a loop from **low** to **high** and back to **low** corresponds to detection of a period realisation (as of Definition 4.2). During period detection a number of relevant information is stored in the data variables of $\mathcal{A}_{per}$ (Table 4.1) which are then exploited for estimating the considered target expressions. The analysis of the simulated trajectory ends by entering location **end** as soon as the N-th period has been detected.

An execution of $\mathcal{A}_{per}$. Let us assume that initially $\mathcal{A}_{per}$ is in location **low**. From **low** the automaton follows the profile exhibited by the observed population A, thus moving to **mid** (and possibly back) as soon as the population of A grows and a state of the $mid = (L, H)$ region is entered (i.e., corresponding to the $L < A < H$ invariant of **mid** location becoming satisfied), and then to **high** (and possibly back) as soon as the population of A enters the $high = [H, +\infty)$ region (corresponding to the $A \geq H$ invariant of **high** location). On

name	domain	update definition	description
t	$\mathbb{R}_{\geq 0}$	*reset*	total time elapsed
n	$\mathbb{N}$	*increment*	counter of detected periods
t_p	$\mathbb{R}_{\geq 0}$	*reset*	duration last period
$\bar{t}_p$	$\mathbb{R}_{\geq 0}$	$f(\bar{t}_p, t_p, n) = \frac{\bar{t}_{p_n} \cdot n + t_p}{n+1}$	period mean
$s^2_{t_p}$	$\mathbb{R}_{\geq 0}$	$g(s^2_{t_p}, \bar{t}_p, t_p, n) = \frac{n-1}{n-2} \cdot s^2_{t_p} + \frac{(\bar{t}_p - t_p)^2}{n}$	period variance

Table 4.1: Variables of the $\mathcal{A}_{per}$ automaton.

entering the **high** location the boolean variable *top* is set to true (i.e., $top{=}1$). This allows then for distinguishing between the **mid**-to-**low** transitions of kind $\mathbf{mid} \xrightarrow{E,(\ldots \wedge top=1),\ldots} \mathbf{low}$, which correspond to an actual closure of a period realisation (i.e., those $\tau_{j\downarrow}$ preceded by a sojourn in the $high = [H, +\infty)$ region), from those of kind $\mathbf{mid} \xrightarrow{E,(\ldots \wedge top=0),\ldots} \mathbf{low}$ which correspond to a return to **low** without having previously sojourned in **high**. Observe that from **mid** location there are four possible (mutually exclusive) ways of entering the **low** location. If the sojourn in **mid** has not been preceded by a sojourn in **high** edge $\mathbf{mid} \xrightarrow{E,(n<N \wedge top=0),\ldots} \mathbf{low}$ is enabled. On the other hand if the sojourn in **mid** has been preceded by a sojourn in **high** but **low** is going to be re-entered for the first time (i.e., $n = -1$) then the timer t is reset (representing the start time of actual period detection) and the counter of detected periods n is set to zero (again representing the actual beginning of counting of period detection). Furthermore if the sojourn in **mid** has been preceded by a sojourn in **high** and the period to be detected is the first one (i.e., $0 \leq n \leq 1 \wedge top = 1$) then we increment the counter n of detected period, we reset the flag *top* and update the value of the average duration of detected period $\hat{t}_p$ while we do not update the variable $s^2_{t_p}$ as in order to update the value of the variance of the detected period duration we need that at least two periods have been detected. Finally if the period to be detected is the n-th with $n \geq 2$ (i.e., corresponding to guard $2 \leq n \leq N \wedge top = 1$) we do the same update operations of the previous case but also update $s^2_{t_p}$.

The automata uses variable n to count the number of noisy periods detected along a trajectory, and stops as soon as the N^{th} period is detected (i.e. event bounded measure). The boolean variable *top*, which is set to *true* on entering of the *high* location, allows for detecting the completion of a period (i.e. crossing from *mid* to *low* when *top* is *true*). Two clock variables, t and t_p, maintains respectively the total simulation time as of the beginning of the first detected period (t) and the duration of the last detected period (t_p). Finally variable $\overline{t_p}$ maintain the average duration of all (so far) detected periods while $s^2_{t_p}$ stores the variance (or variability) of duration (i.e. how far the duration of each detected period is distant from its average value computed along a trajectory) of all (so far) detected periods.

Theorem 2. If a trace σ_A is noisy periodic w.r.t. amplitude levels $L, H \in \mathbb{N}$ then it is

accepted by automaton $\mathcal{A}_{per}$ with parameters L, H and $N \in \mathbb{N}$.

Proof. See [Bal15b]. □

HASL expressions associated to $\mathcal{A}_{per}$. We define different HASL expressions to be associated to automaton $\mathcal{A}_{per}$.

- $Z_1 \equiv E[last(\bar{t}_p)]$: corresponding to the mean value of the period duration for the first N detected periods.
- $Z_2 \equiv PDF(\bar{t}_p, s, l, h)$: corresponding to the PDF of the average period duration over the first N detected periods, where $[l, h]$ represents the considered support of the estimated PDF, and $[l, h]$ is discretized into uniform subintervals of width s
- $Z_3 \equiv E[last(s^2_{t_p})]$: corresponding to the variance of the period duration.

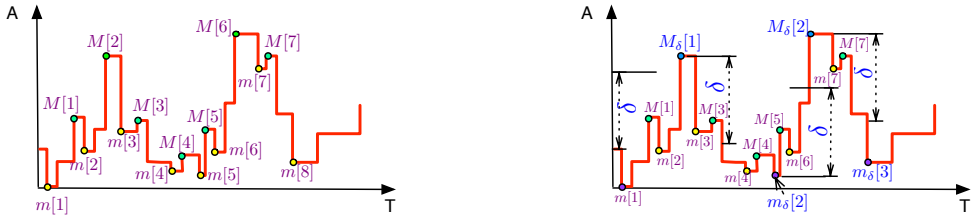
Expression Z_1 represents the expected value assumed by variable $\bar{t}_p$, that is, the average duration of the first N periods detected along a trace, at the end of accepted trajectory (i.e., a trajectory that contains N periods). Similarly expression Z_2 evaluates the PDF of the average duration of the first N periods by assuming the interval $[l, h]$ as the support of the PDF and considering that $[l, h]$ is discretised in $(h-l)/s$ uniform subintervals of width s. On the other hand Z_3 is concerned with assessing the expected value that variable $s^2_{t_p}$ has at the end of a trace consisting of N noisy periods. By definition (see Table 4.1) $s^2_{t_p}$ corresponds to the *variance* of the duration of the detected periods, (i.e., how much the N periods detected along a trace differ from their average duration). Observe that the measured period variance (i.e. Z_3) provides us with a useful measure of the *irregularity*, from the point of view of the period duration, of the observed oscillation.

4.2.3 Peaks-detection based noisy periodicity

The drawback of the partition based period detection scheme discussed in the previous section is that, the detected periods depend on the chosen L, H thresholds, and these have to be chosen by the modeller manually beforehand, i.e., normally by looking at the shape of a sampled trajectory and then choosing where to "reasonably" set the L and H values before executing the measurements with automaton $\mathcal{A}_{per}$. To improve things here we propose a different approach which is aimed at identifying where the peaks (i.e., the local maxima/minima) of oscillatory traces are located.

Since traces of a DESP consist of discrete increments/decrements of at least one unit, it is up to the observer to establish what should be accounted for as a local maximum (minimum) during such detection process. Intuitively a local max/min of a trace σ_A is a state $\sigma_A[i]$ ($i \in \mathbb{N}$) that corresponds to a change of trend in the population of A. This is formally captured by the following definition.

Definition 4.5 (local maximum/mininimum of a trace)**.** For σ_A the projection w.r.t. to species A of a path σ of an n-dimensional DESP $\mathcal{D}$ population model, state $\sigma_A[i]$ is a maximum, if $\sigma_A[i-1] < \sigma_A[i] > \sigma_A[i+1]$, or a minimum, if $\sigma_A[i-1] > \sigma_A[i] < \sigma_A[i+1]$.



(a) local maxima ($M[i]$) and minima ($m[i]$) of an alternating trajectory σ_A.

(b) δ-separated local maxima ($M_\delta[i]$) and minima ($m_\delta[i]$) of an alternating trajectory σ_A.

Figure 4.4: Local maxima/minima and δ-separated local maxima/minima.

In the remainder we refer to a trace that consists of an infinite sequence of local maxima interleaved with an infinite sequence of local minima as an *alternating trace* (Definition 4.6).

Definition 4.6 (alternating trajectory)**.** A trajectory σ of an n-dimensional DESP $\mathcal{D}$ population model is said *alternating* with respect to the i^{th} $(1 \leq i \leq n)$ observed species of $\mathcal{D}$, if σ_i contains infinitely many local minima (or equivalently local maxima).

For σ_A an alternating trace we denote $\sigma_A^M = M[1], M[2], \ldots$, respectively $\sigma_A^m = m[1], m[2], \ldots$, the projection of σ_A consisting of the local maxima, respectively minima, of σ_A. Figure 4.4(a) shows the local maxima and minima for an example of alternating trace σ_A. In the following we point out two simple properties relating the definition of noisy periodic and alternating trace.

Proposition 5. If σ_A is a *noisy periodic* trace (as of Definition 4.1) then it is also *alternating*.

Proposition 5 is trivially true as by definition a noisy period trace visit infinitely often the *low* and *high* region of the state space, thus necessarily it contains an infinite sequence local maxima interleaved with local minima.

Proposition 6. If σ_A is an *alternating* trace (as of Definition 4.6) then it is not necessarily *noisy periodic*.

Proposition 6 simply points out that, by definition, an alternating trajectory may be diverging (for example if it consist of increasing steps which are always larger than the decreasing ones), in which case clearly it is not noisy periodic.

In the remainder we introduce a HASL based procedure for detecting the local maxima and local minima of alternating traces. However rather than considering detection of "simple" local maxima/minima as in Defintion 4.5, we refer to detection of a generalised notion of local maxima/minima of a trace, that is, maxima and minima which are distanced, at least, by a certain value δ. We formalise this notion in the next definition.

Definition 4.7 (δ-separated local maxima)**.** Let $\delta \in \mathbb{R}^+$, and σ_A the A projection of a trace σ of an n-dimensional DESP $\mathcal{D}$ population model. A state $\sigma_A[i]$ is the j-th, $j \in \mathbb{N}_{>0}$, δ-separated local maximum (minimum), denoted $M_\delta[j]$ ($m_\delta[i]$), if *i)* it is the largest local maximum (smallest local minimum) whose distance from the preceding δ-separated local minimum $m_\delta[j-1]$ (maximum $M_\delta[j-1]$) is at least δ and *ii)* from it, it is possible to reach a smaller (larger) δ-separated state.

To understand the meaning of δ-separated minima and maxima definition let us consider the example in Figure 4.4(b). The nature of the first δ-separated point (either a maximum or a minimum) depends on the initial profile of the trace: if the trace, from its initial state $\sigma_A[0]$, first enters the region $\sigma_A[0] + \delta$ than the first δ-separated point will be a maximum (conversely if it first enters the $\sigma_A[0] - \delta$ region it will be a minimum). For the trace in Figure 4.4(b) the first δ-separated point is the maximum $M_\delta[1]$ since none of the preceding local minima $m[1]$,$m[2]$ and local maximum $M[1]$ is sufficiently far apart from the initial state $\sigma_A[0]$ (i.e. $m[1]$,$m[2]$ and $M[1]$ have distance less than δ from $\sigma_A[0]$). Observe that the detection of $M_\delta[1]$ as the actual first δ-separated maximum is completed only on entering the ($M_\delta[1]$–δ) region, which happens on observing the downward jump that follows local maximum $M[3]$. To better understand that detection of δ-separated minima/maxima may involve a "temporary detection" phase followed by a "detection finalisation" let us continue the unfolding of the trace. Having entered the $M_\delta[1] - \delta$ region we encounter the local minimum $m[4]$, which is temporarily detected as the first δ-separated minimum, since in fact it is distanced more than δ from $M_\delta[1]$. However $m[4]$ is followed by $m[5]$ a smaller local minimum reached without exiting the ($m[4]+\delta$) region. Thus $m[5]$ replaces $m[4]$ as temporary first δ-separated minimum. $m[5]$ becomes the actual first δ-separated minimum only on entering the ($m[5]+\delta$) region (which in this case corresponds with reaching of state $M_\delta[2]$) since no smaller local minima has been detected before entering ($m[5]+\delta$).

Definition 4.8 (δ-separated alternating trace)**.** A trajectory σ of an n-dimensional DESP $\mathcal{D}$ population model is said δ-separated *alternating* with respect to the i^{th} ($1 \leq i \leq n$) observed species of $\mathcal{D}$, if it contains contains infinitely many δ-separated local minima (and equivalently δ-separated local maxima), where $\delta \in \mathbb{R}^+$.

For σ_A a δ-separated alternating trace we denote $\sigma_A^{M_\delta} = M_\delta[1], M_\delta[2], \ldots$, respectively $\sigma_A^{m_\delta} = m_\delta[1], m_\delta[2], \ldots$, the projection of σ_A consisting of the δ-separated local maxima, respectively minima, of σ_A. Observe that the δ-separated max/min (Figure 4.4(b)) are a subset of the "simple" max/min (Figure 4.4(a)). Furthermore the following property holds:

Proposition 7. For $\delta = 1$ the sequence of δ-separated maxima (minima) of an alternating trace σ_A coincides with the list of local maxima (minima), that is: $\sigma_A^{M_1} = \sigma_A^M$ and $\sigma_A^{m_1} = \sigma_A^m$.

The detection of the δ-separated local maxima (minima) for a trace σ_A can be described in terms of an iterative procedure through which the list of detected max/min are constructed as σ_A unfolds. Such a procedure is formally implemented by the LHA $\mathcal{A}_{peaks}$ (Figure 4.5) which we introduce later on. Here, based on the example illustrated in Figure 4.4(b), we informally summarise how detection of δ-separated max/min works. The detection requires storing of the most recent (temporary) δ-separated max (min) into a variable named x_M (x_m), while once detection of a δ-separated maximum (minimum) is completed the corresponding variable x_M (x_m) is copied into a dedicated list, named $Lmax$, resp. $Lmin$ (see Table 4.2), which is filled in with the detected δ-separated max/min points. To understand how detection through automaton $\mathcal{A}_{peaks}$ works let us consider again the sample path in Figure 4.4(b). The first element encountered is the local minimum $m[1]$ which is then stored into $x_m = m[1]$. As the trace further unfolds the subsequent maxima (green points) are ignored as long as their distance from the temporary minimum x_m is less than δ, as is the case with $M[1]$. Similarly any local minimum $m[i]$ (yellow point) that is encountered after that stored in x_m is ignored (e.g., $m[2]$), unless it is smaller than x_m, in which case x_m is updated with the newly found smaller minimum. As σ_A unfolding proceeds we find the next local max $M[2]$ which is distant more than δ from the temporary minimum x_m: this means that x_m currently holds an actual δ-distanced minimum hence its value is appended to $Lmin$ and the procedure starts over, in a symmetric fashion, for the detection of the next maximum.

The rational behind the notion of δ-separated max/min is that for locating the actual peaks of a stochastically oscillating trace it is important to be able to distinguish between the minimal peaks corresponding to stochastic noise, the actual peaks of oscillation. With the δ-separated max/min characterisation we provide the modeller with a means to establish an *observational perspective*: by choosing a specific value for the δ parameters the modeller establishes how big a level of noise he/she wants to ignore when detecting where the oscillation peaks are located.

In the remainder we introduce the LHA $\mathcal{A}_{peaks}$ which formally implements the detection of the δ-separated peaks of alternating traces.

The automaton $\mathcal{A}_{peaks}$. We introduce an LHA, denoted $\mathcal{A}_{peaks}$ (Figure 4.5), designed for detecting δ-distanced local maxima/minima along alternating traces of a given observed species called A. It requires a parameter δ (the chosen noise level) and the partition of the event set $E = E_{+A} \cup E_{-A} \cup E_{=A}$ where E_{+A} (respectively $E_{-A}, E_{=A}$) is the set of events resulting in an increase (respectively decrease, no effect) of the population of A.

The rationale behind the structure of $\mathcal{A}_{peaks}$ is to mimic the cyclic structure of an alternating trace through a loop of four locations, two of which (i.e. **Max** and **Min**) are targeted to the detection of local maxima, resp. minima. The simulated trace yields the

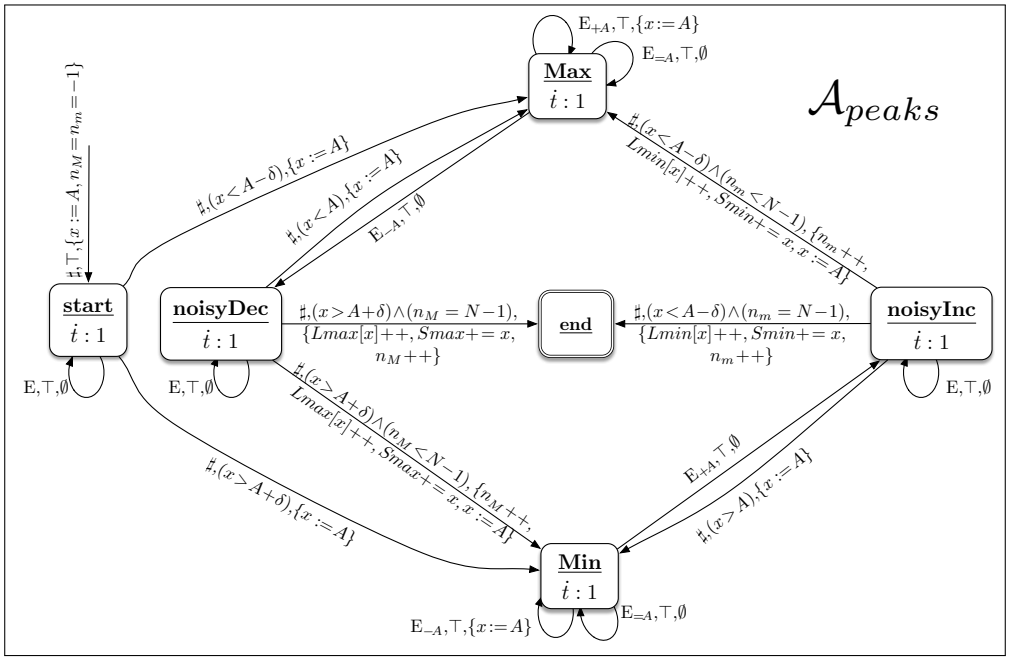


Figure 4.5: The $\mathcal{A}_{peaks}$ LHA for detecting local maxima/minima (for observed species A) of noisy periodic traces where local maxima/minima are detected with respect to a chosen level of noise δ.

automaton to loop between **Max** and **Min** hence registering the minima/maxima while doing so. The detailed behavior of $\mathcal{A}_{peaks}$ is as follows. Processing of a trace starts with a configurable filter of the initial transient (represented as a box in Figure 4.5) through which a simulated trace is simply let unfolding for a given $initT$ duration.The actual analysis begins in location **start** from which we move to either **Max** or **Min** depending whether we initially observe an increase (i.e. $x < A-\delta$) or a decrease (i.e. $x > A+\delta$) of the population of the observed species A beyond the chosen level of noise δ. Once within the **Max**→**noisyDec** →**Min**→ **noisyInc** loop the detection of local maxima and minima begins. Location **Max** (**Min**) is entered from **noisyInc** (**noisyDec**) each time a sufficiently large (w.r.t. δ) increment (decrement) of A is observed. On entering **Max** (**Min**), we are sure that the current value of A has moved up (down) of at least δ from the last value stored in x while in **Min** (**Max**), hence that value (x) is an actual local minimum (maximum) thus we add it up to $Smin$ ($Smax$), then we increment the frequency counter corresponding to the level of the detected minimum $Lmin[x]$ (maximum $Lmax[x]$)[5] before storing the new value of A in x and finally increase n_M (n_m) the

[5]with a slight abuse of notation we refer to $Lmin[]$ and $Lmax[]$ as arrays whereas in reality within COSMOS/HASL they correspond to a set of variables $Lmin_i$, $Lmax_j$, each of which is associated to a

Data variables			
name	domain	update definition	description
t	$\mathbb{R}_{\geq 0}$	*reset*	time elapsed since beginning measure (first non-spurious period)
$n_M(n_m)$	$\mathbb{N}$	*increment*	counter of detected local maxima (n_M), minima (n_m)
x	$\mathbb{N}$	*current value of* observed species A	(overloaded) variable storing most recent detected maximum/minumum
$Smax(Smin)$	$\mathbb{N}$		sum of detected maxima (minima)
$Lmax[](Lmin[])$	$\mathbb{N}^n$		array of frequency of heights of detected maxima (minima)

Table 4.2: The data variables of automaton $\mathcal{A}_{peaks}$ of Figure 4.5 for locating the peaks of a noisy oscillatory traces

counter of detected maxima (minima). Once in **Max** (**Min**) we stay there as long as we observe the occurrence of reactions which do not decrease (increase) the value of A, hence either reactions of E_{+A} (E_{-A}) or of $E_{=A}$. While in **Max** (**Min**) if we observe a reaction of E_{+A} (E_{-A}), then we store the new increased (decreased) value of A in x, which becomes the new potential next local maximum (minimum). On the other hand while in **Max** (**Min**) if a "decreasing" ("increasing") reaction E_{-A} (E_{+A}) occurs we move to **noisyDec** (**noisyInc**) from which we can either move back to **Max** (**Min**), if we observe a new increase (decrease) that makes the population of A overpass x (x overpass A), or eventually entering **Min** (**Max**) as soon as the observed decrease (increase) goes beyond the chosen δ (see above). For the automaton $\mathcal{A}_{peaks}$ depicted in Figure 4.5, the analysis of the simulated trace ends, by entering the **end** location either from **noisyDec** or **noisyInc**, as soon as N maxima (or minima, depending on whether the first observed peak was a maximum or a minimum) have been detected. Notice that $\mathcal{A}_{peaks}$ can straightforwardly be adapted to different ending conditions. The data variables of $\mathcal{A}_{peaks}$ are summarised in Table 4.2.

HASL expressions associated with $\mathcal{A}_{peaks}$. We define different HASL expressions to be associated with automaton $\mathcal{A}_{peaks}$.

given level of the observed population, thus $Lmin_1$ counts the frequency of observed minimum at value 1, $Lmin_2$ the observed minima at value 2 and so on. The number of required $Lmin_i$, $Lmax_j$ variables, which is potentially infinite, can be actually bounded without loss of precision to a sufficiently large value $Lmin_m$ (*resp.* $Lmax_m$) which must be established manually beforehand, for example by observing few previously generated traces.

- $Z_{max} \equiv E[last(Smax)/n_M]$: corresponding to the expected value of the average height of the maximal peaks for the first N detected maxima.
- $Z_{min} \equiv E[last(Smin)/n_m]$: same as Z_{max} but for minima.
- $Z_{PDFmax} \equiv E(last(Lmax)/n_M)$: enabling to compute the PDF of the height (along a path) of the maximal peaks
- $Z_{PDFmin} \equiv E(last(Lmin)/n_m)$: enabling to compute the PDF of the height (along a path) of the maximal peaks

Expression Z_{max} (Z_{min}) represents the average value of the detected δ-separated maxima (minima). This is obtained by considering the sum of all detected δ-separated local maxima (minima), which is stored in $Smax$ ($Smin$) and dividing it by the number of detected maxima n_m (n_m). Expression Z_{PDFmax} (Z_{PDFmin}) allows to estimate the PDF of the height of the detected δ-separated local maxima (minima). This is achieved by dividing the frequency counters of each detected maximal (minimal) peak's height, whose values are stored in array $Lmax$ ($Lmin$), by n_M (n_m), the number of detected maxima (minima).

Theorem 3. If σ_A is a δ-separated (with $\delta \in \mathbb{R}^+$) alternating trace then it is accepted by automaton $\mathcal{A}_{peaks}$ with parameters δ and $N \in \mathbb{N}$.

Proof. See [Bal15a]. □

4.2.4 Case study: circadian clock

In order to demonstrate the HASL based analysis of stochastic oscillator we introduced through the $\mathcal{A}_{per}$ et $\mathcal{A}_{peaks}$ automata we considered a popular example of oscillator, the so-called *circadian clock*, a biological mechanism responsible for keeping track of daily cycles of light and darkness. Specifically the circadian clock model we considered is that proposed by Vilar *et al.* [VKBL02]: it consists of a genetic circuit responsible for the expression of two co-related proteins schematically depicted in Figure 4.6 and formally expressed by 16 chemical reactions in Equation (4.10). Protein expression is a two steps process: in the first phase a gene transcribes a messenger RNA (mRNA) molecule; in the second phase the mRNA molecule is translated into the target protein. For the model of circadian clock we consider here we denote M_A, the mRNA species transcribed by gene D_A, and M_R the mRNA transcribed by gene D_R. M_A and M_R are then translated into proteins A,

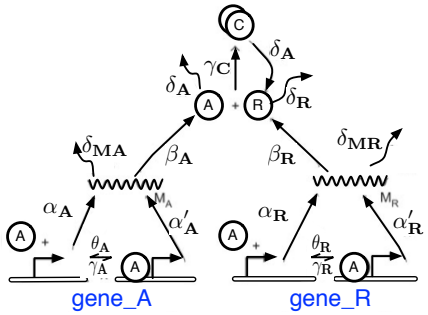


Figure 4.6: Circadian Clock oscillator network: gene D_A expresses activator protein A through transcription of mRNA M_A, while gene D_R expresses the repressor protein R through transcription of mRNA M_R.

respectively R.

$$
\begin{aligned}
R_1 &: A + D_A \xrightarrow{\gamma_A} D'_A & \qquad R_9 &: M_A \xrightarrow{\beta_A} M_A + A \\
R_2 &: D'_A \xrightarrow{\theta_a} A + D_A & \qquad R_{10} &: M_R \xrightarrow{\beta_R} M_R + R \\
R_3 &: A + D_R \xrightarrow{\gamma_R} D'_R & \qquad R_{11} &: A + R \xrightarrow{\gamma_C} C \\
R_4 &: D'_R \xrightarrow{\theta_R} D_R + A & \qquad R_{12} &: C \xrightarrow{\delta_A} R \\
R_5 &: D'_A \xrightarrow{\alpha'_A} M_A + D'_A & \qquad R_{13} &: A \xrightarrow{\delta_A} \emptyset \\
R_6 &: D_A \xrightarrow{\alpha_A} M_A + D_A & \qquad R_{14} &: R \xrightarrow{\delta_R} \emptyset \\
R_7 &: D'_R \xrightarrow{\alpha'_R} M_R + D'_R & \qquad R_{15} &: M_A \xrightarrow{\delta_{M_A}} \emptyset \\
R_8 &: D_R \xrightarrow{\alpha_R} M_R + D_R & \qquad R_{16} &: M_R \xrightarrow{\delta_{M_R}} \emptyset
\end{aligned}
\tag{4.7}
$$

Protein A acts as an *activator* for both genes by attaching to promoter region of D_A and D_R (i.e. when A is attached to a gene the mRNA transcription increases). Species D'_A and D'_R represent the state of gene D_A, respectively D_B, when an activator molecule (A) is attached to their promoter. Note that gene D_R acts as a *repressor* of D_A since when A bounds to its promoter D_R sequesters the activator A and, as a result, the transcription of D_A slows down. The repressing role of D_R is further due to the fact that the expressed protein R inactivates the activator A by binding to it and forming the complex C. Finally the model in Figure 4.6 accounts for degradation of all species: thus the mRNAs M_A and M_R, as well as the expressed proteins A and B degrades with given rates (see Table 4.3). Notice that protein A degrades also when attached to R (i.e. when in complex C), and, as a consequence, C turns into R at a rate equivalent to the degradation rate of A.

α_A	$50\ h^{-1}$	α_R	$0.01\ h^{-1}$	δ_A	$1\ h^{-1}$	δ_R	$0.2\ h^{-1}$
$\alpha_{A'}$	$500\ h^{-1}$	$\alpha_{R'}$	$50\ h^{-1}$	$\gamma_A = \gamma_R$	$1\ mol^{-1}h^{-1}$	γ_C	$2\ mol^{-1}h^{-1}$
β_A	$50\ h^{-1}$	β_R	$5\ h^{-1}$	θ_A	$50\ h^{-1}$	θ_R	$100\ h^{-1}$
δ_{MA}	$10\ h^{-1}$	δ_{MR}	$0.5\ h^{-1}$				

Table 4.3: reactions' rates for the circadian oscillator

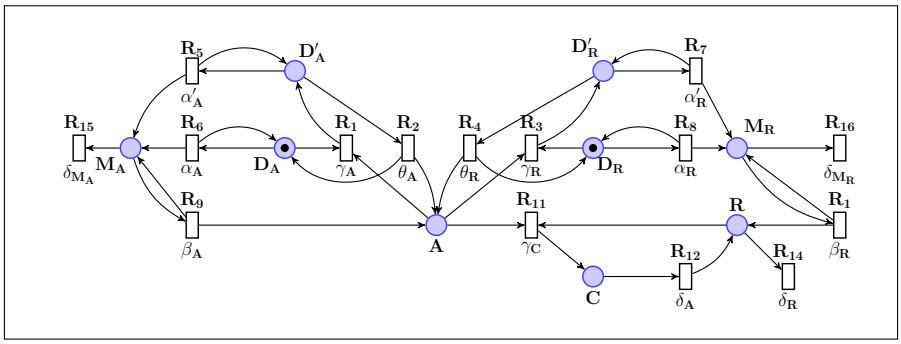


Figure 4.7: GSPN encoding of the system (4.10) of chemical equations corresponding to the circadian-clock.

The model of Figure 4.6 corresponds to the system of chemical equations (4.10), whose (continuous) kinetic rates (taken from [VKBL02]) are given in Table 4.3.

Stochastic model Equations (4.10) can give rise to either a system of ODEs or to a stochastic process. Here we focus on the discrete-stochastic semantics: Figure 4.7 shows the GSPN encoding of equations (4.10) developed with COSMOS. The configuration of the GSPN (i.e. the stochastic process) requires setting the initial population and the rates of each transition (i.e. reaction). For the initial population, following [VKBL02], we observe that the model comprises one gene D_A and one D_R, which can either be in free-state (no activator A is attached to the promoter) or in activator-bound state, i.e. D'_A, respectively D'_R. As a consequence the population of species D_A and D_R is bounded by the following invariant constraints: $D_A + D'_A = 1$ and $D_R + D'_R = 1$ (in fact places DA, DA' and DR, DR' of net in Figure 4.7 are the only places covered by P-invarriants). The remaining species are initially supposed to be "empty", hence they are initialised to 0. Concerning the transition rates, for simplicity we assume a unitary volume of the system under consideration, hence all continuous rates in Table 4.3 can be used straightforwardly as rates of the corresponding discrete-stochastic reactions. In this case we assume all reactions following a negative exponential law.

The oscillatory dynamics of the GSPN model of Figure 4.7 can be observed by plotting of a simulated trajectory (Figure 4.8). Observe that the frequency of oscillations varies considerably with the degradation rate of the repressor (R) protein: a faster degradation

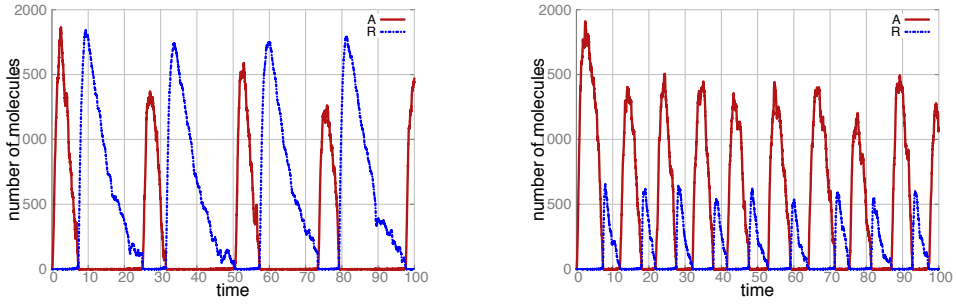


Figure 4.8: Single trajectory showing the oscillatory character of activator A and repressor B dynamics with normal repressor's degradation rate $\delta_R = 0.2$ (left) and with $10\times$ speed-up, i.e. $\delta_R = 2$ (right).

of R (right), intuitively, results in a higher frequency of oscillations. In the remainder we formally assess the oscillatory characteristics (i.e. the period and the peaks of oscillations) of the circadian clock model by application of the previously described approach, i.e. by analysing the stochastic process deriving from synchronisation of the circadian clock GSPN model with the $\mathcal{A}_{per}$ and $\mathcal{A}_{peaks}$ automata.

Measuring the period of the circadian clock. We performed a number of experiments aimed at assessing the effect that the degradation rate of the repressor protein (δ_R) has on the period of the circadian oscillator. Figure 4.9 (right) shows three plots representing the PDF of the period (obtained through the HASL formula $(\mathcal{A}_{per}, PDF(Last(t)/N))$ for three values of δ_R. With $\delta_R = 0.2$ (i.e. the original value as given in [VKBL02]) the PDF is centred at $t = 24.9$, i.e. slightly more of the standard 24 hours period expected for a circadian clock. On the other hand speeding up the repressor degradation of 10 times (i.e. $\delta_R = 2$) yields a slightly more than halved oscillation period (i.e. PDF centred at $T = 10.8$). Finally slowing down the degradation rate of a half (i.e. $\delta_R = 0.1$) yields a less than doubled oscillation period (i.e. PDF centred at $T = 40.7$). Notice that plots in Figure 4.9 (right) also indicate that slowing down of the degradation of the repressor R affects as well the variability of the oscillation period as witnessed by the increasing width of the bell-shape form of the PDF plots in Figure 4.9 (right).

Figure 4.9 (left) shows plots for the period mean value (red plot) and the period variance (blue plot) obtained through by iterative evaluation of the HASL formulae $(\mathcal{A}_{per}, E[last(\bar{t}_p)])$, respectively, $(\mathcal{A}_{per}, E[last(s^2_{t_p})])$ in function of the degradation rate δ_R. They indicate that slowing down the degradation of the repressor yields, on one hand, to a lower the period of oscillations, and on the other, augmenting the irregularity of the periods (i.e. augmenting the period's variances). All plots in Figure 4.9 result from sampling of

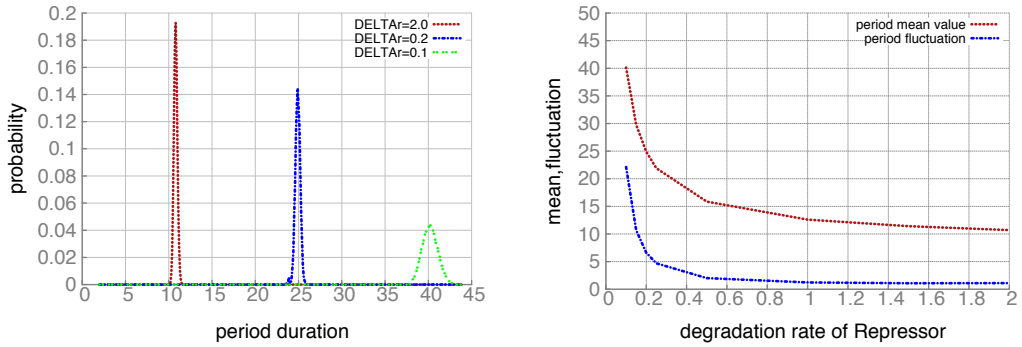


Figure 4.9: The PDF (left) and the mean value vs the variance (right) of the period of oscillations of protein A of the circadian clock measured with $\mathcal{A}_{per}$ in function of the repressor's degradation rate.

finite trajectories consisting of $N = 100$ periods, where periods have been detected using $L = 1$ and $H = 1000$ as partition thresholds, and target estimates have been computed with confidence level 99 and confidence-interval width of 0.01. Furthermore the PDF plots in Figure 4.9 (right) have been computed using a discretisation of the period support interval $[0, 50]$ into subintervals of width 0.1.

Measuring the peaks of oscillations of the circadian clock. We performed a number of experiments aimed at assessing the effect that the degradation rate of the repressor protein (δ_R) has on the peaks of oscillation for both protein A and R. Figure 4.10 shows plots for the mean value of the minimal and maximal peaks of oscillations for both A and R. Results indicate that while the degradation rate δ_R has no effect on the oscillation peaks of A (both maximal and minimal peaks of A are constant independently of δ_R), it affects the maximal peaks (only) of R. Specifically the mean value of R's maximal peaks decreases with the increasing of δ_R (while the minimal peaks of R are constantly at 0), notice that this is in agreement with what indicated by the single trajectories depicted in Figure 4.8, as these show that the oscillations of both A and R consist of positive half-period (where the population is positive) interleaved by zero half-period (where the population is zero). Notice that Figure 4.10 contains also plot for the absolute maximum of population of A and R measured along the sampled trajectories through trivial HASL expressions $Z \equiv AVG(max(x))$ (where x is a variable used to record the population of the observed species along a synchronising path). All plots in Figure 4.10 result from sampling of finite trajectories containing of $N = 100$ maximal peaks and using a noise parameter $\delta = 10\% AVG(max(x))$, meaning that for evaluating the mean value of maximal peaks we discarded all critical points (i.e. the noise spikes) distanced one another less than 10%

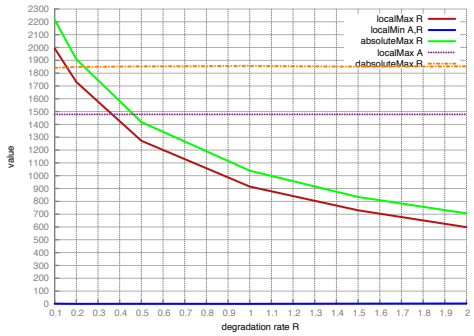


Figure 4.10: The mean value of the minimal and maximal peaks of proteins A and R of the circadian clock measured with $\mathcal{A}_{peaks}$ in function of the repressor's degradation rate.

of the absolute maximum of the observed species. Finally, again points of every plot in Figure 4.10 have been computed with confidence level 99 and confidence-interval width of 0.01.

4.3 Tuning stochastic oscillators

The second contribution described in this chapter is that of a framework for tuning of stochastic oscillators, that is, a framework that allows for searching the parameter space of an oscillator model so to identify the parameter's values that comply with a target oscillation period. Such framework combines the partition based HASL characterisation of noisy periodicity outlined in the first part of this chapter with the Automaton-ABC parameter inference framework outlined in Chapter 3. As we have seen the adaptation of the ABC parameter inference scheme to HASL relies on the existence of a notion of *distance* between the paths sampled from a given instance of a model and the sought (target) behaviour. Therefore, in this case, we need to identify a suitable *distance* that can effectively drive the inference of parameters according to a desired oscillation character, in this case the oscillation period.

Definition 4.9 (distance from target period)**.** For σ_A a noisy periodic trajectory and $\bar{t_p}^{(obs)} \in \mathbb{R}_{>0}$ a target mean period duration we define the distance of σ_A from $\bar{t_p}^{(obs)}$ w.r.t. the first $n \in \mathbb{N}$ period realisations as

$$\mathrm{dist}(\sigma_A, n, \bar{t_p}^{(obs)}) = \mathrm{dist}(\bar{t_p}(n), s^2_{t_p}(n), \bar{t_p}^{(obs)}) = \min\left(\frac{\bar{t_p}(n) - \bar{t_p}^{(obs)}}{\bar{t_p}^{(obs)}}, \frac{\sqrt{s^2_{t_p}(n)}}{\bar{t_p}^{(obs)}}\right) \tag{4.8}$$

where $\bar{t_p}(n)$ ($s^2_{t_p}(n)$) denotes the sample mean value (the variance) computed w.r.t. the first n periods detected along σ_A (as per Definition 4.2 and Definition 4.4).

Notice that distance (4.8) establishes a form of multi-criteria selection of parameters as both the mean value and the variance of the detected periods are constrained. For example with a 10% tolerance (i.e. $\epsilon = 0.1$ in ABC terms) only the parameters $\theta \in \Theta$ that issue a relative error (w.r.t. the target mean period) not beyond 0.1 are selected.

4.3.1 An automaton for the distance from a target period

In Figure 4.11 we introduce the automaton $\mathcal{A}^{\bar{t_p}^{(obs)}}_{per}$, i.e, an adaptation of the $\mathcal{A}_{per}$ (Figure 4.3), suitable for assessing the distance (as per Definition 4.9) between the mean period measured on paths of DESP oscillator $\mathcal{M}_\theta$ and a target period duration $\bar{t_p}^{(obs)}$. The au-

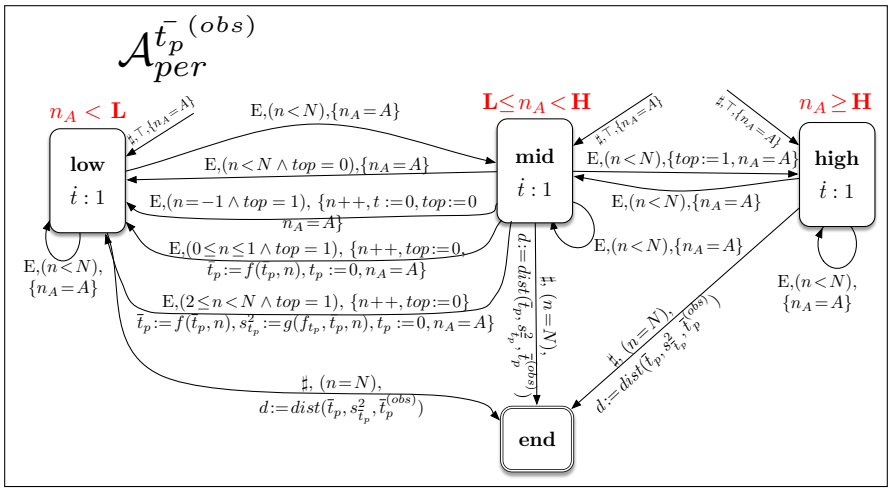


Figure 4.11: $\mathcal{A}_{per}$: an LHA for selecting noisy periodic traces (with respect to an observed species A) related to partition $low = (-\infty, L]$, $mid = (L, H)$ and $high = [H, +\infty)$.

tomaton $\mathcal{A}^{\bar{t_p}^{(obs)}}_{per}$ differs from $\mathcal{A}_{per}$ only in the autonomous arcs that leads to the accepting location **end**: differently from $\mathcal{A}_{per}$ those arcs are equipped with an update through which a dedicated extra varaible, named d, is set with the value of the measured distance as per Equation (4.8) (see Table 4.4 for the list of variables).

4.3.2 Case studies

We demonstrate the HASL-ABC for tuning stochastic oscillator through a couple of case studies.

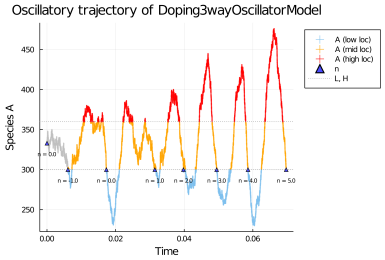
name	domain	update definition	description
n	$\mathbb{N}$	*increment*	counter of detected periods
t_p	$\mathbb{R}_{\geq 0}$	*reset*	duration last period
$\bar{t}_p$	$\mathbb{R}_{\geq 0}$	$f(\bar{t}_p, t_p, n) = \frac{\bar{t}_{p_n} \cdot n + t_p}{n+1}$	period mean
$s^2_{t_p}$	$\mathbb{R}_{\geq 0}$	$g(s^2_{t_p}, \bar{t}_p, t_p, n) = \frac{n-1}{n-2} \cdot s^2_{t_p} + \frac{(\bar{t}_p - t_p)^2}{n}$	period variance
d	$\mathbb{R}_{\geq 0}$	$\min(\frac{\bar{t_p} - \bar{t_p}^{(obs)}}{\bar{t_p}^{(obs)}}, \frac{\sqrt{s^2_{t_p}}}{\bar{t_p}^{(obs)}})$	distance from target period

Table 4.4: Variables of the $\mathcal{A}_{per}^{\bar{t_p}^{(obs)}}$ automaton.

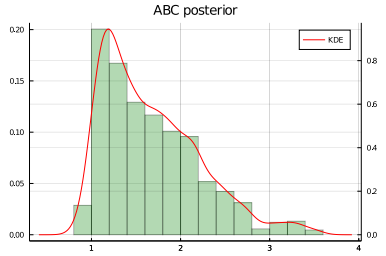
A synthetic 3-ways oscillator

The CRN in (4.9) represents a simple model of sustained oscillator called *doping 3-way oscillator* [Car09].

$$\begin{array}{lll} R_1 : A+B \xrightarrow{r_A} 2B & R_2 : B+C \xrightarrow{r_B} 2C & R_3 : C+A \xrightarrow{r_C} 2A \\ R_4 : D_A+C \xrightarrow{r_C} D_A+A & R_5 : D_B+A \xrightarrow{r_A} D_B+B & R_6 : D_C+B \xrightarrow{r_B} D_C+C \end{array} \tag{4.9}$$



(a) Synchronisation of a path of the 3-way oscillator with $\mathcal{A}_{per}$: coloured segments represent the corresponding $\mathcal{A}_{per}$ location, while n is the period counter variable.



(b) Posterior distribution for the r_A parameter obtained by application of the automaton-ABC with $\mathcal{A}per$.

Figure 4.12: A sampled trajectory of the 3-way oscillatory (left) with coloured highlights of $\mathcal{A}_{per}$ corresponding location transitions and the resulting posterior distribution (right) obtained for r_A while keeping $r_B = r_C = 1$.

It consists of 3 main species A, B and C forming a positive feedback loop (through reactions R_1, R_2, R_3) plus 3 corresponding invariant (*doping*) species D_A, D_B, D_C whose goal is to avoid extinction of the main species (through reactions R_4, R_5, R_6). It can be shown that the total population is invariant and that the model yields sustained noisy oscillation for the 3 main species, whose period and amplitude depend on the model's

parameters r_A, r_B and r_C (as well as on the total population). Figure 4.12(a) depicts a path sampled from (a given configuration of) model (4.9).

Experiment 1 (one-dimensional). In this experiment we considered $\mathbf{s}_0 = (A_0, B_0, C_0, (D_A)_0, (D_B)_0, (D_C)_0) = (333, 333, 333, 10, 10, 10)$ as initial state, we fixed the rate constants $r_B = r_C = 1.0$, and estimated the posterior distribution for r_A considering a uniform $\mathcal{U}(0, 10)$ prior and a target mean period $\bar{t}_p^{(obs)} = 0.01$. For the automaton $\mathcal{A}_{per}$ the noisy-period dependent partition we considered is $L = 300$ and $H = 360$ while for each trajectory we observed $N = 4$ periods. For the ABC algorithm we used $N = 1000$ particles and considered a 20% tolerance ($\epsilon = 0.2$). Figure 4.12(b) shows the resulting automaton-ABC posterior. We observe that 1) the posterior support being included in the prior is $[0.0, 4.0] \subset [0, 10.0]$, i.e., we have reduced the parameter space to a subset where it is probable to obtain trajectories with a mean period of 0.01 (relative to a 20% tolerance) and 2) the posterior has only one mode, which is quite sensible, as having fixed r_B and r_C, one would expected the mean period duration being directly linked to the kinetics of reaction R_1, hence to its only parameter, the kinetic rate constant r_A.

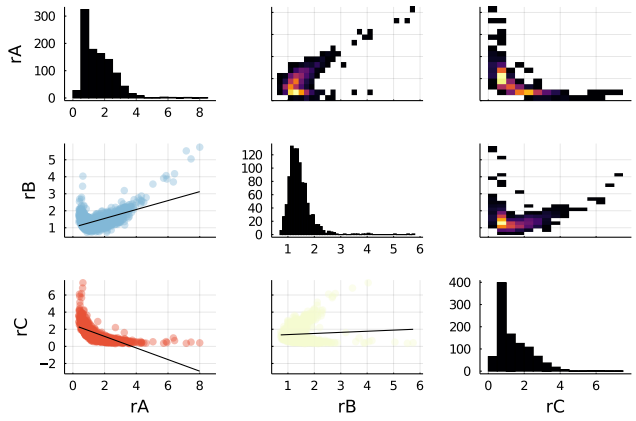


Figure 4.13: Correlation plot of automaton-ABC posterior with $\mathcal{A}per$ posterior for the 3D experiment of doping 3-way oscillator.

.

Experiment 2 (three-dimensional). Here we performed a 3-dimensional version of the previous experiment assuming the same Uniform prior distribution for all 3 parameters, that is, $r_A, r_b, r_C \sim \mathcal{U}(0, 10)$, however considering a different target mean period $\bar{t}_p^{(obs)} = 20$. Figure 4.13 shows the correlation plot of the resulting posterior distribution obtained by application of the automaton-ABC method with $\mathcal{A}_{per}$. The one-dimensional histogram on the diagonal of Figure 4.13 represents the marginal distribution of each parameter while the plots in the upper triangular submatrix show the histograms of the two-dimensional marginal distribution (e.g., plot on position (1,3) refers to the marginal

distribution $p(r_A, r_C|\bar{t_p}^{(obs)})$), whereas the lower triangular matrix show the scatter plots of the two-dimensional marginal distributions. Based on the diagonal plots we observe that most of the parameters that results in a period close to the target one ($\bar{t_p}^{(obs)} = 20$) are within the support $[0,4] \times [0,3] \times [0,4]$ and, furthermore the particles form a 3D parabolic shape since the three two-dimensional projections have a parabolic shape, according to plots in the upper triangular part of Figure 4.13. Also, one can notice that for each two-dimensional histogram, the region near point $(1,1)$ is a high probability one, which is consistent with the previous one-dimensional experiment.

Repressilator

We consider a model (4.10) of a synthetic genetic network known as Repressilator which was developed to reproduce oscillatory behaviours within a cell [EL00]. It consists of 3 proteins P_1, P_2, P_3 forming a negative feedback loop, with P_1 repressing P_2's transcription gene G_2, P_2 repressing P_3's transcription and so on.

$$\begin{array}{lll} R1 : G_1 \xrightarrow{r_1} G_1 + M_1 & R2 : G_2 \xrightarrow{r_2} G_2 + M_2 & R3 : G_3 \xrightarrow{r_3} G_3 + M_3 \\ R4 : M_1 \xrightarrow{\beta} M_1 + P_1 & R5 : M_2 \xrightarrow{\beta} M_2 + P_2 & R6 : M_3 \xrightarrow{\beta} M_3 + P_3 \\ R7 : M_1 \xrightarrow{1} \emptyset & R8 : M_2 \xrightarrow{1} \emptyset & R9 : M_3 \xrightarrow{1} \emptyset \\ R10 : P_1 \xrightarrow{1} \emptyset & R11 : P_2 \xrightarrow{1} \emptyset & R12 : P_3 \xrightarrow{1} \emptyset \end{array} \tag{4.10}$$

Following [EL00] we assumed mass-action dynamics with a common rate constant β for translations reactions (R_4, R_5, R_6) and constant rate 1 for species degradation (R_7 to R_{12}). Conversely transcription dynamics $(R1, R_2, R_3)$ is assumed to follow a Hill function dynamics given by the follow parameter definitions $r_1 = \frac{\alpha}{1+[P_3]^n} + \alpha_0$, $r_2 = \frac{\alpha}{1+[P_1]^n} + \alpha_0$ and $r_3 = \frac{\alpha}{1+[P_2]^n} + \alpha_0$, where n is the Hill coefficient, α is related to transcription growth and α_0 is the parameter related to the minimum level of transcription growth. The parameter space is therefore 4-dimensional with $\theta = (\alpha, \beta, n, \alpha_0) \in \mathbb{R}^4$.

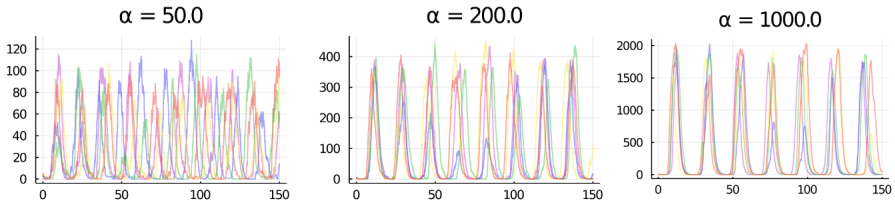


Figure 4.14: P_1-projections of paths simulated with $(\beta, n, \alpha_0) = (2, 2, 0)$ and $\alpha \in \{50, 200, 1000\}$.

Each parameter affects the resulting oscillation, e.g., Figure 4.14 shows examples of noisy periodic paths corresponding to different values of α, with $(\beta, n, \alpha_0) = (2, 2, 0)$.

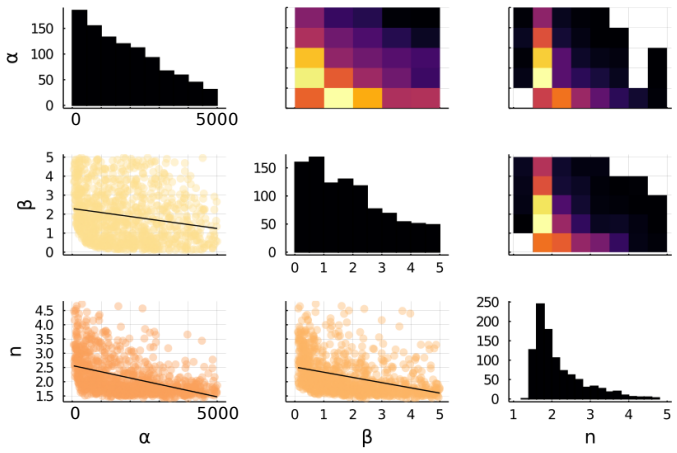


Figure 4.15: Correlation plot of automaton-ABC posterior with $\mathcal{A}per$ posterior for the 3D experiment of repressilator model.

Experiment 1. This is a 3-dimensional experiment in which we considered $\mathbf{s}_0 = ((M_1)_0, (M_2)_0, (M_3)_0, (P_1)_0, (P_2)_0, (P_3)_0) = (0, 0, 0, 5, 0, 15)$ as initial state, we fixed $\alpha_0 = 0$, and estimated the posterior distribution for the remaining parameters considering the following prior $\alpha \sim \mathcal{U}(50, 5000)$, $\beta \sim \mathcal{U}(0.1, 5.0)$, $n \sim \mathcal{U}(0.5, 5.0)$. The target mean period was $\bar{t_p}^{(obs)} = 20$, $L = 50$ and $H = 200$ the noisy-period dependent partition For the ABC algorithm we used $N = 1000$ particles and considered a 10% tolerance ($\epsilon = 0.1$). Figure 4.15 shows the correlation plot of the resulting automaton-ABC posterior. We observe that with this setting the Repressilator oscillations are most sensitive to parameter n whose marginal posterior is much narrower than that of α and β (i.e. varying n induces more instability than α and β).

Experiment 2. This is a 4-dimensional version of the previous experiment in which we considered also a uniform prior for $\alpha_0 \sim \mathcal{U}(0.1, 5.0)$. Figure 4.16 shows the correlation plot of the resulting automaton-ABC posterior. We observe that adding a degree of freedom on α_0 has changed the correlation between α and β as well as α and n whereas correlation between β and n seems to have the same shape.

4.4 Perspectives

As a first contribution in this chapter (Section 4.2) we tackled the problem of using an automata-based formalism to formally analyse stochastic oscillators. In this respect the contribution is twofold. Firstly we have generalised Spieler's time-automata noisy periodicity detection, and shown that through HASL one can assess oscillation related indicators that could not be accounted for otherwise. Secondly we have outlined an alternative

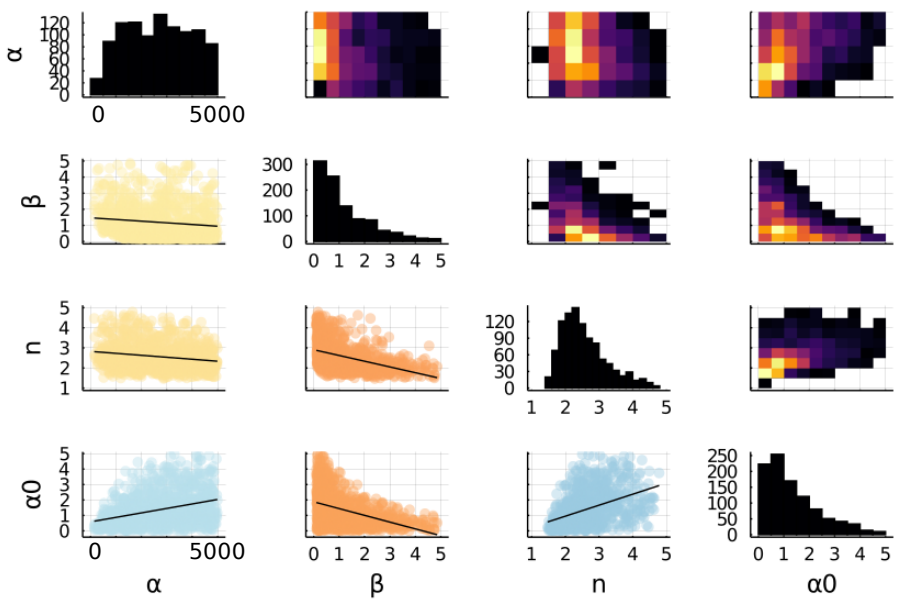


Figure 4.16: Correlation plot of automaton-ABC posterior with $\mathcal{A}per$ posterior for the 4D experiment of repressilator model.

automata-based approach for analysing periodicity which is aimed at identifying the peaks of stochastic oscillations leading to the characterisation of amplitude related indicators. Although the proposed HASL-based framework opens up to gaining detailed insights about the characteristics of a stochastic oscillators, it provides little help for establishing whether the considered system oscillate sustainably or not. As shown by Spieler the existence of sustained oscillation can be established by checking of a steady-state property on the product Markov chain[6], i.e. the ergodic-by-construction Markov chain given by the product of the considered model and the DTA period detector. An interesting perspective would be to investigate whether the HASL framework for oscillation analysis could be adapted to establishing whether the oscillations are sustained. Intuitively that would entail the integration, within the HASL model checking framework, of simulation based methods for the verification of steady-state measures, such as *perfect sampling* [RP09], which indeed is an aspect I am very much willing to investigate. Another interesting evolution of this contribution would be to move towards a full automatisation of periodicity detection. In fact the partition based period detection approach depend upon the (manually) chosen constant parameters L and H that characterise the partition and therefore both the period detection and the measured indicators (period mean value and variance) are very much

[6]the model oscillate sustainably if converging paths of the product process have zero probability at steady-state

affected by the chosen L and H values. As a consequence if one choses too "extreme" L and H values the detection of a period may become a very unlikely (possibly a zero probability) event, therefore, in practice, one must have an idea about the amplitude (and location) of the high and low peaks of the oscillation, in order to correctly set the L and H parameter of the detector automata beforehand. A partition independent detection of the oscillation periods, intuitively, could be worked out following the peaks based period detection approach outlined in Section 4.2.3, although the integration of partition-based with peak-based period detection has not yet been investigated so far.

As a second contribution of this chapter (Section 4.3) we have shown how to adapt the ABC-based parametric verification framework outlined in Chapter 3 to the problem of tuning of stochastic oscillators, that is, to identifying the probability distribution with which the parameters of a stochastic oscillator match a given period of the oscillations. Such contribution currently only take into account the oscillation period as the constraining factor driving the parameter search. An interesting perspective would be to extend this work so that also the oscillation amplitude can be accounted for during the parameter search process, something that intuitively could be achieved based on the amplitude related HASL specifications outlined in Section 4.2.

Chapter 5

Data-driven predictive modeling of periodic phenomena

As argued in the previous chapter many real life phenomena exhibit some form of periodicity, examples of which can be found in different domains e.g., road traffic, patient arrival to an emergency department or photovoltaic energy production. In Chapter 4 we tackled the problem of how to automatically assess (probabilistic) periodicity related indicators of a formally given stochastic oscillator model and also how to tune the parameters of such a model so that it exhibits a desired periodicity. In this chapter we consider a complementary aspect, namely, the derivation of (stochastic) models based on observations of periodic phenomena. We stress that the ability to derive models that accurately reproduce the periodicity exhibited by data samples issued by periodic systems is vital for predictions, planning and finally optimisation of such systems. Here we report on a contribution in the context of optimisation of hospitals' emergency department (ED), for which periodicity is indeed relevant as evidenced by datasets we had access to.

Outline of the contribution. In collaboration with Andras Horváth, Roberto Aringhieri and in the context of the PhD thesis work of Davide Duma (all three of them from University of Turin) we studied the problem of deriving a stochastic model of the patient arrivals (to an ED) that is capable of correctly reproducing the statistical character of a given arrivals dataset. To this aim we considered a real dataset relative to the activity of the ED of an hospital of the city of Cantù in northern Italy and which contains different data including the complete set of timestamps of the arrival of each patients over one year (i.e. the year 2015). Based on the statistical analysis of the dataset, which evidenced the periodic character of the arrival process, we derived different instances of Markovian models belonging to different classes (i.e. Markov renewal processes, Markov arrival processes, hidden Markov models) and showed that only one family of model, amongst those considered, is capable of fully reproducing the periodic statistical character of the dataset. To

further validate the derived models we developed a formal encoding in terms of stochastic Petri nets. This allowed us to compare the performances of each arrival models, coupled with a model of ED services, through assessment of a key performance indicator (KPI) formally encoded and accurately assessed through HASL statistical model checking. The obtained results evidenced that one amongst the derived arrival models faithfully match the dataset behaviour hence it can reliably be used for analysing the performances of different ED services.

Structure of the chapter. The chapter is organised as follows: Section 5.1 reports on the results of statistical analysis of the ED dataset, evidencing the periodic nature of patients arrivals. In Section 5.2 we provide a comparative analysis of how different families of stochastic models, i.e., Markov renewal processes, Markov arrival processes and hidden Markov model, derived from dataset are capable of adequately reproducing the periodic nature of the observations. Finally in Chapter 5.3 we tackled the validation of derived models. To this aim we considered a simple model of ED services (representing the services a ED patient undergoes after triage), we combined it with each derived model as well as a with a model reproducing exactly the dataset arrivals and by assessing relevant performance indicators on the arrivals/service combined models we established that only one kind of derived model is capable of adequately matching the dataset.

5.1 Statistical analysis of an emergency department dataset

The first goal of our study was to gain an understanding on relevant characteristics of the patient's arrival process by analysing the one year timespan dataset we had access to. We considered two perspectives: that of the patients' *inter-arrival time* and that of the patients' *number of arrivals per time frame*, which we represented through different families of random variables given in the following definition.

Definition 5.1. For D a dataset spanning over several weeks time frame and whose entries are equipped with timestamps relative to the occurrence of a given type of event we consider the following families of random variables:

- X_i $(i = 1, 2, 3, ...)$: the i-th inter-arrival time, i.e., the time elapsed between the i-th and $(i+1)$-th occurrence of the considered kind of event.
- $Y_{\mathcal{H},i}, Y_{\mathcal{D},i}$ and $Y_{\mathcal{W},i}$ $(i = 1, 2, 3, ...)$: the number of occurrences of the considered event during the i-th hour ($Y_{\mathcal{H},i}$), day ($Y_{\mathcal{D},i}$), respectively week ($Y_{\mathcal{W},i}$).

Clearly, variables of families $Y_{\mathcal{H},i}, Y_{\mathcal{D},i}$ and $Y_{\mathcal{W},i}$ are less informative than those of family X_i and yet they are relevant as they allow us for easily investigate the periodic nature of the ED's patient arrival process. For example, the variables $Y_{\mathcal{H},3+24j}$ with $j = 0, 1, 2, ...$

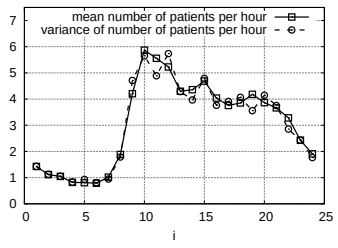


Figure 5.1: Mean and variance of number of arrivals in the i-th hour of the day.

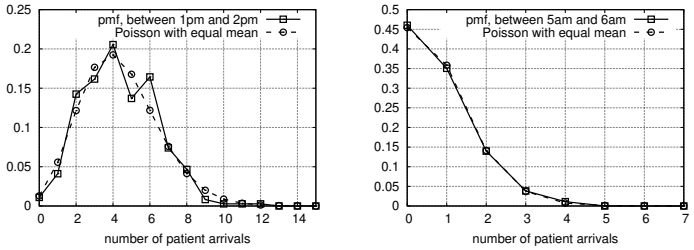


Figure 5.2: Pmf of number of arrivals between 1pm and 2pm (left) and between 5am and 6am (right) and pmf of the Poisson distribution with equal mean.

refer to the number of arrivals between 2am and 3am during the 1st, the 2nd, the 3rd day, and so on.

For both families of variables we computed statistical indicators such as the sample mean, the sample variance, the sample probability distribution and the sample autocorrelation function, the latter aimed at investigating the periodic character of the patient arrival process.

Arrival per hour of the day. Based on variables of the $Y_{\mathcal{H},i}$ family we estimated the mean value $E_{\mathcal{H},i} = E\left[\{Y_{\mathcal{H},i+24j}|j = 0, 1, 2, ...\}\right]$ and the variance $Var_{\mathcal{H},i} = Var\left[\{Y_{\mathcal{H},i+24j}|j = 0, 1, 2, ...\}\right]$ of the number of patient arrivals during the i-th hour of the day which are both depicted in Figure 5.1 in function of i. The highest mean number of patient arrivals is about 5.9 per hour and is registered between 9am and 10am whereas the lowest number of arrivals is instead less than 1 patient per hour and takes place between 5am and 6am: as expected, the time of the day has a strong impact on the number of patient arrivals. We observe that the variance, $Var_{\mathcal{H},i}$, has very similar values to those of the mean $E_{\mathcal{H},i}$, which is compatible with a well-known characteristic of the Poisson process, hence suggesting that the patient arrival process may be adequately modelled through a Poisson process with time inhomogeneous intensity. To further investigate this aspect, we compare in Figure 5.2 the probability distribution of the number of arrivals during a given hour of the day computed over the dataset with the corresponding Poisson distribution i.e. that with the same mean[1]. Plots in Figure 5.2 compare the resulting probability mass function (pmf) for two different time intervals: 1pm-2pm which is the hour of the day where the Poisson distribution differs most[2] from the actual distribution of the number of arrivals and the interval 5am-6am is the hour where the Poisson distribution is most similar to the experimental distribution.

To investigate the periodicity of one-hour intervals we considered the hourly autocor-

[1] the Poisson distribution with mean given by the $E_{\mathcal{H},i}$ for the corresponding hour i.

[2] the difference is calculated as the sum of absolute differences in the pmf.

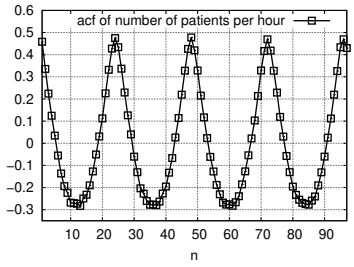


Figure 5.3: $R_{\mathcal{H},n}$, i.e., acf of the sequence $Y_{\mathcal{H},i}$ as function of n.

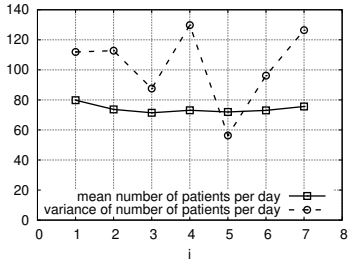


Figure 5.4: Average and variance of number of arrivals in the ith day of the week.

relation function (acf) defined as:

$$R_{\mathcal{H},n} = \frac{E[(Y_{\mathcal{H},i} - E[Y_{\mathcal{H},i}])(Y_{\mathcal{H},i+n} - E[Y_{\mathcal{H},i}])]}{Var[Y_{\mathcal{H},i}]} \quad \text{with } n = 1, 2, 3, ...$$

and whose value is depicted in Figure 5.3 in function of n. The persistent periodicity of the hourly arrivals is proved by the sustained oscillation exhibited by the autocorrelation plot. The periodic nature of the autocorrelation plot is justified by the non-constant hourly patient arrival intensity (as proved by the mean value plot in Figure 5.1) and its sustained character is evidenced by the fact that autocorrelation oscillations remain unaltered even for very large values of n.

Arrival per day of the week. To figure out whether the periodicity is present also w.r.t. the day of the week we performed the same type of analysis but taking into account variables $Y_{\mathcal{D},i}$. In Figure 5.4 we depict $E_{\mathcal{D},i} = E[\{Y_{\mathcal{D},i+7j} | j = 0, 1, 2, ...\}]$ and $Var_{\mathcal{D},i} = Var[\{Y_{\mathcal{D},i+7j} | j = 0, 1, 2, ...\}]$ with $i = 1, 2, ..., 7$ and observe that the mean number of arrivals is essentially constant over the days of the week whereas the variance differs to a large extent from the mean on most days of the week (which makes the daily arrival process incompatible with a Poisson process). Furthermore the autocorrelation calculated also w.r.t. $Y_{\mathcal{D},i}$ (not shown here) exhibited only a very mild correlation between number of arrivals of consecutive days therefore we may assert the absence of periodicity when considering arrivals per day of the week, which indicates we will not have to account for it when developing models aimed at reproducing the statistical character of the dataset.

Inter-arrival analysis. Finally, we considered the family of random variables X_i ($i = 1, 2, 3...$) representing the patients' inter-arrival times and computed the same statistical indicators. The resulting mean inter-arrival time is $E[X_i] = 19.43$ (minutes), the variance $Var[X_i] = 483.18$, while the shape of the probability density function computed over X_i is depicted in Figure 5.5. Figure 5.6 instead depicts the autocorrelation computed for X_i. The damped oscillation profile exhibited by the autocorrelation is due to the fact that

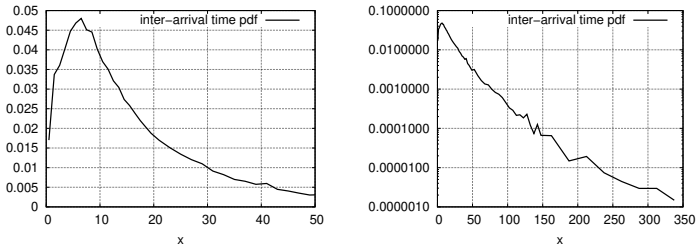


Figure 5.5: Front (left) and tail (right) of the pdf of the inter-arrival times.

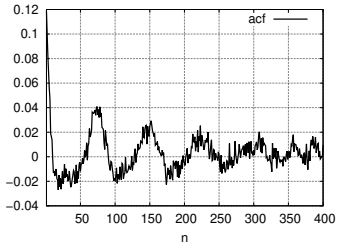


Figure 5.6: Autocorrelation function of the inter-arrival times.

although periodicity exists between the inter-arrivals The autocorrelation present in X_i is depicted in Figure 5.6 where there is oscillation due to the fact that a period with high arrival intensity (8am-22pm) are followed by a period with low arrival intensity (22pm-8am). This oscillation however is less easy to interpret than that in Figure 5.3 and fades away as n grows because for large values of n there is no deterministic connection between the time of the day of the ith and the $(i+n)$th arrival.

5.2 Deriving adequate models of patient arrivals

We seek to derive adequate stochastic models capable of reproducing the statistical character of a dataset (in our case the Cantu hospital ED patients' arrival dataset) including the sustained periodicity of the number of arrivals per hour it exhibits. The search for an adequate model involves selecting a suitable family of stochastic processes and, by application of a parameter estimation approach, identifying a model instance that adequately approximate the dataset. The choice of the target family of stochastic process depends very much on the statistical perspective one wishes to mimic.

For example to adequately reproduce the PDF of the inter-arrival time we may consider the family of Markov renewal processes (MRPs) and derive the parameters of the corresponding continuous phase-type (CPH) distribution. Although MRPs can be used to obtain an arbitrarily precise approximation of the inter arrival time PDF they fall short when it comes to reproduce periodicity as they do not allow to account for existing correlation in the observed inter arrival times, as is indeed the case with the considered dataset (Figure 5.6). In this respect the family of Markov arrival processes (MAPs) constitute an improvement as they do allow one to account for correlation between inter-arrival times hence they allow for generating arrivals that not only match the inter-arrival PDF but that are also correlated. However if the considered dataset exhibits correlation also w.r.t. the number of arrivals per time-frame (as in our case), MAPs may not be a convenient option to mimic both kind of correlation. In fact the ability of MAPs to cope with correlated

data affects the size of the the model (i.e. the number of phases it consists of). In the remainder we show that the size of a MAP capable of reproducing (to some extent) the correlation between inter-arrivals times is already critical, hence if we wanted to reproduce also the correlation between the number of arrivals per hour we would need an even larger MAP for which, however, fitting techniques would not be applicable due to the too large number of states. Therefore in order to obtain a model capable of matching the hourly periodicity of the arrival process, we resort to the class of hidden Markov models (HMMs) and we apply *learning approaches* to derive models' instances from the dataset.

5.2.1 Markovian renewal process approximation

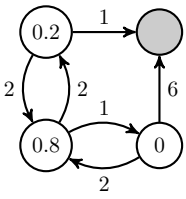
The first family of models we consider for approximating the arrival process is that of Markov renewal process, i.e. a kind of process suitable for modelling independent and identically distributed inter-arrival times. With a MRP the dataset inter-arrival time distribution is approximated by a degree-n phase type (PH) distribution, i.e. the distribution of the time to absorption of a (continuous-time) Markov chain consisting of n transient states (the phases) and one absorbing state [Neu75]. A degree-n CPH distribution is conveniently described by the vector $\alpha \in [0,1]^n$ of initial probabilities of the transient states[3] and the matrix $Q \in \mathbb{R}^{n \times n}$ consisting of the transition intensities among the transient states and the opposite of the sum of the intensities of the outgoing transitions in the diagonal (this allows to determine the transition intensities toward the absorbing state). Based on basic properties of CTMCs the analytical expression of the pdf of the time to absorption $f : \mathbb{R}^+ \to [0,1]$ of a CPH is given by $f(x) = \alpha e^{Qx}(-Q)\mathbb{1}$ where $\mathbb{1}$ is the column vector of 1s, and, consequently, that of its i-th ($i \in \mathbb{N}^+$) moment is given by $m_i = i!\alpha(-Q)^{-i}\mathbb{1}$.

Figure 5.7 illustrates a toy example of a degree-3 CPH whose initial probability vector α and intensities matrix A are given in the left, the corresponding state-transition graph is depicted in the middle (the probability values inside the states indicates the initial probability of the state while real values labelling the arcs represent transition intensities and the gray state is the absorbing one) and the corresponding PDF of the time to reach the absorbing state is shown on the right.

Having fixed the degree n for the target CPH, our goal is to derive α and Q such that the pdf of the associated CPH distribution is a good approximation of the patient inter-arrivals pdf. There are two families of approaches to face this problem, i.e., those based on the maximum likelihood method as opposed to those based on moments matching methods. Maximum likelihood methods are computationally more hungry as they require to consider the entirety of the parameters (the n parameters of α and the n^2 of Q) whereas moments matching approaches only needs to match $2n-1$ moments [BHT08] in order to obtain an estimate of the target $n+n^2$ parameters of α and Q (based on the moments every entry of α and Q is determined directly), hence, in this work we used moment matching methods, as they considerably reducing the computational cost.

[3] we assume in this work that the initial probability of the absorbing state is 0

$$\alpha = (0.2,\ 0.8,\ 0)$$
$$Q = \begin{pmatrix} -3 & 2 & 0 \\ 2 & -3 & 1 \\ 0 & 2 & -8 \end{pmatrix}$$



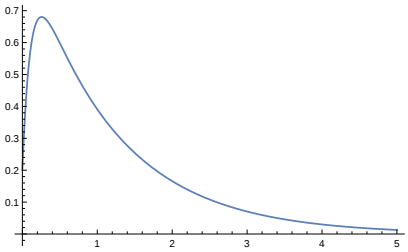


Figure 5.7: A degree-3 CPH: initial probability vector α, intensities matrix Q, state-graph (middle) and corresponding time-to-absorption PDF (right).

The approaches proposed in [HT07, ACH11] construct a degree-n CPH distribution matching $2n-1$ moments if the moments can be realized by a degree-n CPH distribution.

Figure 5.8 shows the resulting initial probability vector α and transition matrix Q for a degree-n CPH (with $n = 1, 2$ and 3) obtained from the Cantu hospital dataset through the BuTools package [BUT], a software tool that implements the CPH moment matching approach.

n	$2n-1$ (moments matched)	estimated α and Q
1	1	$\alpha = (1)$ $Q = (-0.051453)$
2	3	$\alpha = (0.0409229,\ 0.959077)$ $Q = \begin{pmatrix} -0.0213213 & 0.0213213 \\ 0 & -0.0570911 \end{pmatrix}$
3	5	$\alpha = (0.0284672, 0.881506, 0.0900266)$ $Q = \begin{pmatrix} -0.699657 & 0 & 0 \\ 0.0667986 & -0.0667986 & 0 \\ 0 & 0.0260063 & -0.0260063 \end{pmatrix}$

Figure 5.8: Results of moment matching of degree-n CPH processes w.r.t. the patient's arrivals dataset.

From the estimated degree-n CPH (Figure 5.8) we computed (through uniformisation) the PDF of the time-to-absorption for the 3 cases $n = 1, 2, 3$ and compared them with the inter-arrival PDF computed over the dataset (Figure 5.9). We point out that only the degree-3 CPH provides a good approximation of the dataset PDF, whereas the 2-phases CPH is capable of approximating only the front of the PDF while the 1-phase CPH fails to approximate the front too.

It is evident that, even if the inter-arrival time distribution is well approximated, a

Figure 5.9: Front (left) and tail (right) of pdf of CPH distribution approximations of the inter-arrival time distribution based on moment matching.

renewal process is significantly different from the data set under study because it cannot exhibit correlation and periodicity. Nevertheless, as we will show in Section 5.3.2, the renewal process with a proper PH distribution can give good approximation of some performance indices. In the following we propose models in which there is correlation in the inter-arrival time sequence.

5.2.2 Markovian arrival processes

To overcome the limitations of MRPs, i.e. the impossibility that they generate correlated arrivals, we turned to the family of continuous time Markovian arrival processes (MAP) [Neu79]. A MAP is a kind of process in which the emulation of events of interest (arrivals in our case) is governed by a background CTMC, that is, a CTMC whose states are used to model Poisson processes of given arrival intensities, and whose state-transitions allows to switch between different Poisson processes hence to model correlations. As a result a MAP allows for modelling of non-exponential and correlated inter-arrival times and may be seen as the generalization of Poisson point process.

The infinitesimal generator matrix of the underlying CTMC of an n-state MAP will be denoted by the $n \times n$ matrix D. Arrivals can be generated in two ways. First, in each state i of the CTMC a Poisson process with intensity λ_i is active and can give rise to arrivals during a sojourn in the state. Second, when in the CTMC a transition from state i to j occurs an arrival is generated with probability $p_{i,j}$. A convenient and compact notation to describe a MAP is obtained by collecting all intensities in two matrices, $D^{(0)}$ and $D^{(1)}$, in such a way that $D^{(0)}$ contains the transition intensities that do not generate an event and $D^{(1)}$ those that give rise to one, including the intensities of the Poisson processes in the diagonal of $D^{(1)}$. Moreover, the diagonal entries of $D^{(0)}$ are set in such a way that we have $D^{(0)} + D^{(1)} = D$. Accordingly, the entries of $D^{(0)}$ and $D^{(1)}$ are

$$\begin{aligned} \forall i, j, i \neq j: \quad & D^{(0)}_{i,j} = (1 - p_{i,j}) D_{i,j}, \quad D^{(1)}_{i,j} = p_{i,j} D_{i,j} \\ \forall i: \quad & D^{(0)}_{i,i} = - \sum_{\forall j, j \neq i} D_{i,j} - \lambda_i, \quad D^{(1)}_{i,i} = \lambda_i \end{aligned}$$

In most of the literature, the vector of steady state probabilities of the background chain, denoted by $\gamma = (\gamma_1, ..., \gamma_n)$, is used as initial probability vector of the model[4]. In this paper we will do the same. As an example, consider the 2-state MAP described by

$$D^{(0)} = \begin{pmatrix} -3 & 1 \\ 0 & -7 \end{pmatrix} \quad D^{(1)} = \begin{pmatrix} 0 & 2 \\ 2 & 5 \end{pmatrix} \quad D = \begin{pmatrix} -3 & 3 \\ 2 & -2 \end{pmatrix} \tag{5.1}$$

which implies that during a sojourn in state 1 no arrivals are generated while during a sojourn in state 2 a Poisson process with intensity equal to 5 is active. Moreover, a transition from state 1 to state 2 generates an arrival with probability 2/3 while transitions from state 2 to 1 are always associated with an arrival. The steady state probabilities of the background chain are $\gamma = (0.4, 0.6)$.

The well-known Poisson process is a MAP with a single state ($n = 1$). The renewal process used in Section 5.2.1 can be expressed in terms of a MAP by setting $D^{(0)} = Q$, $D^{(1)} = (-Q)\mathbb{1}\alpha$. The Markov modulated Poisson process, in which a Poisson process is active in every state of a background CTMC, is a MAP whose $D^{(1)}$ matrix contains non-zero entries only in its diagonal.

As for CPH distributions, two families of parameter estimation methods have been developed in the literature: the first based on the maximum-likelihood principle ([Ryd96, Buc03, OD09]) and the second based on matching a few statistical parameters of the arrival process. The representation given by $D^{(0)}$ and $D^{(1)}$ contains $2n^2 - n$ parameters (in every row of $D = D^{(0)} + D^{(1)}$ the sum of the entries must be zero) and it is redundant as an n-state MAP is determined by $n^2 + 2n - 1$ parameters ($2n - 1$ moments of the inter-arrival times and n^2 joint moments of consecutive inter-arrival times; see [TH07] for details). Maximum-likelihood based methods suffers from the same drawbacks described before in case of CPH distributions. In this paper we experiment with methods that belong to the second family of approaches.

A 2-state MAP is determined by 3 moments of the inter-arrival times and the lag-1 auto-correlation of the inter-arrival time sequence [BHHT08, TH07]. Our sequence has however such a lag-1 auto-correlation that cannot be realized with only 2 states. In [Hor13] a method was proposed that creates a MAP with any 3 inter-arrival time moments and lag-1 auto-correlation. This method, implemented in the BuTools package, provides the following 6-state MAP:

$$D^{(0)} = \begin{pmatrix} -0.0293 & 0.0293 & 0 & 0 & 0 & 0 \\ 0 & -0.1883 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.0103 & 0.0103 & 0 & 0 \\ 0 & 0 & 0 & -0.0989 & 0.0989 & 0 \\ 0 & 0 & 0 & 0 & -0.0989 & 0.0989 \\ 0 & 0 & 0 & 0 & 0 & -0.0989 \end{pmatrix} \tag{5.2}$$

[4]Notice that γ exists and is independent of the initial state as the background CTMC is, by definition, ergodic.

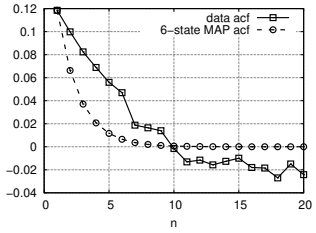


Figure 5.10: Acf of the inter-arrival time sequence of the original data and of the MAP given in (5.2-5.3).

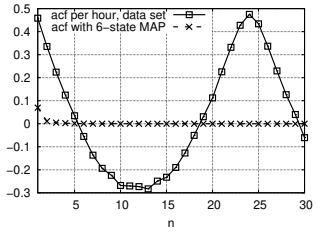


Figure 5.11: Acf of the number of arrivals per hour sequence of the original data and of the MAP given in (5.2-5.3).

$$D^{(1)} = \begin{pmatrix} 0. & 0. & 0. & 0. & 0. & 0. \\ 0.023 & 0.1288 & 0.0002 & 0.0362 & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. \\ 0.0037 & 0.0207 & 0.0004 & 0.074 & 0. & 0. \end{pmatrix} \tag{5.3}$$

with which the steady state probability vector of the background Markov chain is $\gamma = (0.149503, 0.153343, 0.0126432, 0.22817, 0.22817, 0.22817)$.

As opposed to the approach used in Section 5.2.1, in the arrival process generated by a MAP there can be correlation between subsequent inter-arrival times. In Figure 5.10 we depict the acf in the inter-arrival time sequence generated by the above 6-state MAP and that computed on the available dataset. As guaranteed by the applied method for $n = 1$ the auto-correlation is matched exactly but then it fails to follow the auto-correlation of the data. The acf of the sequence counting the number of arrival per hour generated by the same MAP is given instead in Figure 5.11 which is very different from the that of the dataset. In the number of patients per day sequence the auto-correlation of the 6-state MAP with $n = 1$ is 0.00469433, i.e., it is negligible, as opposed to that in the data where it is 0.22.

As seen above, 6 states are necessary to match three moments and just the lag-1 autocorrelation of the inter-arrival times. This indicates that a MAP with large number of states is necessary to capture the peculiar statistical features of the patient arrival process. General purpose MAP fitting techniques are not applicable however with such large number of states.

5.2.3 Hidden Markov processes

A Hidden Markov model (HMM) [Rab90] can be thought of as a generalisation of a DTMC for which an external observer cannot directly see the states but only observe some *output* whose probability to be emitted depends on the state. In practice an HMM is characterised by: *i)* a set of states $\mathcal{S} = \{s_1, \ldots, s_n\}$; *ii)* a set of possible observations $\mathcal{O} = \{o_1, \ldots, o_m\}$;

iii) a $n \times n$ state-transition probability matrix $A = \{a_{ij}\}$ with a_{ij} being the probability of transitioning from state s_i to s_j; *iv)* a $n \times m$ state-observation probability matrix $B = \{b_{ij}\}$ where b_{ij} is the probability of observing o_j when the chain enters state s_i; *v)* an initial distribution over the set of states S denoted by α. Accordingly, an HMM is completely determined by the triple (α, A, B).

There exist three classical problems for HMM, all of which requiring a sequence of observations $O = (o_1, \dots o_k)$. The first one is the *evaluation problem*: given an HMM by (α, A, B) and a sequence O, calculate the probability that the HMM produces O. The second, called *decoding problem*, consists of determining the most probable state sequence given a HMM and a sequence O. The third one, the *learning problem*, has only O as input and is about finding such triple (α, A, B) with which observing O is most probable.

In this paper we focus on the third type of problem and in particular we develop and study two HMMs aimed at reproducing the statistical characteristics of the patients' arrival dataset (described in Section 5.1). We start off with a rather coarse-grained (only 3-state) but general HMM, which reveals to be capable of reproducing only marginally the auto-correlation characteristics of the patients arrivals data. Then we propose a HMM with particular underlying DTMC that shows good agreement with the dataset from a statistical point of view.

A 3 states HMM

Here we consider a 3 state HMM in which every time slot corresponds to one hour and the possible observations are $\mathcal{O} = \{0, 1, ..., 14\}$ interpreted as the number of patient arrivals per hour[5]. This means that the sequence generated by the HMM has to be post-processed if we need to specify the exact arrival instance for each patient. This post-processing will consist of distributing the arrivals in uniform manner inside the hour.

There are 2 free parameters in π, $3 \times 2 = 6$ in A and $3 \times 14 = 42$ in B (because π must be normalized and also the rows both in A and B must be normalized). We applied the Baum-Welch algorithm [BPSW70] to determine the optimal parameters starting from several initial parameter sets chosen randomly. With this relatively small model, the final optimal parameters obtained by the Baum-Welch algorithm are independent of the initial values (apart from permutations of the states). The obtained HMM is with $\pi = (0, 1, 0)$ and

$$A = \begin{pmatrix} 0.865 & 0.135 & 0 \\ 0 & 0.835 & 0.165 \\ 0.134 & 0 & 0.866 \end{pmatrix} \quad B = \begin{pmatrix} 0.0007 & 0.0227 & 0.0807 & 0.1569 & 0.1756 & 0.1731 \\ 0.024 & 0.1103 & 0.2237 & 0.2249 & 0.19 & 0.1323 \\ 0.3533 & 0.3752 & 0.1989 & 0.0583 & 0.0116 & 0.0025 \\ 0.1652 & 0.0991 & 0.0625 & 0.0292 & 0.0177 & 0.009 & 0.0058 & 0.0006 & 0.0006 \\ 0.0544 & 0.0303 & 0.0052 & 0.0025 & 0 & 0 & 0.0004 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{5.4}$$

[5]More than 14 patients per hour are very rare in our dataset.

The mean number of arrivals per hour with the above HMM is 3.0862 and its variance is 5.67669 while in the data trace the mean and the variance are 3.08676 and 5.69061, respectively.

In Figure 5.12 (left) we depict the autocorrelation of the number of arrivals per hour. As one can expect, the 3-state HMM is not able to reproduce the sustained oscillation present in the data shown in Figure 5.3 (autocorrelation is negligible for $n \geq 10$) but does much better than the 6-state MAP given in (5.2-5.3) (see Figure 5.11).

A 24 states HMM

In order to have a model that is able to exhibit oscillation in the autocorrelation function of the number of arrivals per hour, we define a 24-state HMM in which each state corresponds to an hour of the day and the transition probabilities are such that the process deterministically cycles through the 24 states. Accordingly, A is 24×24 and its entry in position (i, j) is 1 in $j = (i + 1) \pmod{24}$ and it is 0 otherwise. The initial probability vector is set to $\pi = (1, 0, ..., 0)$. As before, the possible observations are $\mathcal{O} = \{0, 1, ..., 14\}$.

The Baum-Welch algorithm is such that parameters set to 0 initially remain 0. Consequently, the algorithm does not change the matrix A and the vector π and has an effect only on the entries of B. The number of parameters is larger, it is $24 \times 14 = 336$, but thanks to the deterministic behavior of the underlying DTMC the Baum-Welch algorithm determines the parameters in a single iteration. With the resulting HMM the mean number of arrivals per hour is 3.08619 while the variance is 5.67465. The hourly autocorrelation is shown in Figure 5.12 (center). This model provides very similar hourly autocorrelation to that of the data set.

We experimented also with a 24-state HMM letting the Baum-Welch algorithm to change any entry of the matrix A (i.e., the initial entries of A are random strictly greater than 0 and strictly smaller than 1). This way the number of free parameters is $24 \times 23 + 24 \times 14 = 888$. With this large number of parameters the Baum-Welch algorithm performs 4798 iterations and requires about 3 minutes of computation time on a standard portable computer. The resulting HMM has mean and variance equal to 3.08539 and 5.67468, respectively. The matrix A has a similar structure to the one before but the probabilities of going to the next state is not 1 but a value between 0.6 and 0.99. The resulting autocorrelation structure is depicted in 5.12 (right). With the non-deterministic underlying Markov chain the autocorrelation cannot exhibit sustained oscillation and the autocorrelation vanishes after about 150 hours (i.e., 6-7 days).

5.3 Empirical validation of derived models

In the previous section we analysed the adequacy of each kind of *arrival* model derived from the dataset in "isolation", that is: we used each model to generate arrivals observations and compared relevant statistics (e.g. pmf of inter-arrivals, auto-correlations of number of

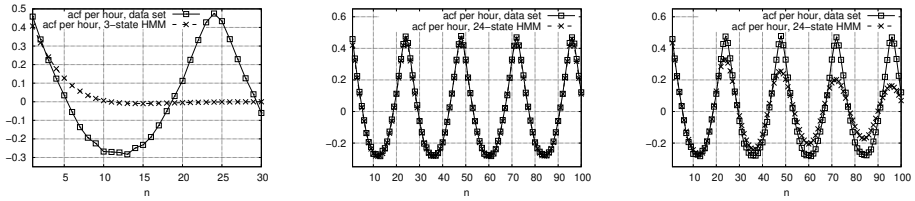


Figure 5.12: Comparing the autocorrelation function (acf) of the dataset with that of the 3-state HMM (left), the 24-state HMM with deterministic (center) and non-deterministic (right) underlying Markov chain.

arrivals per hour) computed on model's generated arrivals against that computed on the original dataset. To extend the validation of the derived models here we consider a simple ED *service* model (Figure 5.14) representing the services a patient may go through during his permanence at the ED and we "stimulate" it with arrivals generated on one hand by each of the *arrival* models derived from the dataset as well as with the arrivals of the actual dataset. We then compare meaningful *service* related statistics computed w.r.t. the combined arrival/service models against that computed by stimulating the service model with the original (arrivals) dataset.

For convenience we resort to the HASL model checking platform for the validation of the combined arrival/service models, hence we rely on stochastic Petri net[6] encoding of both the arrival and the service process. The Petri net encoding of the Markovian arrival models derived from the dataset is pretty straightforward however the complexity of the resulting Petri net increases with that of the considered model. For the Markov renewal, the MAP and the 3-states HMM the resulting Petri net is pretty simple however the complex structure of the 24-states HMM arrival model resulted in a Petri net consisting of 63 places and 398 transitions (15×24 of which are immediate, 14 are timed with uniform distribution over $[0, 1]$ and 24 are timed deterministic with delay equal to 1).Therefore, below we outline a formal Petri net encoding of periodic HMM models based on which we implemented an automatic generator of the corresponding NMGSPN models in terms of an input file for the COSMOS model checker. Similarly from the actual dataset consisting of N arrival events we also generate automatically its the Petri net encoding, which trivially corresponds with a Petri net consisting of N concurrently enabled deterministically timed transition whose deterministic delay is set to that of the corresponding arrival in the dataset.

5.3.1 Petri Net encoding of HMM arrival models

We introduce a Petri net formal encoding of an HMM that representing an arrival process whose arrivals are supposed to occur per discrete time unit. Such encoding allows us to

[6]specifically the NMGSPN outlined in Chapter 2.1.2.

automatically translate an arrival HMM model into a corresponding NMGSPN that can then be composed with a Petri net model that represents the services required by arrived entities.

Definition 5.2 (Petri net encoding of an n-states HMM)**.** Let $\mathcal{H} = (\pi, A, B)$ be an n-state m-observations HMM whose observations per discrete time unit are assumed to follow a uniform distribution and where π is the initial state distribution vector of $\mathcal{H}$, A is the $n \times n$ state-transition probability matrix of the background chain and B is the $n \times m$ state-observation probability matrix. We define $\mathcal{N}_{\mathcal{H}} = (P, T, I, O, M_0)$ the NMGSPN encoding of $\mathcal{H}$ as follows:

- $P = \{P_0, P_1, ..., P_n, O_1, ..., O_n, W, Arrs\}$ is the set of places whose elements are:
  - P_0 is the initial place used to start the model according to the initial probabilities π
  - P_i $(1 \leq i \leq n)$ corresponds with the i-th state of the background chain of $\mathcal{H}$
  - O_i $(1 \leq i \leq n)$ models the triggering of observations that can occur while $\mathcal{H}$ is in the i-th state.
  - W is used to collect the observations (i.e. number of arrivals) triggered while $\mathcal{H}$ is in a given state and that has to occur in the next discrete time unit
  - $Arrs$ used to gather that actual observations occurred in the last discrete time unit
- $T = T_{\mathcal{I},1} \cup T_{\mathcal{D}} \cup T_{\mathcal{I},2} \cup \{t_{dist}\}$
  - $T_{\mathcal{I},1} = \{t_{0,i} : 1 \leq i \leq n, \pi_i > 0\}$, set of immediate transitions used to probabilistically start the process according to π, i.e. the weight of $t_{0,i}$ is given by $\pi(i)$.
  - $T_{\mathcal{D}} = \{t_{i,j} : 1 \leq i, j \leq n, A_{i,j} > 0\}$ set of timed transition whose transitions are associated with fixed delay of one time unit and where $t_{i,j}$, if present, represents the probabilistic transition from state i to j as per the matrix A, hence $t_{i,j}$ is assigned with weight $A_{i,j}$.
  - $T_{\mathcal{I},2} = \{t^{(o)}_{i,j} : 1 \leq i \leq n, 1 \leq j \leq m, B_{i,j} > 0\}$ set of immediate transitions in which $t^{(o)}_{i,j}$ models the probabilistic occurence of j-th observation while in the i-th state of $\mathcal{H}$ as per matrix B, hence $t^{(o)}_{i,j}$ is assigned weight $B_{i,j}$.
  - t_{dist} is a timed transition associated with a Uniform distribution over time interval $[0, 1]$ used to model the actual delay of occurrence of each observation in the next time unit. Transition t_{dist} is associated with infinite server policy.

- $I : T \to Bag(P)$ is the set of input arcs of transitions in T

$$I(t) = \begin{cases} \{P_0\} & \textbf{if } t = t_{0,i}, i \in \{1, n\} \wedge \pi(i) > 0 \\ \{P_i\}, & \textbf{if } t = t_{i,j}, (i,j) \in \{1,n\} \times \{1,m\} \ \wedge A_{i,j} > 0 \\ \{O_i\}, & \textbf{if } t = t_{i,j}^{(o)}, (i,j) \in \{1,n\} \times \{1,m\} \wedge B_{i,j} > 0 \end{cases}$$

- $O : T \to Bag(P)$ is the set of output arcs of transitions in T

$$O(t) = \begin{cases} \{P_i\} + \{O_i\} & \textbf{if } t = t_{0,i}, i \in \{1, n\} \wedge \pi(i) > 0 \\ \{P_j\} + \{O_j\}, & \textbf{if } t = t_{i,j}, (i,j) \in \{1,n\} \times \{1,m\} \wedge A_{i,j} > 0 \\ (j-1)\{W\}, & \textbf{if } t = t_{i,j}^{(o)}, (i,j) \in \{1,n\} \times \{1,m\} \wedge B_{i,j} > 0 \end{cases}$$

- $M_0 = \{P_0\} \subset Bag(P)$ is the initial marking

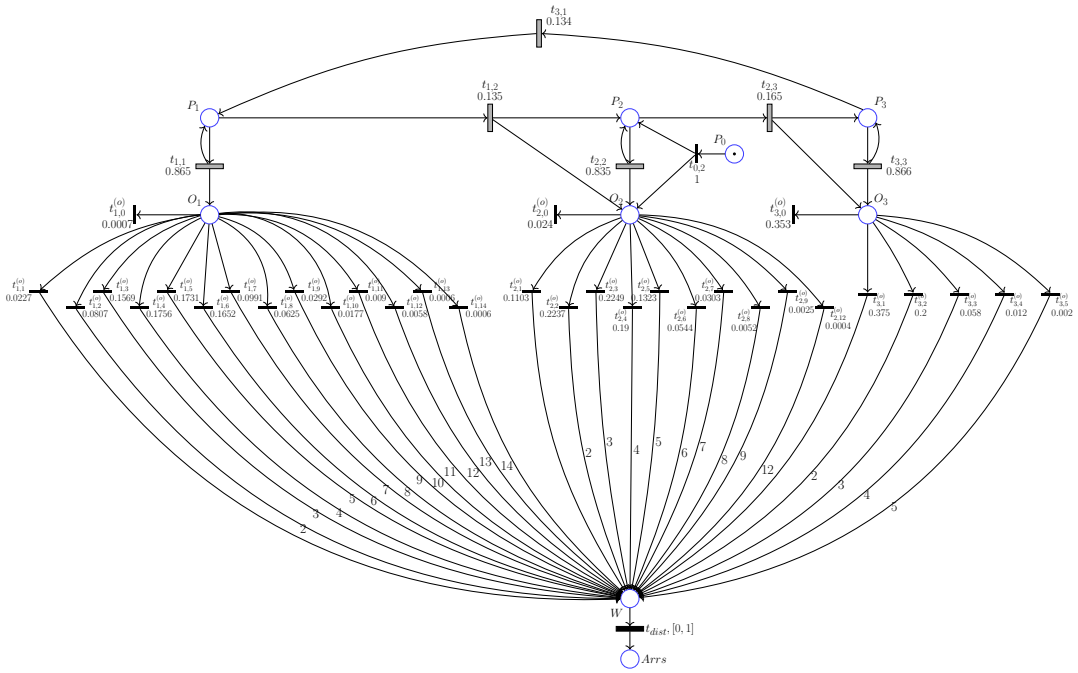


Figure 5.13: Petri net encoding of the 3-state HMM model given in (5.4).

Example 7 (Encoding of 3-states HMM derived from ED dataset)**.** Figure 5.13 shows the NMGSPN encoding of the 3-state HMM given in (5.4) which we derived from the ED dataset. It consists of 8 places and 41 transitions, 7 of which are timed the rest immediate.

Timed deterministic transitions of $T_{\mathcal{D}}$ are depicted as filled-in gray thick rectangles, the single timed uniform transition t_{dist} is drawn as a black rectangle and labeled with its firing interval while immediate transitions of $T_{\mathcal{I},1} \cup T_{\mathcal{I},2}$ are depicted as thin black rectangles. The label of each timed deterministic transitions as well as that of the immediate transitions contains the "weight" of the transition (i.e. a positive real-value used to determine the probability with which the transition fires when concurrently enabled with other transitions) which corresponds with entry $A_{i,j}$, for timed deterministic transitions $t_{i,j} \in T_{\mathcal{D}}$, with entry $B_{i,j}$, for the immediate "output" transitions $t^{(o)}_{i,j} \in T_{\mathcal{I},2}$ and with entry π_i for initial state transitions $t_{0,i} \in T_{\mathcal{I},1}$, where A, B are the matrices in (5.4) and $\pi = (0, 1, 0)$.

In conformance with the initial distribution of the 3-states HMM $\pi = (0, 1, 0)$, the initial marking M_0 has a single token in place P_0, and accordingly transition $t_{0,2}$ weight is set to 1. On firing of $t_{0,2}$ a token is placed in P_2 representing the activation of the (periodic) background chain by enabling the competing timed deterministic transitions $t_{2,3}$ and $t_{2,2}$. Notice that the timed timed transitions $t_{i,j}$ coupled with the corresponding $t_{i,i}$ ($i \in \{1, \ldots, n\}, j = (i + 1) mod(n)$) model the competition between the occurrence of an observation-less transition to the "next" state of the background chain and an observation filled permanence in the i-th state of the chain in the next 1-time-unit, with the competition being solved probabilistically through the weights of $t_{i,j}$ and $t_{i,i}$. Whenever the competition between $t_{i,j}$ and $t_{i,i}$ is won by $t_{i,i}$ a token is added to O_i activating a probabilistic choice between the immediate transitions $t^{(o)}_{i,k}$ ($k \in \{1, \ldots, m\}$) which model different possible number of arrivals to occur in the next time unit, with $t^{(o)}_{i,0}$ representing 0 arrivals to be observed in the next time unit. Notice that the number of existing immediate transitions $t^{(o)}_{i,k}$ activated by a token in O_i depends on the number of non-null entries in the i-th row of matrix B. For example for the 3-states HMM we know that the maximum number of arrivals per hours is $m = 14$ therefore at most 15 immediate transitions $t^{(o)}_{i,k}$ (i.e. $k \in \{0, \ldots 14\}$) can be activated by a token in O_i. However in the NMGSPN model in Figure 5.13 only O_1 is equipped with all 15 output transitions $t^{(o)}_{1,k}$ while for O_2 we only have 11 output transitions $t^{(o)}_{2,k}$ (e.g $t^{(o)}_{i,10}$ and $t^{(o)}_{i,11}$ are missing as the corresponding entries of B are 0) and similarly for O_3 we only have 6 transitions $t^{(o)}_{3,k}$.

Notice that the number of occurrences of arrivals to be generated in the next time unit is modelled by setting to k the multiplicity of $t^{(o)}_{i,k}$ output arc. Finally the actually occurred arrivals are represented by the tokens collected in the sink place *Arrs*.

Petri net model of ED services. For the sake of simplicity we assumed the ED patient flow consisting of a simple pipeline of 3 services: *triage*, *visit* and *discharge* for each of which we considered exponentially distributed service time with the following settings [7]: *triage*$\sim Exp(12)$, the *visit*$\sim Exp(6)$ and *discharge*$\sim Exp(60)$ (i.e. visit is assumed to be the slowest while discharge the fastest service). Figure 5.14 shows

[7]The rates have been devised form statistical analysis of the Cantù dataset.

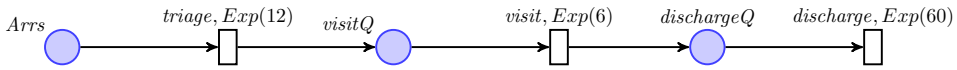


Figure 5.14: A simple NMGSPN model of ED services consisting of 3 pipelined services (to be composed by overlapping of place *Arrs* with an NMGSPN model of the patients arrivals).

the NMGSPN encoding of the ED service model: notice that place *Arrs* is shared with the NMGSPN models of the patient arrivals (representing the composition of the two models). To validate each arrival model we compared their performance with that resulting by the simulation of the ED service model (Figure 5.14) with the actual arrivals of the dataset (for this we generated an NMGSPN encoding of the dataset arrivals).

5.3.2 KPI assessment

In order to compare the performances of the different patient arrival models coupled with the ED services we considered the following key performance indicator (KPI):

$$\phi_1 \equiv \textit{pmf of the number of patients in the ED during a periodic time-window.}$$

We formally specified ϕ_1 by means of the Hybrid Automata Specification Language (HASL) (see Chapter 2) and assessed it through the statistical model checking platform Cosmos [BDD+11a, COS].

Figure 5.15 depicts the HASL specification for ϕ_1. The LHA consists of 4 states (l_0,l_{on},l_{off},l_{end}), 1 clock variable (t), 2 stopwatches (t_a, t_p), $M+1$ real valued variables x_i ($0 \leq i \leq M$), whose final value is the probability that i patients have been observed in the ED during the periodic time windows), plus a number of auxiliary variables and parameters.

The LHA is designed so that measuring, along a trajectory, is periodically switched ON/OFF (parameter TP being the ON period duration) and stops as soon as the NP-th period has occurred. In the initial state l_0 the occurrence of any transition is ignored ($l_0 \xrightarrow{\text{ALL},t<initT,\emptyset} l_0$) for an initial transient of duration $initT$ ($initT$ being a parameter of the LHA) at the end of which the LHA moves to l_{on} ($l_0 \xrightarrow{\sharp,t\geq initT,\emptyset} l_{on}$). In l_{on} the LHA reacts to the occurrence of patients arrival as well as patients discharge events. When a patient arrives (resp. is discharged) while i patients are in the ED ($l_{on} \xrightarrow[\{n+=1,x_i+=t_a,t_a=0\}]{\{tarr\},n=i\wedge tp<TP\wedge tp<TP\wedge np<NP} l_{on}$, resp. $l_{on} \xrightarrow[\{n-=1,x_i+=t_a,t_a=0\}]{\{discharge\},n=i\wedge np<NP} l_{on}$) the patients counter n is incremented (resp. decremented), the duration of the last time interval on which the ED contained i patients is added up to x_i, and the arrival stopwatch t_a is reset. Any event different from a patient's arrival/discharge is ignored in l_{on}

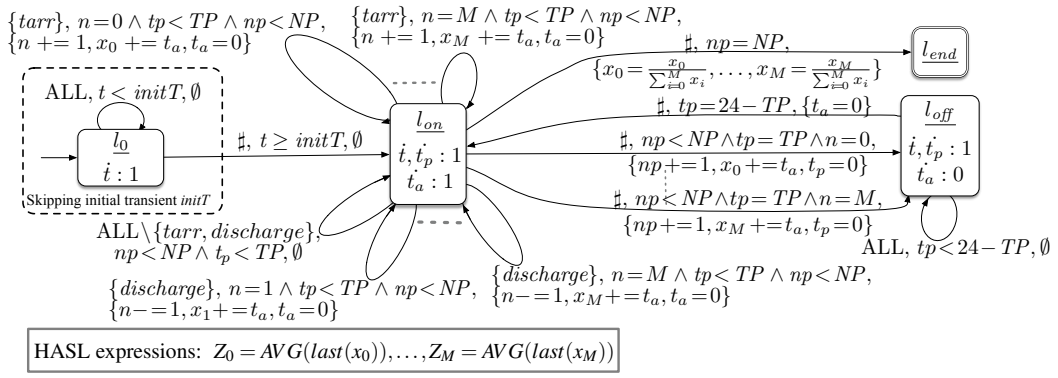


Figure 5.15: Automaton for measuring the pmf of the number of patients in the ED.

($l_{on} \xrightarrow[tp<TP \wedge np<NP, t_p<TP, \emptyset]{\text{ALL}\setminus\{tarr, discharge\}} l_{on}$). As soon as the ON-period expires ($t_p = TP$) the LHA moves from l_{on} to l_{off} where it suspends registering the duration of different number of patients in the ED. In that respect observe that if the end of an ON-period corresponds with i patients being in the ED, the variable x_i (which accumulates the durations of i patients being in the ED) is added up with the time elapsed since the arrival of the last patient ($l_{on} \xrightarrow[np+=1, x_i+=t_a, t_p=0\}]{\sharp, np<NP \wedge tp=TP \wedge n=i} l_{off}$), hence the end of the ON-period is made corresponding with the end of the duration of having i patients in the ED. In l_{off} the automata ignores any event ($l_{off} \xrightarrow{\text{ALL}, tp<24-TP, \emptyset} l_{off}$) while it switches back to l_{on} as soon as the reciprocal (w.r.t. 24 hours) of the ON-period duration elapsed ($l_{off} \xrightarrow{\sharp, tp=24-TP, \{t_a=0\}} l_{on}$) and in so doing the timer t_a (which stores the duration of the occupation at i patients since the last arrival/departure) is reset so that it can correctly be used in the freshly started ON-period. Finally, the LHA stops monitoring as soon as NP ON-periods have been observed along the monitored trajectory: at that moment each x_i is normalised w.r.t. the sum of all x_i, hence on ending the monitoring of trajectory x_i is assigned with the probability that the i patients have been observed in the ED during NP observed periods ($l_{on} \xrightarrow{\sharp, np=NP, \{x_0=\frac{x_0}{\sum_{i=0}^{M} x_i}, \ldots, x_M=\frac{x_M}{\sum_{i=0}^{M} x_i}\}} l_{end}$). Observe that such an LHA can also be used for "non-periodic" measures: it suffices to set $TP = 0$. The HASL specification for ϕ_1 is completed by the list of HASL expressions $AVG(last(x_i))$ which indicate that a confidence interval for average of the last value that x_i has at the end of an accepted trajectory is computed by COSMOS.

Figure 5.16 depicts plots computed with the COSMOS tool and resulting from assessing specification ϕ_1 (i.e., the pmf of the number of patients in the ED) against the NMGSPN models of the patient arrivals coupled with the 3-services model of the ED (Figure 5.14).

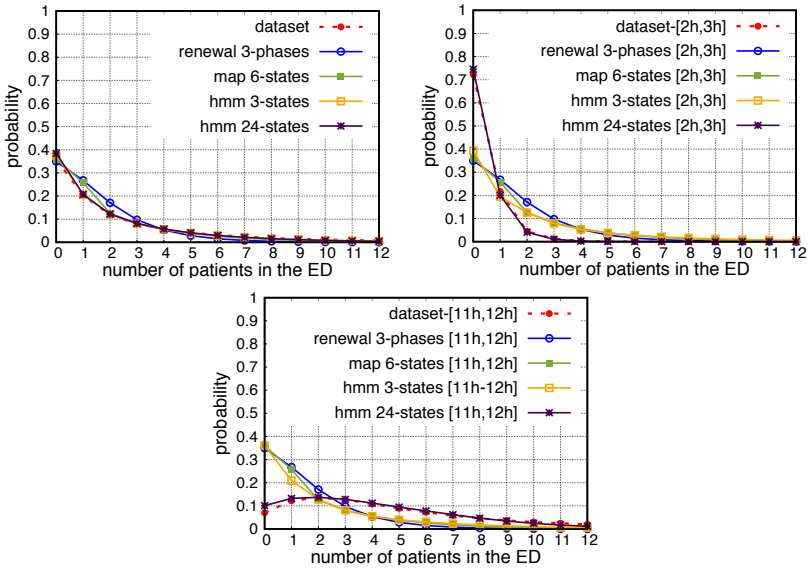


Figure 5.16: Pmf of the num. of patients in the ED computed through 3-phases PH renewal model, 6-state MAP, 3-state HMM and 24-state HMM versus dataset model over the whole day (top left), during a low-arrival hour (top right) and high-arrivals hour (bottom).

Plotted results refer to a 1-year (365 days) observation window and have been computed as 99.99% confidence interval of 10^{-3} width. The top-left plot refers to the pmf measured without taking into account any specific time-window over the day (continuous measuring over 24h for 365 days), while the top-right and bottom plots refer to periodic measuring over a 1-hour period, at low arrival intensity (from 2am to 3am, top-right plot) and at high arrival intensity (from 11am to 12pm, bottom plot), respectively. The obtained results witness the clear advantage of the HMM24 model over the rest: if when no specific hour of the day is considered the pmf of the 3-phases renewal model and of the 6-states MAP provide an acceptable approximation of the pmf computed w.r.t. to dataset (red dashed plot) and the pmf of the HMM3 and HMM24 are essentially indistinguishable, this is no longer the case when the pmf is computed on a specific time window as shown in the top-right and bottom plots. Only the pmf of the HMM24 matches that of the dataset during a low-arrivals [2am,3am] window and a high-arrival [11am,12pm] window: the pmf of the renewal and map models instead are essentially identical to those measured continuously while that for the HMM3 exhibits a slight tendency towards matching the pmf of the dataset at low-arrivals but completely fails to do so when arrivals become intense.

5.4 Perspectives

The contribution presented in this chapter tackles the problem of deriving stochastic models capable of reproducing the periodic character of observations of a real-life phenomenon exhibiting periodicity. Based on a specific dataset that registered observations of patients arrivals to the emergency department of an hospital in northern Italy we derived different instances of Markovian models which we proved able, to different extent, to generate arrivals events reproducing relevant statistical indicators assessed on the dataset itself. More specifically we showed that, amongst different families of Markov models, the class of doubly stochastic processes such as, e.g. Hidden Markov models (HMMs) is worth considering as it has the potential to accurately reproduce the periodic character of the dataset, given it is "properly" set. In fact, generally speaking, classical hidden Markov models fail to adequately capture periodic behaviours because, even if a large number of states is used, after an initial transient phase where the process might be able to mimic the periodicity of the observed phenomenon, as the behavior of the Markov chain, at steady state, "smooths out" and the correlation of the probability of a given state and the elapsed time disappears. To face this problem we forced the background Markov chain of an HMM to be periodic, i.e., any return to any state occurs in multiples of a given number of time steps. Such chains are capable to act as a clock, and hence maintain the periodic pattern of the phenomenon we aim to study, as we demonstrated for the patients arrival dataset. Although promising, the approach outlined in this chapter is somehow limited as it only considers arrival of discrete quantities (arrivals of patients to an ED, or cars to a traffic light), although many real life systems are rather concerned with continuous quantities observations (e.g. energy produced by a photovoltaic system or the amount of yogurt produced in a dairy industry). Therefore it'd be worth to consider how to generalise the proposed approach so that it can cope with other family of periodic phenomena. In practice that implies studying what are the implications of considering different kind of arrival processes while sojourning in the states of the periodic background chain of an HMM. A number of different situations seem worth considering in this respect, including using of deterministic arrival intensity as opposed to random arrival intensity and also using of continuous and/or discrete phase-type distributions to models observations of continuous or discrete observations.

In particular a number of open questions need to be tackled related to parameter estimation algorithms for HMMs with periodic background chain, including: 1) would it be possible to adapt/improve expectation-maximization parameter estimation approaches to HMMs whose background chain is periodic ? (experience suggests that special background structure, like a periodic chain, can lead to more efficient calculation for what concerns the expectation step); 2) would it possible to adapt/improve moment fitting approaches to HMM when phase-type distributions are used to generate observations while sojourning in the states of the periodic background chain ? Those research questions have been integrated within a PhD project proposal that Andras Horváth and myself have written and for which we are currently looking for suitable PhD candidate.

Chapter 6

Other contributions

In this chapter we provide the reader with a brief overview of other contributions not included in this manuscript.

Compositional approaches for stochastic models. Together with Andras Horváth from University of Turin (Italie) we worked on compositional approaches for assessing probabilistic properties of structured Markov chain models. This entails looking at how to exploit the compositional structure of a model in order to alleviate the memory needs of techniques for assessing the model's properties. In a preliminary work [BH08] we focused on the *Boucherie* product processes, a specific class of *product form* continuous time Markov chains and showed that the compositional constraints that lead to the product form result for such class, can be exploited, to a limited extent, in the model-checking problem as well, leading to a decomposed semantics for a fragment of the Continuous Stochastic Logic. As a follow up we looked at a more general class of structured Markov chains those whose state-space can be partitioned in an arbitrary number of disjoint subsets, called macro-states, where macro-states represent the concurrent execution of independent jobs. These kind of chains typical arise from commonly used formalisms such as stochastic Petri nets or stochastic process algebras where the stochastic activities are modelled through phase type distributions. We showed that the computation of the probability that this kind of process passes through a given sequence of macro-states, i.e. the probability of a path consisting of macro-states, can be calculated in a memory efficient manner by means of Laplace transform techniques [BH09a, BH09b].

Transient analysis of stochastic timed automata networks. Together with Nathalie Bertrand of INRIA Bretagne/Atlantique (France), Enrico Vicario and Marco Paolieri from University of Florence (Italie) and Andras Horváth from University of Turin (Italie) we introduced a novel formalism for modelling of concurrent stochastic systems, namely that of interacting stochastic timed automata (STA) and equipped it with an operational

semantics that allow for transient-state analysis [BBH+13]. We introduced STA as an extension of timed automata in which both delays and actions are chosen at random through probability distributions. By imposing specific rules on the transition firing, i.e. from the current state first we randomly select the next action to occur and then we sample the sojourn time (i.e. the delay of occurrence of the selected transition) we ensure that the underlying semantics can be described by means of the stochastic state classes method [BPSV05], that is, a method formerly introduced in the context of stochastic timed Petri nets and that allows for evaluation of steady-state, transient-state and quantitative model checking queries of stochastic processes with regeneration points.

Improved Bayesian parameter estimation. Chapter 3 reports about merging of Bayesian parameter estimation with statistical model checking yielding a novel Approximate Bayesian Computation (ABC) algorithm for parametric verification of temporal logic specifications. That work actually originated from a previous project in which in collaboration with Konstantinos Koutroumpas, Paul-Henry Cournède and Irene Votsi we developed a novel variant of the ABC sequential Monte Carlo (ABC-SMC) scheme, a popular algorithm for data-based parameter estimation of a mathematical (deterministic or stochastic) model. Given a (set of) observation(s) y_{obs} of a given parametric system $\mathcal{M}_\theta$ the goal of ABC-SMC approaches is to estimate the parameters θ that yields a good approximation of the observations. ABC-SMC schemes operate iteratively by progressively increasing the precision of the estimate and employ Kernel distributions to propagate (i.e. "move' around") the parameters within the parameter space. The propagation phase of ABC-SMC turns out to be crucial to the performance of the algorithm, and in this respect the chosen sequence of Kernel distributions can be fundamental. In our contribution, which has been published on the Bioinformatics journal [KBVC16], we introduced a novel version of the ABC-SMC scheme which employs a specific family of stochastic processes, namely the Dirichlet process mixture, as Kernel distributions to move the parameters during the parameters space exploration. We showed that this results in a reduced runtime (i.e. a faster convergence) of ABC-SMC scheme particularly when the posterior distribution of the parameters to estimate is multi-modal. We validated the proposed approach w.r.t. a complex biological network model, that of the Wnt signalling pathway.

Temporal logic based reverse-engineering of genetic regulatory networks. As part of the PhD thesis work of Emmanuelle Gallet, which I co-supervised with Pascale Le Gall, we developed a method for reverse-engineering of parametric, discrete-state, genetic regulatory network (GRN) models. Generally speaking GRN models are sensed to capture the possible evolutions of a population of interdependent proteins whose expression is regulated by *activation/inhibitation* of the corresponding genes. In their discrete abstraction a GRN is modelled by a *transition system* (also called a *dynamics*) whose states encodes the proteins' *counts* (or *levels*) and whose transitions represent the effect that

concurrent regulation dependencies have on the proteins' counts. In this respect the René Thomas formalism constitutes an underspecified (parametric) form of GRN model, one in which *level dependent* genes' inter-dependencies are represented through an *interaction graph* without specifying what is the combined effect that (potentially concurrent) regulators have on the overall state of the network. Such an effect depends on the so-called parameters of the Thomas' GRN, which, once instantiated, allow to obtain a corresponding *dynamics* (i.e. transition system). A relevant problem considered in the literature is that of the identification of GRNs' parameters instances yielding a transition system that fulfil specific behaviours expressed by formulae of the LTL logic. Such a problem suffers from combinatorial explosion as the number of parameters is exponential in the number of genes (and levels) of the network. Solutions have been proposed (e.g. Bernot *et al.* [BCRG04]) that exhaustively search the parameter space and that are implemented by the SMBionet tool [KCAB09], however they are affected by scalability issues. Within Emmanuelle Gallet's PhD thesis we proposed an alternative approach in which we employ *symbolic execution* techniques to solve a parametric LTL model checking problem, the one resulting by considering the product of a parametric (Thomas) GRN with the Büchi automata corresponding to an LTL formula. By so doing we avoid the costly parameter space exhaustive search and obtain a tool that scale up remarkably better than SMBionet. Results of this research line have lead to the following publications [GMGB14, GMGB15], other than to realisation of a prototype software tool named SPuTNIk.

Framework for formal modelling of manufacturing systems. The design of modern industrial production systems is strongly affected by product quality and delivery reliability requirements: the ability to guarantee that products are issued within given time deadlines and that they match given quality standards are essential factors. As part of a research project in a recent collaboration with Andras Horváth (University of Turin) we developed a probabilistic model checking framework for formal modelling and performance analysis of so-called *synchronous production lines* that is, production systems consisting of a multi-stage working line where a workpiece is progressively transformed into the end product through a sequence of processing stages performed by dedicated pipelined workstations (i.e. a workpiece outputted by one workstation is fed in to the next downstream station). Workstations are assumed to be unreliable (can break down hence requiring repairing) and are separated by finite-size buffers (where pieces are temporally stored throughout the intermediate phases of manufacturing) which may induce the blocking of a workstation (e.g if the downstream buffer is full). Furthermore the overall dynamics of such systems is assumed to be strongly synchronous: the state of machines (working or broken) as well as that of the buffers evolve synchronously according to the elapsing of discrete units of time.

It has been shown [CT12, CAH14, AHC14, CHA15, ACHG16] that this kind of production systems are suitable to be modelled through discrete-time Markov chains (DTMCs) and that relevant key performance indicators (KPIs), such as the *lead time distribution* (i.e. the probability distribution of the time taken by the system to output one product),

can be assessed by application of classical transient-analysis of Markov chain models. The downside of those kind of approaches is that they are based on an explicit matrix representation of the DTMC that represents the considered production line, hence they provide little support from a modeller point of view. Furthermore the type of KPIs that one can take into account with this approaches is somehow limited.

In order to tackle these limitations we introduced (two versions of) a model-checking based framework which allows one to automatically generate a DTMC model, through an adequate modelling formalism, for a production line of arbitrary size while, at the same time, a number of relevant KPIs expressed in terms of some temporal logic formalism are also generated. The generated model and KPIs can then be promptly used to run model checking experiments relying on the model checking platform targeted by the framework hence proving an effective means for customised performance analysis of production lines. A first version of such framework [BH21a] is aimed at the PRISM model checking platform and relies on the Continuous Stochastic Logic for expressing relevant KPIs. With this we provide the user with a generator of PRISM models as well as of relevant KPIs expressed as CSL formulae and we further take advantage of model checking to validate the generated models through the verification of a number of *sanity checks* properties. Although effective this first version of the production line modelling framework is undermined by the limited expressiveness of the CSL property language which restrain the KPIs that one can take into account. To overcome this issue we developed a second version of such framework [BH21b] which targets the COSMOS model checking platform hence taking advantage of the much more expressive HASL [BBD+15b] property language. Since COSMOS models are expressed through a stochastic extension of the Petri nets formalism the model generator of this version of the framework generates stochastic Petri nets encoding of production lines of arbitrary size as well as a number of sophisticated KPIs encoded as HASL formulae.

Formal analysis of wireless network protocols. The exchange of information between computing nodes over a wireless network is regulated through Medium Access Control (MAC) algorithms implementing so-called carrier sense multiple access collision avoidance (CSMA-CA) schemes. The basic idea is to employ probabilistic *backoff* procedures in order to reduce collisions between concurrently transmitting nodes, therefore letting a transmitting node *hold on*, for a probabilistically chosen duration, if the shared (wireless) medium is already occupied. This backoff mechanism is at the basis of popular IEEE wireless network standards, ranging from the basic 802.11 standard that addresses wireless local area networks (WLAN), to its *prioritised* extensions in which prioritised traffic classes are introduced to support QoS in broadband wireless network, as is the case with the 802.16 (WiMAX) standard or with the 802.11p standard for Vehicular Ad Hoc Networks (VANETs), i.e. a relavant class of networks in the realm of Intelligent Transportation Systems.

Performance analysis of wireless network systems based on IEEE MAC protocols at-

tracted the attention of researchers also in the formal methods community. Inspired from a study by Jeremy Sproston *et al.* [KNS02] where performance properties of the *basic access* (BA) version of the 802.11 MAC were formally assessed through the PRISM model checker over a (toy) single-hop wireless network, together with Alice Miller (University of Glasgow) we addressed formal performance analysis of the more complex *ready-to-send/clear-to-receive* (RTS/CTS) version of 802.11 MAC and considered the comparison against its energy saving version, namely S-MAC i.e. a backoff collision avoidance mechanism operating on top of a sleep-awake cycle through which the nodes' radio is periodically turned off so to preserve the battery consumption in energy constrained wireless sensor nodes. To account for more realistic situations we considered a larger three-hops network and assessed how performance indicators (*transmission latency*) trade off against energy indicators (*energy consumption*) for transmission of data packets over a 3-hops [BM06]. The main limitation of such performance modeling effort is in terms of scaling as the complexity of the underling discrete-time Markov chain model (encoded in the PRISM model checking language) is affected by the network size yielding the necessity to generate very large, error prone, PRISM code which also quickly lead to saturating the memory when model checking algorithms are run on them.

In order to allow the modellers to take into account for larger networks consisting of an arbitrary number of stations, in collaboration with Nicolas Vasselin and Benoit Barbot (Université Paris-Est Créteil), we developed a novel modelling framework for performance analysis of protocols of the IEEE 801.11 MAC family and specifically suitable for comparing the standard RTS/CTS 802.11 MAC for wireless local area networks (WLAN) against its prioritised version used in Vehicular Ad Hoc Networks (VANETs) [BBV19]. This framework relies on statistical model checking (the COSMOS platform) hence removes the memory limitations numerical model checking approaches suffer of, and is based on a high-level stochastic Petri net formalisms, namely the so-called symmetric stochastic nets, which provides for a compact representation of complex systems consisting of a multitude of components copies. Based on the expressiveness of coloured Petri nets formalism the framework's is highly configurable as, for example, the number of stations the network consists of corresponds with a *color domain* of the Petri net model. Therefore the framework can account for i) an arbitrarily large number of stations, ii) different traffic conditions (saturated/non-saturated), iii) different hypothesis concerning the shared channel (ideal/non-ideal).

Another relevant class of protocols is the 802.16 WiMAX standard, which addresses QoS needs in broadband networks by means of dedicated mechanisms, such as the Admission Control (AC), the Traffic Policer (TP) and the Traffic Shaper (TS) with different responsibility in the QoS management process. The AC which resides on the base-station (BS), establishes whether to accept or reject an incoming connection request according to the negotiated QoS parameters (maximum sustained traffic rate, minimum reserved traffic rate, and maximum latency), the TP, which resides on subscriber-stations (SSs), drops incoming requests that violate a pre-negotiated QoS agreements, while the TS, as opposed

to the TP, delays (rather than dropping) incoming requests violating a pre-negotiated QoS agreement. More specifically the role of TS is to send packets into the network according to the QoS needed by each service class and avoiding congestion. In collaboration with Lynda Mokdad (University Paris-Est Créteil) and Jalel Ben-Othman (University Paris 13) we analysed QoS tradeoffs between strict (deterministic) prioritised TS versus a novel probabilistic TS policy [BBM12].

Another relevant issue related to the security of (the energy constrained) wireless sensor networks is that of detection of denial-of-service (DoS) attacks. A DoS attack takes place when an intruder manages to get control of a network node and inject a mischievous behaviour for example by flooding the network with unwanted traffic leading to draining of the nodes' batteries hence ceasing the network functioning. Mechanisms for detection of DoS attacks in hierarchically clustered WSNs assign a node a *control* responsibility of analysing the traffic inside a cluster and to send warnings to the cluster head whenever an abnormal behavior (e.g., high packets throughput) is detected. It is therefore paramount to avoid control nodes (cNodes) to run out of battery leaving the network unprotected. As part of the PhD thesis work of Quentin Monet (University Paris-Est Créteil), we proposed new mechanisms for DoS detection. In a first work [BMM13] we proposed a novel dynamic cNodes displacement scheme according to which cNodes are periodically elected, at random, among ordinary nodes of each cluster in order to improve the cNode battery lifetime hence the network's w.r.t. the classical mechanisms whereby cNodes are statically displaced in strategic positions within the network topology. In a follow up work we extended that approach by proposing additional dynamic cNode election schemes in which residual energy level is also taken into account amongst the criteria for periodically electing a cluster's cNode. We demonstrated the effectiveness of the newly introduced dynamic cNode schemes through experiments run on the network simulation platform ns-2 whose outcomes showed improvements of the load balancing in the network, while maintaining good detection coverage, w.r.t. static cNode schemes [MMB+17].

Jump-diffusion approximated analysis of density dependent Markov chains. *Density dependent (continuous time) Markov chains* (DDMC) is a subclass of CTMC used to model populations of interacting objects whereby the intensity of the interactions (state-transitions) can be expressed as a function of a parameter N that represents the density of the objects present in a given volume (e.g a cell's volume), which makes chemical reaction networks (i.e. systems consisting of biochemical species interacting within a cell of given volume) a typical case of system suitable for DDMC modelling. It has been shown that DDMCs can be approximated by less computationally hungry processes with a "fluid" state space, i.e. either by ordinary differential equations (ODEs) or by stochastic differential equations (SDE) [Kur76]. Differently from ODEs approximation, SDE approximation of DDMC, known as *diffusion* approximation, preserve the stochasticity of the original DDMC, which on the contrary is lost with ODE approximation. A limitation of

the DDMC diffusion approximation is that it can be applied only to models where the probability of reaching a boundary of the state space of the process is negligible, which motivated the introduction of so-called *jump diffusion* approximation where essentially the SDE used to approximate a given DDMC is enriched with a (discrete) Poisson counting component that gives rise to jumps that mimic the behavior of the original CTMC at the boundaries [BS17, BBH$^+$14].

In collaboration with Marco Beccuti, Enrico Bibbona, András Horváth, Roberta Sirovich and Jeremy Sproston from University of Turin we developed a framework to estimate the probability of timed properties via *jump diffusion approximation* of large DDMC models of biological pathways [BBB$^+$17]. More specifically we demonstrated that statistical estimation of probabilistic time-bounded properties against DDMC models of chemical reaction networks can take advantage of jump-diffusion approximations. Specifically we showed that by replacing (costly) "exact" trajectories sampled from the DDMC through Gillespie's stochastic simulation algorithm (SSA) with "approximated" trajectories sampled through jump-diffusion approximation one can obtain a remarkable speed up in estimating the probability of time bounded measures.

Chapter 7

Conclusion and perspectives

This thesis manuscript summarised some of my contributions in the domain of formal modelling for complex systems featuring probabilities and parameters. More specifically Chapter 2 gave an overview of an expressive, hybrid automata based, framework for statistical model checking that targets a very generic class of stochastic processes, while Chapter 3 tackled the problem of reverse-engineering of parametric Markov chains model, by introducing a framework that allows the modeller to tune a stochastic model's parameters so that the model's meets given, formally expressed, requirements. Chapter 4 then showed the relevance of the approaches discussed in Chapter 2 and Chapter 3, by considering a non-trivial problem, that is, the formal analysis of stochastic oscillators. Finally Chapter 5 presented a recent contribution tackling the problem of deriving stochastic models capable of reproducing the periodic behaviour of a dataset representing an observed phenomenon. Ongoing work and perspectives were given at the end of each chapter, I recall the main ones here and more generally discuss directions I envision for my research in the coming years.

Speeding up statistical model checking. Despite low *space complexity* SMC runtime can be remarkably high due to the cost of sampling (large amount of) paths through stochastic simulation algorithms. Indeed existing SMC frameworks use *exact* stochastic simulation sampling schemes, such as Gillespie's [Gil77], through which each *jump* in a path as a random duration and is obtained through a *stochastic race* between all enabled events. In this manner the unfolding of a path can become extremely costly particularly for models with a high degree of concurrency. In an attempt to reduce the simulation runtime *approximate* stochastic simulation algorithms, the basic idea being to speed up the simulation process by replacing the "exact" random long jumps of a path with *discrete jumps* of arbitrary fixed duration yielding an approximated path. Such approximated algorithms have been introduced in the system's biology community as means to simulate large chemical reaction networks but, at the best of my knowledge, have not been considered in the SMC context. I have been starting looking at the problem of taking advantage of

approximated simulation scheme within SMC, the main problem to tackle in this respect is to figure out how to compensate or limit the inherent error that approximated paths induce on the estimation of the satisfaction probability of a temporal logic formula. A promising approach in this respect is the multi-level simulation algorithm [LBGY16] which based on the *thickening property* of Poisson random variables, allows for cleverly generating, in parallel, paths at different level of precision and progressively compensating the error of approximated paths hence to obtain precise estimations at lower computational cost than exact schemes.

Evolution of the automaton-ABC parametric verification framework. The parametric model checking framework presented in Chapter 3 is, in its current formulation, limited to non-nested spatio-temporal properties. Even though this kind of properties is that found in most case studies in the literature it would be worth to work out a generalisation to the full MITL spectrum, similarly to what has been done in studies that address the robustness of MITL against real valued signals of non-probabilistic models [DM10]. This would entail to investigate the possibility to come up with a satisfiability distance definition based on the recursive syntax of MITL formulae, and then to work out a machinery for assessing distances of nested formulae against trajectories sampled from a CTMC model, which might not be an easy task considering that in automata-based model checking approaches, the characterisation of automata capable of accepting paths that satisfy nested temporal formulae turned out not an easy task. Another relevant evolution I wish to engage with is that of integration of the automaton-ABC framework within the COSMOS tool. To this day only a prototype implementation (written in the Julia programming language `https://gitlab-research.centralesupelec.fr/2017bentrioum/tcs2021`) of the parametric verification framework exists, which although computationally efficient is little user friendly (i.e. the necessary *distance automata* must be user as objects of some Julia class). The integration within the COSMOS platform is highly desirable as it would allow the user to rely on the HASL formalism all together, hence allowing one to express models as stochastic Petri nets and to encode the distance automata directly as a LHA.

Stochastic process mining. As part of the PhD work of Pierre Cry (first year PhD student at MICS laboratory of CentraleSupélec I am co-supervising since October 2022) I recently got interested in the extensions of process mining to the realm of stochastic models. Process mining algorithms are concerned with the derivation of formal models, usually in form of a workflow Petri net, capable of matching observations stored in *event logs* (collections of traces consisting of sequences of events witnessing executions of the observed process). The quality of mined models is established by comparing the behavior *seen* in the event-log with that exhibited by the model, w.r.t. different criteria including *fitness* (how much of the event-log behavior is reproduced by the model) and *precision* (how much of the model's behavior is not contained in the event-log). The vast majority of process mining frameworks aim untimed, non-probabilistic behaviour, i.e., they only account for the temporal order of events observed in the log while completely ignoring

timing and probability of event occurrence. The extension of process mining to stochastic models seems little explored so far in the literature. From a theoretical point of view two type of problems seem relevant for extending process mining to the stochastic realm. First one have to adapt mining algorithms so that they issue stochastic models based on "timed" event logs, something which, intuitively, seem to require a fitting phase where the stochastic distributions to be used in the stochastic (Petri net) model are established based on the timing information contained in the log. Secondly *conformance checking* of mined models will also need to be adapted to the stochastic case, and that likely will entail adaptation of methods for comparing *stochastic languages*, such as, e.g. the (Wasserstein distance) earth movers method.

Data-driven predictive modeling of periodic phenomena. In Chapter 5 we presented preliminary results about the derivation of models that reproduce the periodicity exhibited by observations stored in a dataset. Following a rather empirical approach we showed that the periodicity of the data can be accurately mimicked by imposing a corresponding "periodicity" in the structure of the background chain of a hidden Markov model and more specifically that sustained periodicity can be obtained by means of a "clock" background chain, i.e., a "deterministic" chain consisting of as many states as the number of (discretised) intervals in which observations are grouped and whose states form a loop through a sequence of probability 1 transitions. These preliminary findings calls for further developments. First of all it is worth investigating whether the using of less constrained periodic background chain (not necessarily clock chains) can still lead to reproduce the sustained periodicity of the data. Also through our experiments we observed that parameter estimation (expectation maximisation) algorithms used to fit the HMM to the dataset seem to converge immediately, which suggests that an explicit solution for parameter estimation might exist for this specific class of models, that is, for HMM with periodic background chain. Furthermore the overall approach should be generalised so to address other cases, such as, that of *continuous data* as opposed to *discrete data* (i.e. number of arrivals) we took into account so far. In this respect it will also worth considering the use of more generic stochastic processes in order to generate the actual observations in each state of the background chain. So far, since we addressed discrete dataset only, we only considered point Poisson process as the type of process to generate the observations. It'd be worth considering Phase type distribution both discrete (in case of discrete dataset) and continuous (in case of continuous dataset). That would entail also investigating how known expectation maximisation and moments fitting methods would cope with fitting the parameters in the case of this particular class of (periodic) structured HMM. In collaboration with András Horváth (University of Turin) we have submitted a PhD project proposal to a funding program for a co-supervising (*cotutelle*) PhD thesis between University of Turin and University Paris Saclay for which we hope to get funded.

The expected outcome fo such a PhD project is twofold. From a practical point of view we expect that an effective framework is developed for mining of models based on periodic

dataset. Such framework should allow for (semi) automatic derivation of stochastic models that accurately reproduce the periodic character of the dataset and should be validated through different datasets we have access to, both from hospital emergency department as well as from manufacturing industry domain. From a theoretical standpoint we expect that the working out of such a framework will likely lead to novel theoretical results, such as, for example, the introduction of novel parameter estimation procedure optimised to the case of structured periodic stochastic models.

Bibliography

[ABB+13] Elvio Gilberto Amparore, Paolo Ballarini, Marco Beccuti, Susanna Donatelli, and Giuliana Franceschinis. Expressing and computing passage time measures of GSPN models with HASL. In José Manuel Colom and Jörg Desel, editors, *Application and Theory of Petri Nets and Concurrency - 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings*, volume 7927 of *Lecture Notes in Computer Science*, pages 110–129. Springer, 2013.

[ABDF11] E.G. Amparore, M. Beccuti, S. Donatelli, and G. Franceschinis. Probe automata for passage time specification. In *Quantitative Evaluation of Systems (QEST), 2011 Eighth International Conference on*, pages 101 –110, sept. 2011.

[ACD91] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for probabilistic real-time systems. In *ICALP'91*, LNCS 510, 1991.

[ACH11] F. Avram, D.F. Chedom, and A. Horváth. On moments based Padé approximations of ruin probabilities. *Journal of Computational and Applied Mathematics*, 235(10):3215 – 3228, 2011.

[ACHG16] Alessio Angius, Marcello Colledani, András Horváth, and Stanley B Gershwin. Analysis of the lead time distribution in closed loop manufacturing systems. *IFAC-PapersOnLine*, 49(12):307–312, 2016.

[ACHH92] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, LNCS 736, 1992.

[ACHH93] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors, *Hybrid Systems*, pages 209–229, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2), 1994.

[AHC14] A. Angius, A. Horváth, and M. Colledani. Moments of accumulated reward and completion time in Markovian models with application to unreliable manufacturing systems. *Performance Evaluation*, 75:69–88, 2014.

[AMBC+95] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, 1995.

[ANJB10] Matthew A Nunes and David J Balding. On optimal selection of summary statistics for approximate bayesian computation. *Statistical applications in genetics and molecular biology*, 9:Article34, 01 2010.

[AP18] Gul Agha and Karl Palmskog. A survey of statistical model checking. *ACM Transactions on Modeling and Computer Simulation*, 28:1–39, 01 2018.

[ASSB96] Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert Brayton. Verifying continuous time markov chains. In Rajeev Alur and Thomas A. Henzinger, editors, *Computer Aided Verification*, pages 269–276, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[ASSB00] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model-checking CTMCs. *ACM Trans. on Computational Logic*, 1(1), 2000.

[Bal15a] Paolo Ballarini. Analysing oscillatory trends of discrete-state stochastic processes through hasl statistical model checking. *International Journal on Software Tools for Technology Transfer*, pages 1–22, 2015.

[Bal15b] Paolo Ballarini. Analysing oscillatory trends of discrete-state stochastic processes through HASL statistical model checking. *STTT*, 17(4):505–526, 2015.

[BB22] Paolo Ballarini and Benoît Barbot. Cosmos: Evolution of a statistical model checking platform. *SIGMETRICS Perform. Evaluation Rev.*, 49(4):65–69, 2022.

[BBB+17] Paolo Ballarini, Marco Beccuti, Enrico Bibbona, András Horváth, Roberta Sirovich, and Jeremy Sproston. Analysis of timed properties using the jump-diffusion approximation. In Philipp Reinecke and Antinisca Di Marco, editors, *Computer Performance Engineering - 14th European Workshop, EPEW 2017, Berlin, Germany, September 7-8, 2017, Proceedings*, volume 10497 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 2017.

[BBC19] Mahmoud Bentriou, Paolo Ballarini, and Paul-Henry Cournède. Reachability design through approximate bayesian computation. In *International Conference on Computational Methods in Systems Biology*, pages 207–223. Springer, 2019.

[BBC21] Mahmoud Bentriou, Paolo Ballarini, and Paul-Henry Cournède. Automaton-abc: A statistical method to estimate the probability of spatio-temporal properties for parametric markov population models. *Theor. Comput. Sci.*, 893:191–219, 2021.

[BBD+15a] P. Ballarini, B. Barbot, M. Duflot, S. Haddad, and N. Pekergin. Hasl: A new approach for performance evaluation and model checking from concepts to experimentation. *Performance Evaluation*, 90:53 – 77, 2015.

[BBD+15b] Paolo Ballarini, Benoît Barbot, Marie Duflot, Serge Haddad, and Nihal Pekergin. HASL: A new approach for performance evaluation and model checking from concepts to experimentation. *Perform. Eval.*, 90:53–77, 2015.

[BBD+15c] Paolo Ballarini, Benoît Barbot, Marie Duflot, Serge Haddad, and Nihal Pekergin. Hasl: A new approach for performance evaluation and model checking from concepts to experimentation. *Perform. Eval.*, 90:53–77, 2015.

[BBDH17] Benoît Barbot, Béatrice Bérard, Yann Duplouy, and Serge Haddad. Statistical Model-Checking for Autonomous Vehicle Safety Validation. In *Conference SIA Simulation Numérique*, Montigny-le-Bretonneux, France, March 2017. Société des Ingénieurs de l'Automobile.

[BBH+13] Paolo Ballarini, Nathalie Bertrand, András Horváth, Marco Paolieri, and Enrico Vicario. Transient analysis of networks of stochastic timed automata using stochastic state classes. In Kaustubh R. Joshi, Markus Siegle, Mariëlle Stoelinga, and Pedro R. D'Argenio, editors, *Quantitative Evaluation of Systems - 10th International Conference, QEST 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*, volume 8054 of *Lecture Notes in Computer Science*, pages 355–371. Springer, 2013.

[BBH+14] Marco Beccuti, Enrico Bibbona, András Horváth, Roberta Sirovich, Alessio Angius, and Gianfranco Balbo. Analysis of Petri net models through stochastic differential equations. In *Proc. PETRI NETS 2014*, volume 8489 of *LNCS*, pages 273–293. Springer, 2014.

[BBM12] Paolo Ballarini, Jalel Ben-Othman, and Lynda Mokdad. Quantitative verification of wimax traffic shaping solutions. In Elhadi M. Shakshuki and Muhammad Younas, editors, *Proceedings of the 3rd International Conference*

on Ambient Systems, Networks and Technologies (ANT 2012), the 9th International Conference on Mobile Web Information Systems (MobiWIS-2012), Niagara Falls, Ontario, Canada, August 27-29, 2012, volume 10 of *Procedia Computer Science*, pages 1026–1031. Elsevier, 2012.

[BBV19] Paolo Ballarini, Benoît Barbot, and Nicolas Vasselin. Performance modelling of access control mechanisms for local and vehicular wireless networks. *CoRR*, abs/1901.04285, 2019.

[BC19] Luca Bortolussi and Francesca Cairoli. Bayesian abstraction of Markov population models. In *International Conference on Quantitative Evaluation of Systems (QEST2019)*, pages 259–276. Springer, 2019.

[BCH+07] C. Baier, L. Cloth, B. Haverkort, M. Kuntz, and M. Siegle. Model checking action- and state-labelled Markov chains. *IEEE Trans. on Software Eng.*, 33(4), 2007.

[BCMR09] Mark A. Beaumont, Jean-Marie Cornuet, Jean-Michel Marin, and Christian P. Robert. Adaptive approximate bayesian computation. *Biometrika*, 96(4):983–990, 2009.

[BCRG04] Gilles Bernot, Jean-Paul Comet, Adrien Richard, and Janine Guespin. Application of formal methods to biological regulatory networks: extending thomas' asynchronous logical approach with temporal logic. *J Theor Biol*, 229(3):339–347, Aug 2004.

[BD15] Paolo Ballarini and Marie Duflot. Applications of an expressive statistical model checking approach to the analysis of genetic circuits. *Theor. Comput. Sci.*, 599:4–33, 2015.

[BDD+10] Paolo Ballarini, Hilal Djafri, Marie Duflot, Serge Haddad, and Nihal Pekergin. HASL: an expressive language for statistical verification of stochastic models. Technical Report TR-LACL-2010-8, LACL University of Paris-Est Créteil, 2010.

[BDD+11a] P. Ballarini, H. Djafri, M. Duflot, S. Haddad, and N. Pekergin. COSMOS: a statistical model checker for the hybrid automata stochastic logic. In *Proceedings of the 8th International Conference on Quantitative Evaluation of Systems (QEST'11)*, pages 143–144. IEEE Computer Society Press, sep. 2011.

[BDD+11b] Paolo Ballarini, Hilal Djafri, Marie Duflot, Serge Haddad, and Nihal Pekergin. HASL : an expressive language for statistical verification of stochastic models. In *Proc. Valuetools'2011*, pages 306–315, 2011.

[BDD+11c] Paolo Ballarini, Hilal Djafri, Marie Duflot, Serge Haddad, and Nihal Pekergin. HASL: An expressive language for statistical verification of stochastic models. In Pete Harrison Samson Lasaulce, Dieter Fiems and Luc Vandendorpe, editors, *Proceedings of the 5th International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'11)*, pages 306–315, Cachan, France, May 2011.

[BDD+11d] Paolo Ballarini, Hilal Djafri, Marie Duflot, Serge Haddad, and Nihal Pekergin. Petri nets compositional modeling and verification of flexible manufacturing systems. In *2011 IEEE International Conference on Automation Science and Engineering*, pages 588–593, 2011.

[BDHA20] Paolo Ballarini, Davide Duma, András Horváth, and Roberto Aringhieri. Petri nets validation of markovian models of emergency department arrivals. In Ryszard Janicki, Natalia Sidorova, and Thomas Chatain, editors, *Application and Theory of Petri Nets and Concurrency - 41st International Conference, PETRI NETS 2020, Paris, France, June 24-25, 2020, Proceedings*, volume 12152 of *Lecture Notes in Computer Science*, pages 219–238. Springer, 2020.

[BDL+12] Peter E. Bulychev, Alexandre David, Kim Guldstrand Larsen, Marius Mikucionis, Danny Bøgsted Poulsen, Axel Legay, and Zheng Wang. Uppaal-smc: Statistical model checking for priced timed automata. In Herbert Wiklicky and Mieke Massink, editors, *QAPL*, volume 85, pages 1–16, 2012.

[BDPF09] G. Balbo, M. De Pierro, and G. Franceschinis. Tagged Generalized Stochastic Petri Nets. In *Computer Performance Engineering*, volume LNCS 5652, pages 1–15, Berlin, 2009. Springer-Verlag.

[Ben21] Mahmoud Bentriou. *Statistical Inference and Verification of Chemical Reaction Networks*. PhD thesis, 12 2021.

[BG10] P. Ballarini and M.L. Guerriero. Query-based verification of qualitative trends and oscillations in biochemical systems. *Theoretical Computer Science*, 411(20):2019 – 2036, 2010.

[BGGM14] Paolo Ballarini, Emmanuelle Gallet, Pascale Le Gall, and Matthieu Manceny. Formal analysis of the wnt/β-catenin through statistical model checking. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications - 6th International Symposium, ISoLA 2014, Imperial, Corfu, Greece, October 8-11, 2014, Proceedings, Part II*, volume 8803 of *Lecture Notes in Computer Science*, pages 193–207. Springer, 2014.

[BH08] Paolo Ballarini and András Horváth. Memory efficient calculation of path probabilities in large structured markov chains. In *Fifth International Conference on the Quantitative Evaluaiton of Systems (QEST 2008), 14-17 September 2008, Saint-Malo, France*, pages 157–166. IEEE Computer Society, 2008.

[BH09a] Paolo Ballarini and András Horváth. Compositional model checking of product-form ctmcs. *Electronic Notes in Theoretical Computer Science*, 250(1):21–37, 2009. Proceedings of the Seventh International Workshop on Automated Verification of Critical Systems (AVoCS 2007).

[BH09b] Paolo Ballarini and András Horváth. Memory efficient analysis for a class of large structured markov chains: work in progress. In Giovanni Stea, Jean Mairesse, and José Mendes, editors, *4th International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS '09, Pisa, Italy, October 20-22, 2009*, page 21. ICST/ACM, 2009.

[BH21a] Paolo Ballarini and András Horváth. Formal analysis of production line systems byprobabilistic model checking tools. In *Proceedings of the 2021 IEEE Emerging Technology and Factory Automation (ETFA)*, 2021.

[BH21b] Paolo Ballarini and András Horváth. Performance analysis of production lines through statistical model checking. In Paolo Ballarini, Hind Castel, Ioannis Dimitriou, Mauro Iacono, Tuan Phung-Duc, and Joris Walraevens, editors, *Performance Engineering and Stochastic Modeling - 17th European Workshop, EPEW 2021, and 26th International Conference, ASMTA 2021, Virtual Event, December 9-10 and December 13-14, 2021, Proceedings*, volume 13104 of *Lecture Notes in Computer Science*, pages 264–281. Springer, 2021.

[BHHK00] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. On the logical characterisation of performability properties. In *ICALP'00*, LNCS 1853, 2000.

[BHHK03a] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for CTMCs. *IEEE Trans. on Software Eng.*, 29(6), 2003.

[BHHK03b] Christel Baier, Boudewijn Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model-checking algorithms for continuous-time markov chains. *Software Engineering, IEEE Transactions on*, 29:524– 541, 07 2003.

[BHHT08] L. Bodrog, A. Heindl, G. Horváth, and M. Telek. A Markovian canonical form of second-order matrix-exponential processes. *European Journal of Operation Research*, 190:459–477, 2008.

[BHT08] L. Bodrog, A. Horváth, and M. Telek. Moment characterization of matrix exponential and Markovian arrival processes. *Annals of Operations Research*, 160:51–68, 2008.

[BKMP15] Benoît Barbot, Marta Kwiatkowska, Alexandru Mereacre, and Nicola Paoletti. Estimation and verification of hybrid heart models for personalised medical and wearable devices. In *13th International Conference on Computational Methods in Systems Biology (CMSB 2015)*, volume 9308 of *LNCS*, pages 3–7. Springer, 2015.

[BM06] Paolo Ballarini and Alice Miller. Model checking medium access control for sensor networks. In *Leveraging Applications of Formal Methods, Second International Symposium, ISoLA 2006, Paphos, Cyprus, 15-19 November 2006*, pages 255–262. IEEE Computer Society, 2006.

[BMM09] Paolo Ballarini, Radu Mardare, and Ivan Mura. Analysing Biochemical Oscillation through Probabilistic Model Checking. *Electronic Notes in Theoretical Computer Science*, 2009.

[BMM13] Paolo Ballarini, Lynda Mokdad, and Quentin Monnet. Modeling tools for detecting dos attacks in wsns. *Secur. Commun. Networks*, 6(4):420–436, 2013.

[BMR12] Paolo Ballarini, Jarno Mäkelä, and Andre S. Ribeiro. Expressive statistical model checking of genetic networks with delayed stochastic dynamics. In David R. Gilbert and Monika Heiner, editors, *Computational Methods in Systems Biology - 10th International Conference, CMSB 2012, London, UK, October 3-5, 2012. Proceedings*, volume 7605 of *Lecture Notes in Computer Science*, pages 29–48. Springer, 2012.

[BMS16] Luca Bortolussi, Dimitrios Milios, and Guido Sanguinetti. Smoothed model checking for uncertain continuous-time markov chains. *Inf. Comput.*, 247:235–253, 2016.

[BPSV05] G. Bucci, R. Piovosi, L. Sassoli, and E. Vicario. Introducing probability within state class analysis of dense-time-dependent systems. In *Second International Conference on the Quantitative Evaluation of Systems (QEST'05)*, pages 13–22, 2005.

[BPSW70] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[BS17] Enrico Bibbona and Roberta Sirovich. Strong approximation of density dependent Markov chains on bounded domains by jump diffusion processes. Technical report, Università di Torino, 2017.

[Buc03] P. Buchholz. *An EM-Algorithm for MAP Fitting from Real Traffic Data*, pages 218–236. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[BUT]

[CAH14] Marcello Colledani, Alessio Angius, and András Horváth. Lead time distribution in unreliable production lines processing perishable products. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–8, 2014.

[Can88] J. Canny. Some algebraic and geometric computations in pspace. In *Proc. STOC'88*, 1988.

[Car09] Luca Cardelli. *Artificial Biochemistry*, pages 429–462. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[CD18] José Chacón and Tarn Duong. *Multivariate Kernel Smoothing and its Applications*. 2018.

[CDM11] Taolue Chen, Marta Diciolla, Marco an Kwiatkowska, and Alexandru Mereacre. Time-bounded verification of ctmcs against real-time specifications. In *9th International Conference, FORMATS 2011*, Lecture Notes in Computer Science, pages 26–42. Springer, 2011.

[CDP+17] Milan Ceska, Frits Dannenberg, Nicola Paoletti, Marta Kwiatkowska, and Lubos Brim. Precise parameter synthesis for stochastic biochemical systems. *Acta Informatica*, 54(6):589–623, 2017.

[CG08] Allan Clark and Stephen Gilmore. State-aware performance analysis with extended stochastic probes. In *Proceedings of the 5th European Performance Engineering Workshop on Computer Performance Engineering*, EPEW '08, pages 125–140, Berlin, Heidelberg, 2008. Springer-Verlag.

[CHA15] M. Colledani, A. Horváth, and Alessio Angius. Production quality performance in manufacturing systems processing deteriorating products. *Cirp Annals-manufacturing Technology*, 64:431–434, 2015.

[Che99] Song Chen. Beta kernel estimators for density functions. *Computational Statistics & Data Analysis*, 31(2):131–145, 1999.

[CHKM09] T. Chen, T. Han, J.-P. Katoen, and A. Mereacre. Quantitative model checking of CTMC against timed automata specifications. In *Proc. LICS'09*, 2009.

[CKKP05] L. Cloth, J.-P. Katoen, M. Khattri, and R. Pulungan. Model checking Markov reward models with impulse rewards. In *Proc. International Conference on Dependable Systems and Networks (DSN'05)*, pages 722–731. IEEE Computer Society Press, 2005.

[COS] Cosmos home page. https://cosmos.lacl.fr/.

[CP34] C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(26):404–413, 1934.

[CR65] Y. S. Chow and Herbert Robbins. On the asymptotic theory of fixed-width sequential confidence intervals for the mean. *Annals of Mathematical Statistics*, 36(2):457–462, 1965.

[CRCD+04] Nathalie Chabrier-Rivier, Marc Chiaverini, Vincent Danos, François Fages, and Vincent Schächter. Modeling and querying biomolecular interaction networks. *Theoretical Computer Science*, 325(1):25–44, 2004. Computational Systems Biology.

[CT12] M. Colledani and T. Tolio. Integrated quality, production logistics and maintenance analysis of multi-stage asynchronous manufacturing systems with degrading machines. *CIRP Annals-Manufacturing Technology*, 61/1:455–458, 2012.

[DHK04] Nicholas J. Dingle, Peter G. Harrison, and William J. Knottenbelt. Uniformisation and Hypergraph Partitioning for the Distributed Computation of Response Time Densities in Very Large Markov Models. *Journal of Parallel and Distributed Computing*, 64(8):309–920, August 2004.

[DHS07] Susanna Donatelli, Serge Haddad, and Jeremy Sproston. Csl^{TA}: an expressive logic for continuous-time markov chains. In *Fourth International Conference on the Quantitative Evaluation of Systems (QEST 2007)*, pages 31–40, 2007.

[DHS09] S. Donatelli, S. Haddad, and J. Sproston. Model checking timed and stochastic properties with CSL^{TA}. *IEEE Trans. on Software Eng.*, 35, 2009.

[DJKV17] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In *Proc. 29th International Conference on Computer Aided Verification (CAV'17)*, 2017.

[DM10] Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *Proceedings of the 8th International Conference on Formal Modeling and Analysis of Timed Systems*, FORMATS'10, pages 92–106, Berlin, Heidelberg, 2010. Springer-Verlag.

[DMDJ06] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society B*, 68(3):411–436, 2006.

[EC80] E. A. Emerson and E. M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *Proc. of the 7th Coll. on Automata, Languages and Programming*, LNCS 85, 1980.

[EH86] E. Allen Emerson and Joseph Y. Halpern. "sometimes" and "not never" revisited: on branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986.

[EL00] Michael B. Elowitz and Stanislas Leibier. A synthetic oscillatory network of transcriptional regulators. *Nature*, 2000.

[Gil77] D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[Gil01] Daniel T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, 115(4):1716–1733, 2001.

[Gly83] P. W. Glynn. On the role of generalized semi-Markov processes in simulation output analysis. In *Proc. Conf. Winter simulation*, 1983.

[GMGB14] Emmanuelle Gallet, Matthieu Manceny, Pascale Le Gall, and Paolo Ballarini. An LTL model checking approach for biological parameter inference. In Stephan Merz and Jun Pang, editors, *Formal Methods and Software Engineering - 16th International Conference on Formal Engineering Methods, ICFEM 2014, Luxembourg, Luxembourg, November 3-5, 2014. Proceedings*, volume 8829 of *Lecture Notes in Computer Science*, pages 155–170. Springer, 2014.

[GMGB15] Emmanuelle Gallet, Matthieu Manceny, Pascale Le Gall, and Paolo Ballarini. Étude de réseaux de thomas par validation de propriétés LTL pour pseudomonas aeruginosa. *Tech. Sci. Informatiques*, 34(5):575–600, 2015.

[Gol02] Albert Goldbeter. Computational approaches to cellular rhythms. *Nature*, 420:238–245, 2002.

[HHWT97] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. Hytech: A model checker for hybrid systems. In Orna Grumberg, editor, *Computer Aided Verification*, pages 460–463, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[Hil05] J. Hillston. Process algebras for quantitative analysis. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*, pages 239–248, Washington, DC, USA, 2005. IEEE Computer Society.

[HJ90] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reability. Technical report, 1990.

[HJB+10] R. He, P. Jennings, S. Basu, A. P. Ghosh, and H. Wu. A bounded statistical approach for model checking of unbounded until properties. In *Proc. ASE'10*, 2010.

[HJK+22] Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. The probabilistic model checker storm. *International Journal on Software Tools for Technology Transfer*, 24(4):589–610, 2022.

[HKM08] T. Han, J. Katoen, and A. Mereacre. Approximate parameter synthesis for probabilistic time-bounded reachability. In *2008 Real-Time Systems Symposium*, pages 173–182, 2008.

[HKPV98] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1):94–124, 1998.

[HLMP04] Thomas Hérault, Richard Lassaigne, Frédéric Magniette, and Sylvain Peyronnet. Approximate probabilistic model checking. In Bernhard Steffen and Giorgio Levi, editors, *Verification, Model Checking, and Abstract Interpretation*, pages 73–84, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[HLP06] T. Herault, R. Lassaigne, and S. Peyronnet. APMC 3.0: Approximate verification of discrete and continuous time Markov chains. In *Proc. QEST'06*, 2006.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):pp. 13–30, 1963.

[Hor13] G. Horváth. Matching marginal moments and lag autocorrelations with maps. In *Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools*, ValueTools '13, pages 59–68, ICST, Brussels, Belgium, Belgium, 2013. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[HPRV11] András Horváth, Marco Paolieri, Lorenzo Ridi, and Enrico Vicario. Probabilistic model checking of non-markovian models with concurrent generally distributed timers. In *Eighth International Conference on Quantitative Evaluation of Systems, QEST 2011, Aachen, Germany, 5-8 September, 2011*, pages 131–140. IEEE Computer Society, 2011.

[HRS13] M Heiner, C Rohr, and M Schwarick. MARCIE - Model checking And Reachability analysis done effiCIEntly. In JM Colom and J Desel, editors, *Proc. PETRI NETS 2013*, volume 7927 of *LNCS*, pages 389–399. Springer, June 2013.

[HT07] A. Horváth and M. Telek. Matching more than three moments with acyclic phase type distributions. *Stochastic Models*, 23(2):167–194, 2007.

[JLS12] Cyrille Jégourel, Axel Legay, and Sean Sedwards. A platform for high performance statistical model checking - plasma. In Cormac Flanagan and Barbara König, editors, *TACAS*, volume 7214, pages 498–503. Springer, 2012.

[JSD18] Cyrille Jegourel, Jun Sun, and Jin Song Dong. On the sequential massart algorithm for statistical model checking. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Verification*, pages 287–304, Cham, 2018. Springer International Publishing.

[KBVC16] Konstantinos Koutroumpas, Paolo Ballarini, Irene Votsi, and Paul Henry Cournède. Bayesian parameter estimation for the Wnt pathway: An infinite mixture models approach. volume 32, pages 781–789, 2016.

[KCAB09] Zohra Khalis, Jean-Paul Comet, Richard Adrien, and Gilles Bernot. The smbionet method for discovering models of gene regulatory networks. *Genes, Genomes and Genomics*, 3, 01 2009.

[KM27] W. O. Kermack and A. G. McKendrick. A Contribution to the Mathematical Theory of Epidemics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 115(772):700–721, 1927.

[KNP02] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism: Probabilistic symbolic model checker. pages 200–204. Springer, 2002.

[KNP07a] M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation*, volume 4486 of *LNCS*, pages 220–270. Springer, 2007.

[KNP07b] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Stochastic model checking. In Marco Bernardo and Jane Hillston, editors, *SFM*, volume 4486, pages 220–270. Springer, 2007.

[KNP11] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[KNS02] Marta Kwiatkowska, Gethin Norman, and Jeremy Sproston. Probabilistic model checking of the ieee 802.11 wireless local area network protocol. In Holger Hermanns and Roberto Segala, editors, *Process Algebra and Probabilistic Methods: Performance Modeling and Verification*, pages 169–187, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[Knu97] Donald E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[Kul95] V.G. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman & Hall, London, UK, 1995.

[Kul16] V.G. Kulkarni. *Modeling and Analysis of Stochastic Systems, Third Edition*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press, 2016.

[Kur76] Thomas G Kurtz. Limit theorems and diffusion approximations for density dependent Markov chains. In *Stochastic Systems: Modeling, Identification and Optimization, I*, pages 67–78. Springer, 1976.

[KZH+09] Joost-Pieter Katoen, Ivan S. Zapreev, Ernst Moritz Hahn, Holger Hermanns, and David N. Jansen. The ins and outs of the probabilistic model checker MRMC. *QEST*, pages 167–176, 2009.

[LBGY16] Christopher Lester, Ruth E Baker, Michael B Giles, and Christian A Yates. Extending the multi-level method for the simulation of stochastic biological systems. *Bulletin of mathematical biology*, 78(8):1640–1677, 2016.

[LKKP05] L.Cloth, J.-P. Katoen, M. Khattri, and R. Pulungan. Model checking Markov reward models with impulse rewards. In *Proc. DSN'05*, 2005.

[Lot09] Alfred J. Lotka. Contribution to the theory of periodic reactions, January 1909.

[LST16] Axel Legay, Sean Sedwards, and Louis-Marie Traonouez. Plasma lab: A modular statistical model checking platform. In *ISoLA (1)*, volume 9952 of *Lecture Notes in Computer Science*, pages 77–93, 2016.

[MMB+17] Quentin Monnet, Lynda Mokdad, Paolo Ballarini, Youcef Hammal, and Jalel Ben-Othman. Dos detection in wsns: Energy-efficient methods for selecting monitoring nodes. *Concurr. Comput. Pract. Exp.*, 29(23), 2017.

[MN04] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In Yassine Lakhnech and Sergio Yovine, editors, *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[MPRR12] Jean-Michel Marin, Pierre Pudlo, Christian P. Robert, and Robin J. Ryder. Approximate bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, Nov 2012.

[Neu75] M.F. Neuts. Probability distributions of phase type. In *Liber Amicorum Prof. Emeritus H. Florin*, pages 173–206. University of Louvain, 1975.

[Neu79] M.F. Neuts. A versatile markovian point process. *Journal of Applied Probability*, 16:764–779, 1979.

[OD09] H. Okamura and T. Dohi. Faster maximum likelihood estimation algorithms for markovian arrival processes. In *Quantitative Evaluation of Systems, 2009. QEST'09. Sixth International Conference on the*, pages 73–82. IEEE, 2009.

[OS99] W. Douglas Obal, II and William H. Sanders. State-space support for path-based reward variables. *Perform. Eval.*, 35:233–251, May 1999.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 46–57, Washington, DC, USA, 1977. IEEE Computer Society.

[PRI] Prism home page. http://www.prismmodelchecker.org.

[QS82] J. Queille and J. Sifakis. Specification and verification of concurrent systems in cesar. pages 337–351. 1982.

[Rab90] L. R. Rabiner. Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[RP09] Diana El Rabih and Nihal Pekergin. Statistical model checking for steady state dependability verification. In *2009 Second International Conference on Dependability*, pages 166–169, 2009.

[Ryd96] T. Ryden. An em algorithm for estimation in markov-modulated poisson processes. *Computational Statistics & Data Analysis*, 21(4):431 – 447, 1996.

[RZK06] A. Ribeiro, R. Zhu, and S. A. Kauffman. A general modeling strategy for gene regulatory networks with stochastic dynamics. *Journal of computational biology : a journal of computational molecular cell biology*, 13(9):1630–1639, November 2006.

[SFB18] Scott A Sisson, Yanan Fan, and Mark Beaumont. *Handbook of approximate Bayesian computation*. Chapman and Hall/CRC, 2018.

[Sil86] B W Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, London, 1986.

[Spi13] D. Spieler. Characterizing oscillatory and noisy periodic behavior in markov population models. In *Proc. QEST'13*, 2013.

[SVA05a] K. Sen, M. Viswanathan, and G. Agha. VESTA: A statistical model-checker and analyzer for probabilistic systems. In *Proc. QEST'05*, 2005.

[SVA05b] Koushik Sen, Mahesh Viswanathan, and Gul Agha. On statistical model checking of stochastic systems. In *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 266–280. Springer, 2005.

[TH07] M. Telek and G. Horváth. A minimal representation of Markov arrival processes and a moments matching method. *Performance Evaluation*, 64(9-12):1153–1168, Aug. 2007.

[VKBL02] J. Vilar, H.-Y. Kueh, N. Barkai, and S. Leibler. Mechanisms of noise-resistance in genetic oscillators. *Proc. National Academy of Sciences of the United States of America*, 99(9):5988–5992, 2002.

[Vol28] Vito Volterra. Variations and Fluctuations of the Number of Individuals in Animal Species living together. *ICES Journal of Marine Science*, 3(1):3–51, 04 1928.

[Wal45] A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 06 1945.

[You05] Håkan LS Younes. Ymer: A statistical model checker. In *Computer Aided Verification*, pages 429–433. Springer, 2005.

[YS06] H.L.S. Younes and R.G. Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Inf. Comput.*, 204(9), 2006.

[ZC13] André Zuliani, Paolo an Platzer and Edmund M. Clarke. Bayesian statistical model checking with application to stateflow/simulink verification. *Formal Methods in System Design*, 43(2):338–367, 2013.