Automaton-ABC: A statistical method to estimate the probability of spatio-temporal properties for parametric Markov population models

Mahmoud Bentriou, Paolo Ballarini, Paul-Henry Cournède

Version of Record: https://www.sciencedirect.com/science/article/pii/S0304397521005739
Manuscript_010bb9ccce1a73baab6a83ef607b4929

Automaton-ABC: a Statistical Method to Estimate the Probability of Spatio-Temporal properties for parametric Markov Population Models

Mahmoud Bentriou[a,*], Paolo Ballarini[a], Paul-Henry Cournède[a]

[a] *Université Paris-Saclay, CentraleSupélec, Laboratory of Mathematics and Informatics (MICS), Gif-sur-Yvette, 91190, France.*

Abstract

We present an adaptation of the Approximate Bayesian Computation method to estimate the satisfaction probability function of a temporal logic property for Markov Population Models.

In this paper, we tackle the problem of estimating the satisfaction probability function of a temporal logic property w.r.t. a parametric Markovian model of Chemical Reaction Network. We want to assess the probability with which the trajectories generated by a parametric Markov Population Model (MPM) satisfy a logical formula over the whole parameter space. In therst step of the work, we formally define a distance between a trajectory of an MPM and a logical property. If the distance is 0, the trajectory satisfies the property. The larger the distance is, the further the trajectory is from satisfying the property. In the second step, we adapt the Approximate Bayesian Computation method using the distance defined in therst step. This adaptation yields a new algorithm, called automaton-ABC, whose output is a density function that directly leads to the estimation of the desired satisfaction probability function. We apply our methodology to several examples and models, and we compare it to state-of-the-art techniques. We show that the sequential version of our algorithm relying on ABC-SMC leads to an efficient exploration of the parameter space with respect to the formula and gives good approximations of the satisfaction probability function at a reduced computational cost.

Keywords:
Networks, Bayesian Inference, Approximate Bayesian Computation, HASL, Linear Hybrid Automata

1. Introduction

Approximate Bayesian computation (ABC) algorithms have gained popularity over the last decade and are applied for parameter inference in many modelling fields, including systems biology [1, 2, 3, 4] and cancer research [5]. They proved decisive in many cases when classical Bayesian parameter inference methods are challenging to implement. ABC allows approximating the posterior distribution of a model without evaluating the likelihood function in complex models when the computation cost is too high or even impossible. ABC methods are likelihood-free and only rely on model simulations. Simply speaking, only parameters for which simulated summary statistics are close to observed ones, relative to a distance, are preserved while the others are dismissed. These parameters are sampled from the ABC posterior, which approximates the true posterior distribution. The initial idea of our work relies on transposing this concept, initially developed for statistical inference, to temporal logic. We want to adapt the ABC approach so that only parameters which yield simulations that fulfil a given temporal logic formula are retained while others are discarded.

In this paper, we propose a new method for parametric verification of Markov Population Models (MPMs) based on the ABC algorithm. We use hybrid automata to formally measure a
a logical property. In keeping with the ABC vocabulary, this score will be called
this paper. We use this distance within the ABC framework to estimate the subset of the parameter space in which the logical property can be satisfied thanks to the obtained ABC posterior. We show that the sequential

*Corresponding author.
Email addresses:
(Mahmoud Bentriou),
(Paolo Ballarini),
(Paul-Henry Cournède)

version of ABC is perfectly suited to the use of the distance from a property, leading to an efficient exploration of the parameter space. Our method also allows the estimation of the satisfaction function of a formula, with a remarkable result linking this satisfaction function to the ABC posterior. We demonstrate the effectiveness of our method by its application to several models of Chemical Reaction Networks (CRN).

The work presented in this paper extends that introduced in [6], in many aspects, including 1) the addition of the methodology for estimating the probability satisfaction function of the considered formula from the output of modified ABC algorithms presented in [6]; 2) an extended application part which better highlights the potential of the framework through diverse case studies; 3) the addition of the proof of the soundness of the approach w.r.t. the satisfiability distances the framework relies on.

The paper is organised as follows: Section 2 introduces background material about parametric MPMs, reachability problems, the hybrid automata specification language, and the ABC framework. In Section 3, the notion of satisfiability distance for reachability problems is introduced and plugged, via hybrid automata specifications, within a novel ABC framework which, therefore, allows for identifying the subspace of parameters for which the probability of satisfying the considered reachability properrty is strictly positive. The novel ABC framework is demonstrated through several experiments in Section 4 and Section 5, while some conclusive remarks and perspectives are discussed in Section 6.

1.1. Related work

In the realm of computational systems biology [7], probabilistic modelling has become increasingly relevant [8], in particular for modelling of Chemical Reaction Network systems that are characterised by low molecular populations such as, for example, genetic regulatory networks [9]. Stochastic simulation algorithms [10] and the corresponding tools, e.g. [11], provide the modellers with practical means for observing the dynamics of a CRN model resulting from a specific parameter instance. On the other hand, stochastic model checking approaches [12] in either their *numerical* [13] or *statistical* (i.e. simulation-based) [14, 15] formulation enrich the ability to analyse the dynamics of a CRN instance through the assessment of formally specified properties expressed in temporal logic terms [16]. There exist two formulations of the stochastic model checking problem, the so-called *threshold problem*, which is concerned with establishing whether the probability that a formula Φ is satisfied by a model instance $\mathcal{M}_\theta$ fulfils a probabilistic constraint $\sim p$ (with $\sim \in \{\leq, <, >\geq\}$ and $p \in [0, 1]$) as opposed to the *estimation problem*, which is concerned with estimating the probability that $\mathcal{M}_\theta$ fulfils Φ. In recent years, the parametric verification of a probabilistic model (also referred to as parameterised model checking) has received much attention. It is concerned with combining parameter estimation techniques with stochastic model checking, i.e. with studying how the stochastic model checking problem for a property Φ is affected by the parameters θ upon which a probabilistic model (i.e. an MPM) $\mathcal{M}$ depends.

In this respect, a significant number of approaches have been proposed that tackle parametric verification w.r.t. the threshold problem perspective such as [17], in which a bounded approximation of parameter space fulfilling a CSL [18] threshold formula is efficiently determined through an adaptation of MPM *uniformisation*, or also [19, 20] and, more recently, a novel ABC-based method [21] that is based on observations even solves parameter inference and statistical parameter synthesis in one go.

Our method, on the other hand, tackles the *estimation problem* in stochastic model checking and is in line with the works of Bortolussi *et al.* [22], where the so-called smoothed model checking (Smoothed MC) method to estimate the satisfaction probability function of parametric MPM is detailed. Methods belonging to this family do not solve the parameter synthesis problem explicitly but provide good estimates of the satisfaction function that can lead to parameter synthesis, e.g. [23] which is also based on Smoothed MC.

2. Background

In this section, we briefly introduce the background material upon which the remainder of the paper relies. We recall the essential concepts of (i) Markov population models (of chemical reaction networks); (ii) temporal logic and reachability problems; (iii) Hybrid Automata Specification Language; and (iv) ABC methods (whose automaton extension is introduced in Section 3).

2.1. Markov Population Models

A Markov population model (MPM) is a form of continuous-time Markov chain (CTMC) [24, 25] 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.

Definition 2.1 (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*[1] *(i.e. $\sum_{s \in S} \pi_0(s) = 1$).*

As a consequence of the *memoryless property* of MPMs, we have that the probability of observing state changes within a given delay is driven by a negative exponential distribution. Therefore, given that the system is in state s at time t, the probability of observing a transition to state s' within time t' is $Pr((s,t) \to (s',t')) = \mathbf{P}(s,s') \cdot (1 - e^{-E(s)\cdot(t'-t)})$ 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'.

Since we are interested in assessing how the probability that an MPM exhibits a given (spatio-temporal) behaviour changes in function of some model's parameters in the remainder, we refer to the notion of parametric MPM (pMPM). In practice, we consider as potential parameters of an MPM the rates of its transitions (i.e. the entries of the infinitesimal generator matrix) yielding the following notion of pMPM.

Parametric MPM. We say an MPM is parametric, denoted $\mathcal{M}_\theta = (S, \mathbf{Q}_\theta, \pi_0)$ if $\mathbf{Q}$ depends on a set of parameters θ belonging to a parameter space Θ.

Chemical reaction networks. In the context of this paper, we consider Chemical Reaction Networks (CRNs) as a formalism for expressing population models. CRNs use *chemical equations* to capture the dynamics of population models, and are commonly used for modelling of biological systems as well as of epidemic spreading scenarios. Although CRNs are often inherently mapped on their continuous-deterministic semantics (by means of which species quantities are given as *concentrations* and their dynamics are described by a system of differential equations) here we focus on their discrete-stochastic semantics, by which species quantities are assumed to be given as individual counts (hence the name *population*) whose dynamics is described by a Markov chain. Therefore the definition of CRN we give below is inherently referred to the discrete-stochastic semantics.

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

Definition 2.2 (Chemical Reaction Network)**.** *A (parametric) chemical reaction network (pCRN) with n species and m reaction channels is a 4-tuple $\mathcal{N}_\theta = (\mathcal{X}_n, \mathcal{R}_m, \theta, \mathbf{X^0})$ 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*

The dynamics of a CRN is governed by its reactions channels. 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 (as we will see in the MPM semantics of CRNs the probability distribution of delay $t' - t$ is $\sim Exp(\eta_j(\mathbf{X}, \theta))$).

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 2.3 (pMPM 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 2.1 (CRN of infection spreading)**.** *As a first example of CRN let us consider the SIR compartmental model [26], which describes the spread of infectious disease among a constant population. The CRN for the SIR is defined as* $\mathcal{N}_{SIR} = (\{S, I, R\}, \{R_1, R_2\}, \{k_i, k_r\})$ *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 (1).*

$$R_1 : S + I \xrightarrow{k_i} 2I \qquad R_2 : I \xrightarrow{k_r} R \tag{1}$$

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.*

Paths/Trajectories of an MPM. In order to analyse the dynamics of MPMs, we need to refer to the notion of *run* of an MPM. Intuitively a run is an observation of the model evolution in time. There are two equivalent ways of representing a run either as a *path* [13], i.e. a sequence of states interleaved by (real-valued) *sojourn times*, or as a *signal* [27], i.e., a function that maps a time domain $\mathbb{T}$ over a domain $\mathbb{D}$ (which is dependent on the set of states of the MPM). In the remainder, we stick with the notion of path for defining the semantics of MITL temporal logic, knowing that such definitions can straightforwardly re-formulated w.r.t. the notion of signals. Given an MPM model $\mathcal{M}_\theta$ whose initial state is $s_0 \in S$, we denote $Path_{\mathcal{M}_\theta}$ the set of all possible paths (or trajectories) of the MPM originating in state s_0. A path/trajectory $\sigma \in Path_{\mathcal{M}_\theta}$ 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 $s_i \in S$. For $\sigma \in Path_{\mathcal{M}_\theta}$ a path, $i \in \mathbb{N}$ and $t \in \mathbb{R}_{>0}$, we denote $\sigma[i] = s_i$ the $(i+1)$-th state of σ, $\delta(\sigma, i) = t_i$ the sojourn-time of σ in s_i, 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 and including 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 (hence requiring the sojourn time of state $\sigma@t$ to be modified accordingly). 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.

Probability measure of MPM paths. An MPM inherently induces a measure of probability[2] on its paths [13]. For $s_0, s_1, \ldots, s_k$ a sequence of states of an MPM $\mathcal{M}$ 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 $\sigma \in Path_{\mathcal{M}}$ 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 $\mathcal{F}(Path_{\mathcal{M}})$ is then defined by induction w.r.t. k as follows:

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

where $t = inf(I_{k-1})$ and $t' = sup(I_{k-1})$. In essence $\Pr_{\mathcal{M}}$ states that for a CTMC/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

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

terms of vocabulary a measurable subset of trajectories of $Path_{\mathcal{M}}$ 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 2.4).

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, with X_1 in $[[1,2]]$ (for $n_1, n_2 \in \mathbb{N}$, with $n_1 \leq n_2$ we denote $[[n_1, n_2]]$ the integer valued interval bounded by n_1 and n_2) while X_2 is unbounded. On the other hand $R_2 \equiv [[0,3]] \cup [[5,8]] \times [[5,\infty[[$ 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 $[0.2, 1.41]$).

2.2. Temporal logic and reachability problems

Temporal logics [28, 29] are formal languages that state properties about the time evolution of a system and define algorithms for automatically verifying whether a system model satisfies a given property. Initially introduced to target the verification of (untimed) non-probabilistic models through either *linear-time reasoning* [30] or *branching-time reasoning* [31] they have then been extended to different domains, including that of real-time systems (e.g. [32, 27]) as well as that of probabilistic systems (e.g. [33, 12, 13, 34]). In temporal logic reasoning, the term *reachability problem* identifies the class of problems consisting in establishing whether a given model *reaches* (i.e., enters) at some point during its execution a specific region of its state-space usually associated with some state condition φ. If the considered model inherently quantifies the elapsing time (like MPMs), then one may also consider *time-bounded reachability* for which the focus is on establishing whether the target region of the state-space is entered within a time-interval $[t_1, t_2] \subset \mathbb{R}_{\geq 0}$. Temporal logic formalisms are equipped with operators for expressing reachability problems. In the context of this paper, we refer to a linear-time temporal logic such as the Metric Interval Temporal Logic (MITL [27]), which allows for stating time-bounded reachability problems for MPM models by combining state-conditions (expressed through inequalities on state variables) with time-bounded temporal operators.

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.

The truth of an MITL formula is defined w.r.t. to a path σ (i.e. a function of time) issued by an MPM model. Formally it is expressed through a so-called *satisfaction* relationship, denoted $\models$. For σ a path of the MPM $\mathcal{M}_\theta$, $\sigma \models \varphi$ reads: *path σ satisfies φ*.

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:

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

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. $\sigma \models \varphi \iff \sigma[0\rangle\varphi$. 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. 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 their corresponding regions are. In the remainder, we assume regions are characterised by MITL propositional formulae in disjunctive normal form (DNF).

Model checking MPMs. In the context of this paper, we consider the verification of MITL formulae against probabilistic models, and more specifically against MPMs. Generally speaking probabilistic model checking boils down to assessing with what probability a (temporal logic) formula φ is satisfied by a probabilistic model $\mathcal{M}_\theta$, which we refer to as the *satisfaction probability* of φ against $\mathcal{M}_\theta$, denoted $Pr(\varphi \,|\, \mathcal{M}_\theta)$. Intuitively the *satisfaction probability* $Pr(\varphi \,|\, \mathcal{M}_\theta)$ is given by the probability measure of the set of paths that satisfy φ, as stated in the following definition.

Definition 2.4 (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 (2) induced by $\mathcal{M}_\theta$ over $Path_{\mathcal{M}_\theta}$.

$Pr(\varphi|\mathcal{M}_\theta)$ may be assessed either exactly through *numerical* model checkers [35, 36] (although these are affected by the state-space explosion problem, hence they are limited to models of reasonable size) or being estimated through *statistical* model checkers [37, 38, 39, 40] (through which the estimates of $Pr(\varphi|\mathcal{M}_\theta)$ are obtained by statistical inference based on trajectory samples of arbitrary size).

Parametric model checking of MPMs. If model checking of MPMs is concerned with evaluating what is the probability that a property φ is satisfied by a model $\mathcal{M}_\theta$, that corresponds to parameters value $\theta \in \Theta \subseteq \mathbb{R}^d$, in many realistic applications, it is crucial to be able to investigate how the satisfaction probability of φ changes with the model's parameters. Indeed, parametric model checking [22] is concerned with evaluating the so-called *satisfaction function* of a formula φ w.r.t. the domain of a model's parameters.

Definition 2.5 (Satisfaction probability function)**.** *Let $(\mathcal{M}_\theta)_{\theta\in\Theta}$ be a parametric MPM. The function:*

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

is called the satisfaction probability function.

The satisfaction probability function expresses how the satisfaction probability of φ changes with the parameters.

An automata-based framework for estimating f_φ. Our goal is to introduce a framework aimed at statistically estimating f_φ. Since we opt for an (adaptation of the) Approximation Bayesian Computation approach, the estimation of f_φ relies on the capability to measure (*on-the-fly*) how *distant* a model's trace σ (issued by stochastic simulation) is from satisfying an MITL formula φ, which we denote $d(\sigma, \varphi)$ (see Definition 3.1). Although algorithms for assessing $d(\sigma, \varphi)$ could be, in principle, conceived based on the MITL syntax directly (as it is the case with, e.g. methods for assessing the *robustness* of a formula against real-valued signals [41, 27]), in our framework we opted to rely on an automata-based formalism: given a (non-nested) *reachability formula* φ, we define a *linear hybrid automaton* $\mathcal{A}_\varphi$ that when *synchronised* with a trace σ is capable of measuring $d(\sigma, \varphi)$. The benefit of such an approach is that, based on the operational semantics of the Hybrid Automata Specification Language (HASL) [42], we can straightforwardly assess $d(\sigma, \varphi)$ simply by reproducing the simulation of the *synchronised product process* $\mathcal{M}_\theta \times \mathcal{A}_\varphi$. Furthermore, by relying on linear hybrid automata, our framework is suitable to be integrated within the COSMOS statistical model checking platform [43] (although the experiments presented in this paper have been obtained through a standalone prototype developed on purpose for this paper).

2.3. Hybrid Automata Specification Language

The *distance automata* we introduce as part of our framework for estimating f_θ rely on the HASL [42] formalism. HASL statistical model checking is based on the idea of employing a linear hybrid automaton (LHA) $\mathcal{A}$ as a *monitor* that filters trajectories issued by

a discrete-state stochastic process[3] $\mathcal{M}_\theta$, while collecting relevant statistics in the process. To this aim, HASL defines a procedure for simulating the (synchronised) product process $\mathcal{M}_\theta \times \mathcal{A}$ whose trajectories are determined by two kinds of events: *synchronised events*, that correspond to the occurrence of a transition in $\mathcal{M}_\theta$ that drags along a synchronising one in $\mathcal{A}$, and *autonomous events*, that happen autonomously in $\mathcal{A}$ (i.e. without a change of state in $\mathcal{M}_\theta$) as soon as a certain (e.g. state or time) condition is satisfied in the current state of $\mathcal{M}_\theta \times \mathcal{A}$. In the following, we give a short overview of the HASL formalism, referring the reader to [42] for an exhaustive treatment.

2.3.1. Hybrid Automata for monitoring of trajectories

An LHA for HASL is an automaton that has access to certain elements of model $\mathcal{M}_\theta$, namely the events and the state-variable of $\mathcal{M}_\theta$. Formally, an LHA is defined as an n-tuple:

$$\mathcal{A} = \langle E, L, \Lambda, Init, Final, X, flow, \rightarrow \rangle$$

where: E is a finite alphabet of events (the events of $\mathcal{M}_\theta$ that drive the synchronisation, in the context of this paper E inherently corresponds with $\mathcal{R}_m$ the set of reactions of a CRN); L is a finite set of locations; $\Lambda : L \rightarrow Prop$, a location invariant function ($Prop$ being the set of atomic propositions built on top of variables X); $Init$ is a subset of L called the initial locations; $Final$ is a subset of L called the final locations; $X = (x_1, ...x_n)$ an n-tuple of data variables; $flow : L \mapsto Ind^n$ is a function that gives, for each location, the rate at which variable x_i evolves (where the rate for variable x_i is given by an *indicator function* that depends on the state of the model $\mathcal{M}_\theta$); $\rightarrow \subseteq L \times \Big((2^E \times \mathsf{Const}) \uplus (\{\sharp\} \times \mathsf{IConst})\Big) \times \mathsf{Up} \times L$, a set of edges, where for an edge $(l, \gamma, E', U, l') \in \rightarrow$, denoted $l \xrightarrow{E',\gamma,U} l'$ in the remainder, we have that E' is either a subset of events $E' \subseteq E$ or $E' = \sharp$ where $\sharp$ denotes the autonomous event (i.e. an event triggered by the LHA hence not corresponding to any event of E) with Const the set of constraints, whose elements are Boolean combinations of inequalities of the form $\sum_{1 \leq i \leq n} \alpha_i x_i + c \prec 0$ where α_i and c are constants, $\prec \in \{=, <, >, \leq, \geq\}$, whereas IConst is the set of left-closed constraints, and $U = (u_1, ..., u_n) \in Up$ is an n-tuple of functions characterising how each LHA variable x_k is going to be updated on traversal of the edge[4].

Selection of a model's trajectory with an automaton $\mathcal{A}$ is achieved through *synchronisation* of $\mathcal{M}_\theta$ with $\mathcal{A}$ (see below), i.e. by letting $\mathcal{A}$ synchronise its transitions with the transitions of the trajectory σ being sampled. To this aim, an LHA admits two kinds of transitions: *synchronising* transitions (associated with a subset $E \subseteq \Sigma$ of event names, with ALL denoting Σ), which may be traversed when an event (in E) is observed on σ (for example a reaction occurs), and *autonomous* transitions (denoted by $\sharp$) which are traversed autonomously (and have priority over synchronised transitions), on given conditions, typically to update relevant statistics or terminate (accept) the analysis of σ. Since automata-based formalisms are at least as expressive as temporal logic based on classical temporal modalities (see [44]), in the remainder we denote $\mathcal{A}_\varphi$ the HASL automaton equivalent to an MITL formula φ (i.e. $\mathcal{A}_\varphi$ accepts a trajectory σ of an MPM model $\mathcal{M}_\theta$ if and only if $\sigma \models \varphi$).

Determinism constraints. An LHA for HASL must ensure that the synchronisation of an arbitrary trajectory $\sigma \in Path_{\mathcal{M}_\theta}$ is finite and unique, hence ruling out possible Zeno-like divergences as well as non-determinism. To this aim an LHA for HASL must comply with the following constraints: **c1 (initial determinism):** at most one initial location $l \in I$ can have its invariant $\Lambda(l)$ verified ($\forall l \neq l' \in I$, $\Lambda(l) \wedge \Lambda(l') \Leftrightarrow \mathsf{false}$); **c2 (determinism on events):** synchronisation of $\mathcal{A}$ w.r.t. an arbitrary event e must be deterministic ($\forall E_1, E_2 \subseteq E : E_1 \cap E_2 \neq \emptyset, \forall l, l', l'' \in L$, if $l'' \xrightarrow{E_1,\gamma,U} l$ and $l'' \xrightarrow{E_2,\gamma',U'} l'$ then either $\Lambda(l) \wedge \Lambda(l') \Leftrightarrow \mathsf{false}$ or $\gamma \wedge \gamma' \Leftrightarrow \mathsf{false}$.); **c3 (Determinism on $\sharp$:)**: at most one autonomous transition can ever be enabled ($\forall l, l', l'' \in L$, if $l'' \xrightarrow{\sharp,\gamma,U} l$ and $l'' \xrightarrow{\sharp,\gamma',U'} l'$ then either $\Lambda(l) \wedge \Lambda(l') \Leftrightarrow \mathsf{false}$ or $\gamma \wedge \gamma' \Leftrightarrow \mathsf{false}$); **c4 (absence of $\sharp$-labelled loops:)** $\mathcal{A}$ cannot contain a loop of autonomous transitions (i.e. 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 $E_i \neq \sharp$). The LHA automata we define as part of our framework (Section 3.2) fulfil the above described constraints.

2.3.2. Synchronised product process $\mathcal{M}_\theta \times \mathcal{A}$

For the sake of brevity here, we provide only an intuitive description of the product process $\mathcal{M}_\theta \times \mathcal{A}$ whose

[3] The class of stochastic processes targeted by HASL includes but is not limited to MPMs as no restrictions are assumed on the nature of the probability distribution associated with events. Therefore, *memorylessness* is not a necessary condition.

[4] Each function u_k ($1 \leq k \leq n$) of an edge update $U = (u_1, ..., u_n) \in Up$ 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 of the MPM.

operational semantics is formally defined in [42]. The states of $\mathcal{M}_\theta \times \mathcal{A}$ are triples $(s, l, \nu) \in (S \times L \times Val) \uplus \{\bot\}$ where s is the current state of the $\mathcal{M}_\theta$, l the current location of the $\mathcal{A}$, $\nu : X \to \mathbb{R}$ the current valuation of the variables of $\mathcal{A}$ and $\bot$ denotes the *rejecting* state, i.e., the state entered when synchronisation fails, hence when a trajectory is rejected. Notice that a configuration of $\mathcal{M}_\theta \times \mathcal{A}$ has the following form $((s, l, \nu), \tau, sched')$, where (s, l, ν) is the current state of $\mathcal{M}_\theta \times \mathcal{A}$, $\tau \in \mathbb{R}^+$ is the current time, and $sched'$ is the schedule of the enabled events of $\mathcal{M}_\theta \times \mathcal{A}$. The synchronisation starts from the initial state (s, l, ν), where s is a possible initial state of $\mathcal{M}_\theta$ (i.e. $\pi_0(s) > 0$), l is an initial location of the LHA (i.e. $l \in Init$) and the LHA variables are all set to zero (i.e. $\nu = 0$)[5]. From the initial state, the synchronisation process evolves through transitions where each transition corresponds to traversal of either a synchronised or an autonomous edge of the LHA[6]. Furthermore, if an autonomous and a synchronised edge are concurrently enabled, the autonomous transition is taken first. Assuming (s, l, ν) is the current state of $\mathcal{M}_\theta \times \mathcal{A}$, let us describe how the synchronisation evolves. If in the current location l of the LHA there exists an enabled autonomous edge $l \xrightarrow{\sharp, \gamma, U} l'$, then that edge will be traversed leading to a new state (s, l', ν') where the state of $\mathcal{M}_\theta$ (i.e. s) is unchanged whereas the new location l' and the new variables' valuation ν' might differ from l, respectively ν, as a consequence of the edge traversal. On the other hand, if an event e of $\mathcal{M}_\theta$ (corresponding to transition $s \xrightarrow{e} s'$) occurs in state (s, l, ν), either an enabled synchronous edge $l \xrightarrow{E', \gamma, U} l'$ (with $e \in E'$) exists leading to new state (s', l', ν') of process $\mathcal{M}_\theta \times \mathcal{A}$ (from which synchronisation will continue) or the synchronisation halts hence the trace is rejected (formally this is achieved with the system entering the rejecting state $\bot$).

Example 2.2 (LHA for the SIR model)**.** *Figure 1 depicts a two-locations LHA (center) for time-bounded measures (over the time interval* $[0, T]$*, T being a constant) of the SIR model (left). The LHA has locations* $L = \{l_0, l_1\}$ *(with* l_0 *the initial location,* l_1 *the final location), variables* $X = \{t, x_1, n_2\}$ *with t a clock variable, x_1 a real-valued variable (for measuring the average population of infected individuals I) and n_2 an integer variable (for counting the number of occurrences of the R_2 reaction). While in l_0, t changes with flow 1 (i.e. as t is a clock) while x_1 flow is given by X_I (i.e. the population of I, hence x_1 measures the integral of the population I while in l_0). The synchronisation with a path σ (i.e. with its I-projection denoted σ_I) of the SIR model is as follows. At time $t = 0$, the LHA starts in l_0 and stays there up to $t = T$. As soon as $t = T$, the synchronisation with σ ends as the* autonomous *transition $l_0 \xrightarrow{\sharp, t=T, \{x_1/=T\}} l_1$ becomes enabled hence is fired (by definition, autonomous transitions have priority over* synchronised *transitions in HASL). As long as $t < T$ the LHA is in l_0 where it synchronises with the occurrences of the SIR reactions: on the occurrence of R_2, transition $l_0 \xrightarrow{\{R_2\}, t<T, \{n_2++\}} l_0$ (which is* synchronised *on event set $\{R_2\}$) is fired hence increasing the counter n_2, whereas on the occurrence of any other reaction (i.e. R_1 in this case), transition $l_0 \xrightarrow{ALL \setminus \{R_2\}, t<T, \emptyset} l_0$ (which is* synchronised *on event set $ALL \setminus \{R_2\}$, where ALL stands for the reactions set $\mathcal{R}_m$ of the considered CRN) fires without updating any variable. Finally, on ending the synchronisation with σ (when $t = T$), variable x_1 is updated to x_1/T which corresponds to the average population of I observed over the time interval* $[0, T]$*. Figure 1 (bottom) includes an example of a path σ of the SIR model (σ_I being the I-projection of σ) and the corresponding synchronisation with the LHA (denoted $\sigma_I \times \mathcal{A}$) assuming $T = 4$ as time-bound of the synchronisation. Notice that the states of $\sigma_I \times \mathcal{A}$ are denoted $(s_I, l, [t, x_1, n_2])$ with s_I the I-projection of the current state of $\mathcal{M}_\theta$, l the current location of $\mathcal{A}$ and $[t, x_1, n_2] \in \mathbb{R}^3$ the values of the variables of $\mathcal{A}$.*

2.4. The ABC method

Given a parametric model $\mathcal{M}_\theta$ and an MITL (reachability) 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 $\theta \in \Theta$. To this aim, we rely upon the class of *Bayesian inference* methods known as Approximate Bayesian Computation (ABC). Generally speaking, *statistical inference* is interested in inferring properties of an underlying probability distribution based on some data observed through an experiment y_{exp}. Bayesian inference methods rely on the Bayesian interpretation of probability. Starting from some *prior distribution* $\pi(.)$, which expresses an *initial belief* on the distribution over the parameters domain Θ, Bayesian methods estimate the *posterior distribution*

[5] Notice that because of the "initial determinism" of LHA, there can be at most one initial state for the product process.

[6] Notice that because of the determinism constraints of the LHA edges (conditions **c2** and **c3**), at most only one autonomous or synchronised edge can ever be enabled in any location of the LHA.

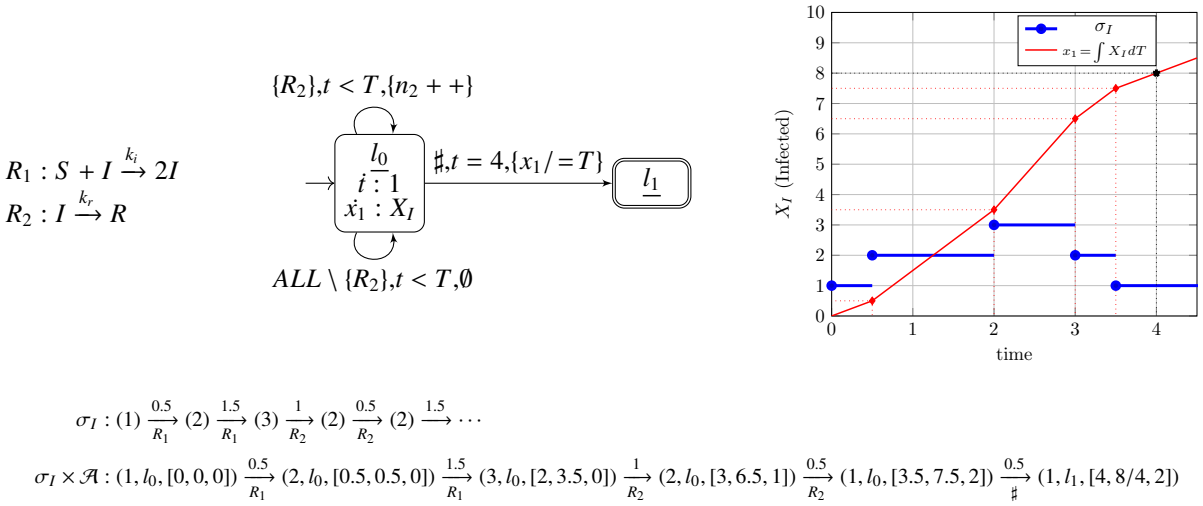

Figure 1: An LHA (center) for assessing the average of the population of infected individuals (i.e. X_I) over time-interval $[0, T]$ for the SIR model (left). Plots (right) show the projection (blue plot) σ_I (w.r.t. species I) of a possible trajectory σ issued by the SIR model (blue plot) and the corresponding evolution (red plot) of the LHA variable x_1 (that stores the integral of the population of I i.e. $\dot{x_1} = X_I$) resulting when σ_I is synchronised with the LHA. Notice that on ending the measurement (i.e. at *time* $= T$), the LHA set $x_1 = x_1/T$, which is indeed the average population of I over $[0, T]$.

$\pi(\theta|y_{exp})$ over Θ based on the observed data y_{exp}. Formally, the posterior distribution is defined by:

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

where $p(y|\theta)$ denotes the *likelihood function*, that is, the function that measures how probable y is to be observed given the model's parameters θ.

An inherent drawback of Bayesian statistics is in that, by definition, the posterior distribution relies on the accessibility to the likelihood function $p(y_{exp}|\theta)$, which, particularly for complex models, may be too expensive to compute or even intractable. ABC algorithms have been introduced to tackle this issue, i.e. as a *likelihood-free* alternative to classical Bayesian methods (we refer to [45, 46] for exhaustive surveys of ABC or rejection-sampling methods).

Simple ABC method. The basic idea behind the ABC method is to obtain an approximate estimate, denoted $\pi_{ABC,\epsilon}$, of the posterior distribution $\pi(\theta|y_{exp})$, in the form of a number of parameter samples θ_i drawn from the ABC posterior distribution. This is achieved through an iterative procedure (Algorithm 1) by which, at each iteration, we start off by drawing a parameter vector θ' from a prior, i.e. $\theta' \sim \pi(.)$, we simulate the model $\mathcal{M}'_\theta$, and we accept parameter θ if the corresponding simulation y' is "close enough" to the observations according to a chosen threshold ϵ. Notice that in Algorithm 1, $\eta : \mathcal{Y} \to \mathcal{S} \subset \mathbb{R}^{k_1}$ represents summary statistics[7] computed on the observations y_{exp} 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. The accepted parameters θ_i together with the corresponding traces y'_i give samples (θ_i, y_i) drawn from the joint distribution: $\pi_{ABC,\epsilon}(\theta, y|y_{exp}) \propto \mathbb{1}_{A_{\epsilon,y_{exp}}}(y)p(y|\theta)\pi(\theta)$ where $A_{\epsilon,y_{exp}} = \{y'/\mathbb{1}_{\rho(\eta(y'),\eta(y_{exp}))\leq\epsilon}\}$ (with $\mathbb{1}_{\rho(\eta(y'),\eta(y_{exp}))\leq\epsilon}$ denoting the indicator function representing the set of traces whose distance from y_{exp} 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_{exp}, so

Algorithm 1 Simple ABC

Require: N: number of particles, y_{exp} 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 $\rho(\eta(y'), \eta(y_{exp})) \leq \epsilon$
    $\theta_i \leftarrow \theta'$
    $y_i \leftarrow y'$
  end for

[7] The choice of summary statistics is a crucial point in ABC (see for example [47]).

the better the approximation.

The ABC-SMC 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 [48]. It is an SMC based approach [49] 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 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 [49]) until they match the next level, tighter, 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_{exp}, $(\epsilon_i)_{1\leq i\leq M}$, 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 algorithm simple ABC
  $\omega_j \leftarrow \frac{1}{N}$
  for $i = 2 : M$ **do**
    for $j = 1 : N$ **do**
      repeat
        Take $\theta_j^{'}$ from $(\theta_j^{(i-1)})_{1\leq j\leq N}$ with prob. $(\omega_j)_{1\leq j\leq N}$
        $\theta_j^{(i)} \sim K(.|\theta_j^{'})$
        $y' \sim p(.|\theta_j^{(i)})$
      until $\rho(\eta(y'), \eta(y_{exp})) \leq \epsilon_i$
      $\omega_j \leftarrow \frac{\pi\left(\theta_j^{(i)}\right)}{\sum_{j'=1}^{N} \omega_{j'}^{(i-1)} K(\theta_j^{(i)}|\theta_{j'}^{(i-1)})}$
    end for
    Normalize $(\omega_j)_j$
  end for

A basic point about ABC algorithms is that, by definition, they all rely on a notion of *distance* (between the simulations y' and observations y_{exp}). Based on this characteristic, in Section 3, we introduce an HASL-based adaptation of the ABC scheme to estimate the satisfaction probability function f_θ. In essence, we plug a distance automaton in the ABC procedure and use it as 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. With the HASL-based extension of ABC, we are going to estimate the ABC-posterior using a zero tolerance, i.e. with $\epsilon = 0$.

2.5. Kernel density estimation

ABC methods only deliver samples $(\theta_i)_i$ drawn from the ABC posterior distribution. However, our goal is to approximate a continuous (density) function (related to f_θ). Therefore, we resort to kernel density estimation (KDE) [50], [51].

The goal of KDE is to derive an approximation $\widehat{\pi}$ of an unknown probability density function π given a finite number of samples ($(\theta_i)_i$ in our case) of a random variable. The approximation $\widehat{\pi}$ is obtained by the sum of the application of some *kernel function* K to the samples. A kernel function is a continuous function. Its application to a sample captures the contribution, in terms of probability mass, brought by the sample to the density π to be estimated (i.e. essentially, a kernel is a manner to weight data samples).

Definition 2.6 (Kernel function)**.** *A function $K : \mathbb{R} \to \mathbb{R}_{\geq 0}$ is a kernel function if:*

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

Based on kernel functions, one can define a kernel density estimator [50].

Definition 2.7 (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:*

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

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

With this estimator, each sample contributes to the probability mass over the whole set Θ, with the idea that the further we are from the observation, the lower the probability is.

Several choices for the kernel function and the calibration method of the bandwidth are available [50], [51]. In our work, we select the bandwidth by minimising the Least Squares Cross-Validation criterion, and

use Gaussian and Beta kernels [52] with the multivariate estimator of [53]. In Section 3.4 we introduce an adaptation of the KDE approach to our goal, that is, given the samples θ_i issued by the Automaton-ABC method, we adapt KDE for approximating the satisfaction probability function f_θ.

3. Methods

We introduce an ABC-based methodology to approximate the *satisfaction function* (Definition 2.5) of a time-bounded reachability MITL formula φ w.r.t. a parametric MPM model $\mathcal{M}_\theta$. Our methodology consists of four aspects: *i)* the formalisation of the notion of distance of a model's path from a *reachability region* (i.e. a time-bounded MPM region related to a time-bounded reachability problem), *ii)* the introduction of the corresponding HASL specifications to measure such distance, *iii)* the definition of a novel ABC method adapted to reachability problems, i.e. an ABC algorithm in which the convergence is driven by the distance from a reachability region, and, finally, *iv)* the derivation of the normalisation constant through which the satisfaction function is obtained from the *ABC posterior density*.

3.1. Satisfiability distances for reachability problems

ABC algorithms (Algorithm 1 and Algorithm 2) allow one to explore a model's parameter space through an iterative procedure whose *convergence* is based on a notion of *distance* (of a model's path from some observations). In order to adapt them to reachability problems, we introduce the notion of *satisfiability distance* (named simply *distance* in the remainder) of a model's path σ w.r.t. a reachability property φ. The distance of σ from φ should be defined so to comply with the following guidelines: *i)* it should express how "far" σ is from satisfying φ (i.e. the farther σ is from satisfying φ the largest the distance), *ii)* it should evaluate to zero whenever $\sigma \models \varphi$ and *iii)* it should favour the convergence of the ABC algorithm. We point out that the step-function nature of the paths of MPMs[8] induces some peculiarities on the characterisation of such a distance, particularly w.r.t. the convergence of the ABC scheme (see details below). Furthermore, we stress that the notion of satisfiability distance we need to introduce is strictly related to that of *time-bounded satisfiability regions* (Section 2.1), i.e. the part of the space-time domain the satisfiability of the considered formula depends upon.

[8] i.e an MPM path consists of a sequence of jumps

Based on MITL formulae, we distinguish three kinds of reachability problems: *eventual reachability* problems ($\mathbf{F}^I\mu$) are concerned with paths entering a region μ within a given time interval $I=[t_1,t_2]$, *global reachability* problems ($\mathbf{G}^I\mu$) are concerned with paths never leaving a region μ within a time interval I, and *conditional reachability* problems ($\mu_1\mathbf{U}^I\mu_2$) are concerned with paths entering a region μ_2 within a time interval I and without ever leaving a region μ_1 beforehand. $\mathbf{F}^{[t_1,t_2]}\mu$ and $\mathbf{G}^{[t_1,t_2]}\mu$ formulae induce a *simple* time-bounded satisfiability region which we denote $S_\mu^{[t_1,t_2]}$ in the remainder, i.e. $S_\mu^{[t_1,t_2]} = S_\mu \times [t_1,t_2]$, where S_μ is the set of states where μ is satisfied, whereas $\mu_1\mathbf{U}^{[t_1,t_2]}\mu_2$ formulae induce a *compound* satisfiability region given by $S_{\mu_1}^{[0,t_1]} \cup S_{\mu_2}^{[t_1,t_2]} \cup S_{\mu_1}^{[t_1,t_2]}$

Based on the notion of satisfiability region associated with a propositional formula μ, we introduce the notion of *satisfiability distance* of a path from a satisfiability region. Such a satisfiability region is associated with different types of temporal formulae built on top of μ.

Definition 3.1 (Satisfiability distance)**.** *Given a path $\sigma \in Path_{\mathcal{M}}$ of a n-dimensional MPM $\mathcal{M}$ with state space $S \subseteq \mathbb{N}^n$, a closed time-bounding interval $[t_1,t_2] \subset \mathbb{R}_{\geq 0}$, an elementary propositional formula μ, we define the distance $d(\sigma,\varphi)$ from the satisfiability region for the following kinds of temporal formulae built on top of μ:*

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

$$d(\sigma,\mathbf{F}^{[t_1,t_2]}\mu) = \begin{cases} d_e((\sigma@t_l(t_2),t_l(t_2)),S_\mu^{[t_1,t_2]}) \\ \qquad\qquad \text{if } t_l(t_2) < t_1 \wedge \sigma@t_l(t_2) \not\models \mu \\ min\Big(d_e((\sigma@t_l(t_1),t_l(t_1)),S_\mu^{[t_1,t_2]}), \\ \qquad \min\limits_{t\in[t_1,t_2]}(d_e((\sigma@t,t),S_\mu^{[t_1,t_2]}))\Big) \\ \qquad\qquad otherwise \end{cases} \tag{3}$$

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 $d_e((s,t),S_1 \times T_1) = \min\limits_{t'\in T_1, s'\in S_1} \sqrt{(t-t')^2 + \sum_{i=1}^n (s[i]-s'[i])^2}$ 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) $\varphi \equiv \mathbf{G}^{[t_1,t_2]}\mu$

$$d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu) = \int_{t_1}^{t_2} d_e(\sigma@t, S_\mu)dt \tag{4}$$

where $d_e(s, S_1) = \min_{s' \in S_1} \sqrt{\sum_{i=1}^{n}(s[i] - s'[i])^2}$ *denotes the euclidean distance of a point* $s \in S$ *from the closest state of a state space subset* $S_1 \subseteq S \subseteq \mathbb{N}^n$. $d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu)$ *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) $\varphi \equiv \mu_1 \mathbf{U}^{[t_1,t_2]}\mu_2$

$$\begin{aligned} d(\sigma, \mu_1 \mathbf{U}^{[t_1,t_2]}\mu_2) =& d(\sigma, \mathbf{G}^{[0,t_1[}\mu_1) + d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu_2) + \\ & d(\sigma, \mathbf{G}^{[t_1,t_{min}]}(\mu_1 \vee \mu_2)) \end{aligned} \tag{5}$$

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.

Remark 3.1 (Null distance)**.** *In agreement with the semantics of the temporal modalities, the distance (3) of an MPM path* σ *from a* $\mathbf{F}^{[t_1,t_2]}\mu$*, formula is 0 if and only if* σ *has at least one point traversing region* $S_\mu^{[t_1,t_2]}$ *(e.g. Figure 2a), while the distance (4) from a* $\mathbf{G}^{[t_1,t_2]}\mu$ *formula is 0 if and only if within* $t \in [t_1, t_2]$ *all points* $\sigma@t$ *fall in* $S_\mu^{[t_1,t_2]}$ *(Figure 2b), and finally the distance (5) from a* $\mu_1 \mathbf{U}^{[t_1,t_2]}\mu_2$ *formula is 0 if and only if there exists* $t' \in [t_1, t_2]$ *such that* $\sigma@t'$ *is in* $S_{\mu_2}^{[t_1,t_2]}$ *while is consistently in* $S_{\mu_1}^{[0,t']}$ *beforehand (Figure 2c).*

Remark 3.2 (Positive distance)**.** *Conversely, distance (3) of* σ *from* $\mathbf{F}^{[t_1,t_2]}\mu$ *yields a positive value, given by minimal Euclidian distance between* σ *and* $S_\mu^{[t_1,t_2]}$*, whenever* σ *contains jumps in* $[t_1, t_2]$ *(Figure 2d), whereas distance (4) of* σ *from* $\mathbf{G}^{[t_1,t_2]}\mu$ *yields a positive value, corresponding with the volume of the hyper-rectangle delimited by the segments of* σ *that (within* $[t_1, t_2]$*) lie outside* $S_\mu^{[t_1,t_2]}$ *(Figure 2e). Finally, for an Until formula* $\mu_1 \mathbf{U}^{[t_1,t_2]}\mu_2$*, we observe that distance (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* 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* μ_2 *within* $[t_1, t_2]$ *and* $d(\sigma, \mathbf{G}^{[t_1,t']}(\mu_1 \vee \mu_2))$ *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 region: if that is not the case (i.e. if* σ *within* $[t_1, t_2]$ *has points in the complementary region* $\neg(\mu_1 \vee \mu_2)$*) then (5) yields a positive value given which accounts for the sum of the minimal distances of each such point from either regions* μ_1 *or* μ_2 *(see plot in Figure 2f).*

Remark 3.3 (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), (4) and (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), is set to the Euclidian distance between region* $S_\mu^{[t_1,t_2]}$ *and the point entered 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 3a). 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 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 3c).*

Below we prove that the satisfiability distances for **F**, **G** and **U** formulae are sound w.r.t. the second criteria listed above, that is: if a path σ satisfies a formula φ then the distance is $d(\sigma, \varphi) = 0$. We start off by proving such property for **F** and **G** formulae concerning *elementary regions*, we then extend the proof to **F** and **G** concerning *non-elementary regions* and finally, based on results proved for **F** and **G**, we extend the prove also to **U** formulae.

Lemma 3.1 (Soundness elementary **F**)**.** *For* $\sigma \in Path_{\mathcal{M}}$ *a path of an MPM* $\mathcal{M}$ *and* μ *an MITL proposition cor-*

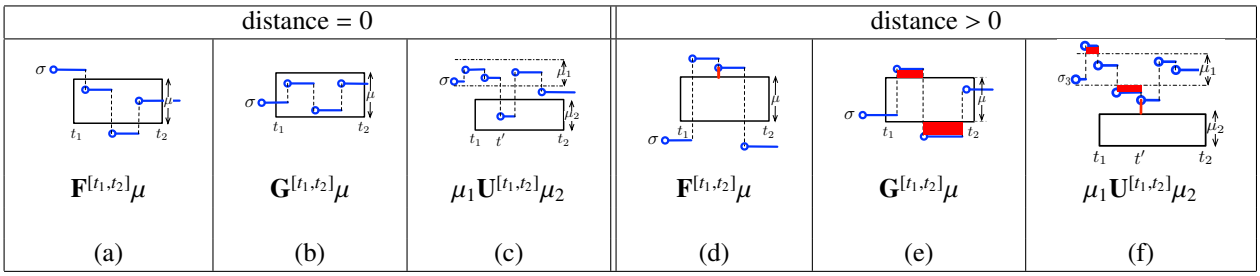

Figure 2: Examples of paths with zero-distance (left) and positive distance (right) from an F, a G and a U region (positive distances are depicted in red).

responding to an elementary region then

$$\sigma \models \mathbf{F}^{[t_1,t_2]}\mu \iff d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu) = 0$$

Proof. $\Rightarrow$ Let us assume that $\sigma \models \mathbf{F}^{[t_1,t_2]}\mu$. which means $(\sigma, 0) \models \mathbf{F}^{[t_1,t_2]}\mu$. Then from MITL semantics $\exists t^* \in [t_1, t_2], (\sigma, t^*) \models \mu$ hence $\sigma@t^* \in S_\mu$. By definition, $d_e((t^*, \sigma@t^*, [t_1, t_2] \times S_\mu) = \min_{t' \in [t_1,t_2], s' \in S_\mu} \sqrt{(t^* - t')^2 + \sum_{i=1}^{n} (\sigma@t'[i] - s'[i])^2}$.
Since $t^* \in [t_1, t_2]$ and $\sigma@t^* \in S_\mu$ then trivially $d_e((t^*, \sigma@t^*, [t_1, t_2] \times S_\mu) = 0$ hence $d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu) = 0$.

$\Leftarrow$ Let us assume that $d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu) = 0$. It means $t_{last} \geq t_1$ and $\min_{t \in [t_1,t_2]} d_e((t, \sigma@t), [t_1, t_2] \times S_\mu) = 0$ (the first case of distance computation cannot produce 0). Then $\exists t^* \in [t_1, t_2] : d_e((t^*, \sigma@t'), [t_1, t_2] \times S_\mu) = 0$. As d_e is a distance, $(t^*, \sigma@t^*) \in [t_1, t_2] \times S_\mu$, i.e $(\sigma, t^*) \models \mu$. From MITL semantics, $(\sigma, 0) \models \mathbf{F}^{[t_1,t_2]}\mu$. ■

Lemma 3.2 (Soundness elementary **G**). *For $\sigma \in Path_\mathcal{M}$ a path of an MPM $\mathcal{M}$ and μ an MITL proposition corresponding to an elementary region then*

$$\sigma \models \mathbf{G}^{[t_1,t_2]}\mu \iff d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu) = 0$$

Proof. $\Rightarrow$ Let us assume that $\sigma \models \mathbf{G}^{[t_1,t_2]}\mu$ which means $(\sigma, 0) \models \mathbf{G}^{[t_1,t_2]}\mu$. By definition, $\mathbf{G}^{[t_1,t_2]}\mu = \neg\mathbf{F}^{[t_1,t_2]}\neg\mu$. From MITL semantics it follows $\forall t \in [t_1, t_2]$, $(\sigma, t) \not\models \neg\mu$ i.e. $\forall t \in [t_1, t_2]$, $\sigma@t \models \mu$. $\forall s' \in S_\mu$, $d_e(s', S_\mu) = 0$. So $\forall t \in [t_1, t_2], d_e(\sigma@t, S_\mu) = 0$ hence $\int_{t_1}^{t_2} d_e(\sigma@t, S_\mu)dt = 0$. In conclusion, $d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu) = 0$.

$\Leftarrow$ Suppose that $d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu) = 0$. Let us prove $\sigma \models \mathbf{G}^{[t_1,t_2]}\mu$.

We have : $\int_{t_1}^{t_2} d_e(\sigma@t, S_\mu)dt = 0$. As $t \to d_e(\sigma@t, S_\mu)$ is a non-negative continuous function and its integral equals 0, then this function is the null function over $[t_1, t_2]$. So, $\forall t \in [t_1, t_2], d_e(\sigma@t, S_\mu) = 0$. This means $\forall t \in [t_1, t_2], \sigma@t \in S_\mu$. In other words, $\sigma \models \mathbf{G}^{[t_1,t_2]}\mu$. ■

Having proved the soundness of the satisfiability distance for **F** and **G** formulae with elementary regions, we straightforwardly extend this result to formulae that involve non-elementary propositions.

Proposition 3.1 (Soundness non-elementary **F** and **G**). *For $\sigma \in Path_\mathcal{M}$ a path of an MPM $\mathcal{M}$ and $\mu \equiv \bigvee \mu_i$ an MITL propositional formula in DNF where each μ_i corresponds to an elementary region then*

$$\sigma \models \mathbf{F}^{[t_1,t_2]}\mu \iff d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu) = 0$$
$$\sigma \models \mathbf{G}^{[t_1,t_2]}\mu \iff d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu) = 0$$

Proof.

$$\begin{aligned} d(\sigma, \mathbf{F}^{[t_1,t_2]} \bigvee \mu_i) = 0 &\Leftrightarrow \min_i d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu_i) = 0 \\ &\Leftrightarrow \exists i \mid d(\sigma, \mathbf{F}^{[t_1,t_2]}\mu_i) = 0 \\ &\Leftrightarrow \exists i \mid \sigma \models \mathbf{F}^{[t_1,t_2]}\mu_i \\ &\Leftrightarrow \sigma \models \mathbf{F}^{[t_1,t_2]} \bigvee \mu_i \end{aligned}$$

$$\begin{aligned} d(\sigma, \mathbf{G}^{[t_1,t_2]} \bigvee \mu_i) = 0 &\Leftrightarrow \min_i d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu_i) = 0 \\ &\Leftrightarrow \exists i \mid d(\sigma, \mathbf{G}^{[t_1,t_2]}\mu_i) = 0 \\ &\Leftrightarrow \exists i \mid \sigma \models \mathbf{G}^{[t_1,t_2]}\mu_i \\ &\Leftrightarrow \sigma \models \mathbf{G}^{[t_1,t_2]} \bigvee \mu_i \end{aligned}$$

For Until formula, result follows from $\sigma \models \mu_1\mathbf{U}^{[t_1,t_2]}\mu_2 \Leftrightarrow \sigma \models \mathbf{G}^{[0,t_1]}\mu_1 \wedge \mathbf{G}^{[t_1,t_{min}]}\mu_1 \wedge \mathbf{F}^{[t_{min},t_2]}\mu_2$ and the proof of the distances for eventual and global regions. ■

Lemma 3.3 (Soundness **U**). *For $\sigma \in Path_{\mathcal{M}}$ a path of an MPM $\mathcal{M}$ and μ_1 and μ_2 two MITL propositions corresponding to elementary regions then*

$$\sigma \models \mu_1 \mathbf{U}^{[t_1,t_2]} \mu_2 \iff d(\sigma, \mu_1 \mathbf{U}^{[t_1,t_2]} \mu_2) = 0$$

Proof. The proof of the above equivalence is a direct consequence of the following decomposition of a time-bounded **U** formulae as a combination of time-bounded **F** and **G** formulae:

$$\sigma \models \mu_1 \mathbf{U}^{[t_1,t_2]} \mu_2 \iff \sigma \models \mathbf{G}^{[0,t_1[} \mu_1 \wedge \exists t' \in [t_1, t_2] : \sigma \models \mathbf{F}^{[t',t']} \mu_2 \wedge \mathbf{G}^{[t_1,t']} (\mu_1 \vee \mu_2)$$

which follows straightforwardly from the semantics of the **U** operator.
⇒ Assuming $\sigma \models \mu_1 \mathbf{U}^{[t_1,t_2]} \mu_2$ then $\sigma \models \mathbf{G}^{[0,t_1[} \mu_1$ and therefore, from Lemma 3.2, $d(\sigma, \mathbf{G}^{[0,t_1[} \mu_1) = 0$. Furthermore if $\sigma \models \mu_1 \mathbf{U}^{[t_1,t_2]} \mu_2$ then also $\exists t' \in [t_1, t_2]$ such that 1) $\sigma \models \mathbf{F}^{[t',t']} \mu_2$, hence, from Lemma 3.1, also $d(\sigma, \mathbf{F}^{[t_1,t_2]} \mu_2) = 0$ and 2) since $d_e(\sigma @ t', \mu_2) = 0$ then t_{min} in (5) is $t_{min} = t'$ and since $\mathbf{G}^{[t_1,t']}(\mu_1 \vee \mu_2)$ then, from Proposition 3.1, also $d(\sigma, \mathbf{G}^{[t_1,t_{min}]}(\mu_1 \vee \mu_2)) = 0$. Thus the three addends in (5) are all zero which proves the implication ⇒.
⇐ similar approach by reversing ⇒. ∎

3.2. HASL specifications for satisfiability distances

Based on the HASL formalism, we discuss the problem of defining hybrid automata specifications to measure the satisfiability distances introduced in the previous section. For the sake of simplicity, the automata presented here refer to temporal formulae built on top of a generic mono-dimensional (elementary) proposition $\mu \equiv x_1 \leq x_O \leq x_2$, where x_O denotes the population of an observable quantity O of an MPM model $\mathcal{M}_\theta$ and $x_1 < x_2 \in \mathbb{N}$. Distance automata for formulae based on n-dimensional regions are simply adaptations of those in Figure 3 and Figure 4.

Distance automaton $\mathcal{A}_F$. Automaton $\mathcal{A}_F$ (Figure 3) is designed to measure the distance (3) of a path σ (of an MPM model $\mathcal{M}_\theta$) 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 (Section 2.3.1) 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 $d := 0$[9] 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), 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.

[9] 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 $d = 0$).

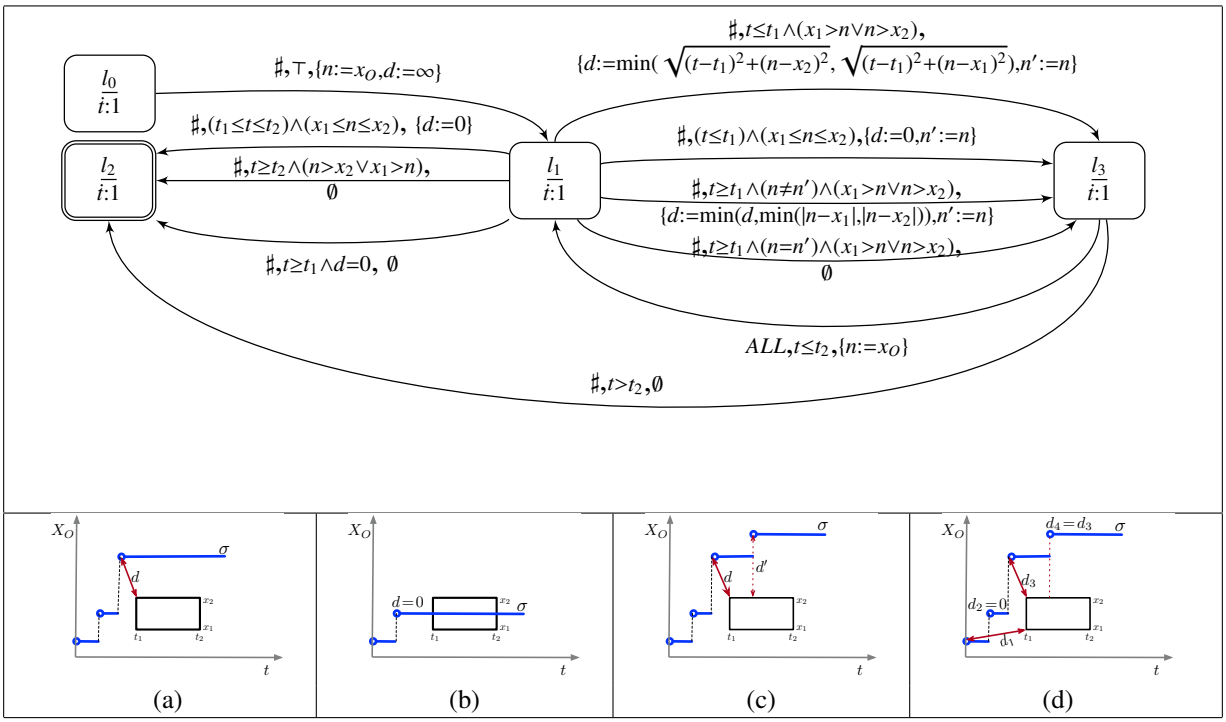

Figure 3: Automaton $\mathcal{A}_F$ (top) for measuring the distance of a path σ for an eventual property concerning observed species X_O and examples (bottom) of measured distance d: positive distance (a), null distance (b), selection of the minimum distance (c) in case of presence of jumps in $[t_1, t_2]$ and evolution of the computed distance d along a path (d).

Proposition 3.2. *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:*

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

where $\mathcal{A}_F(\sigma).d$ denotes the value stored in variable d of automata $\mathcal{A}_F$ when $\mathcal{A}_F$ has synchronised with σ.

Proof. See Appendix A. ■

Distance automaton $\mathcal{A}_G$. Automaton $\mathcal{A}_G$ (Figure 4) 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 (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 ($l_0 \rightarrow l_1$), 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 ($l_1 \rightarrow l_3$ top arc). Otherwise, d is the distance of $\sigma@t$ from $[x_1, x_2]$ otherwise ($l_1 \rightarrow l_3$ midway arc). Indeed, if, for example, the next jump of σ happens at $t > t_2$, then the final distance is given by $d \cdot (t_2 - t_1)$ ($l_1 \rightarrow l_2$ bottom arc). For events occurring at $t \in [t_1, t_2]$, if $\sigma@t \notin [x_1, x_2]$ (sequence $l_1 \rightarrow l_4 \rightarrow l_1$), the distance is incremented 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 $l_1 \rightarrow l_3 \rightarrow l_1$).

Distance automaton $\mathcal{A}_{G \wedge F}$. Automaton $\mathcal{A}_{G \wedge F}$ refers to measuring the distance of paths from a sequence of regions consisting of a G region (related to an observed quantity O) temporally followed by an F region (related to an observed quantity O'). In practical terms, such an automaton is associated with the MITL formula $\mathbf{G}^{[t_1,t_2]}(x_1 \leq x_O \leq x_2) \wedge \mathbf{F}^{[t_3,t_4]}(x_3 \leq x_{O'} \leq x_4)$, for which we assume $t_2 \leq t_3$ (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'. This automaton is

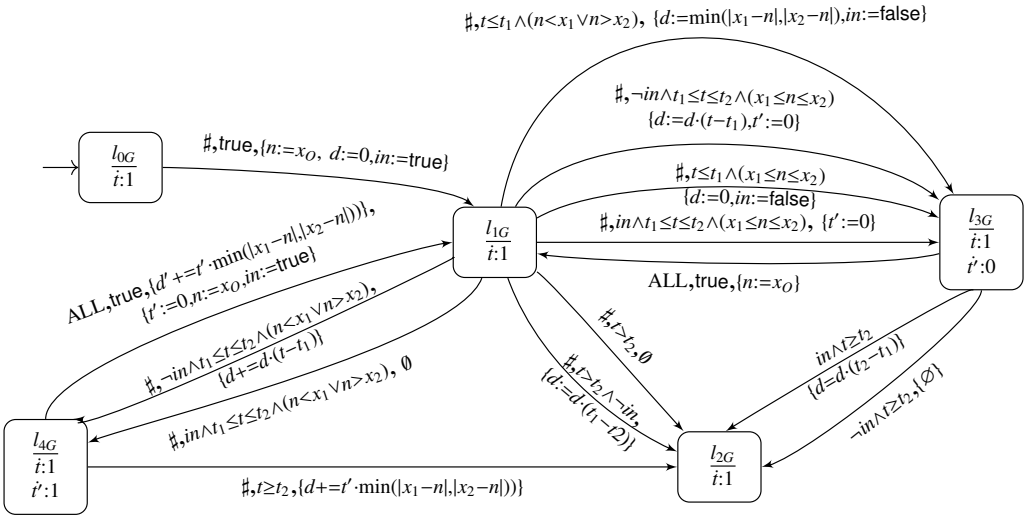

Figure 4: Automaton $\mathcal{A}_G$ for global property.

a concatenation of the automata $\mathcal{A}_G$ and $\mathcal{A}_F$, which is illustrated in Appendix B.

3.3. Automaton-ABC: ABC with satisfiability distance

We now introduce the adaptation of the ABC algorithms discussed in Section 2.4, which we name *automaton-ABC* in the remainder, to estimate the satisfaction probability function of a reachability formula φ by a parametric MPM $(\mathcal{M}_\theta)_\theta$. A preliminary version of the method was presented in [6], where ABC was used to show regions where parameters were susceptible to satisfying the probability. However, the method presented in [6] was not equipped with necessary means for directly estimating the satisfaction probability function. We point out that with the Automaton-ABC, the estimation of the ABC posterior distribution ($\pi_{\varphi-ABC}$) is no longer computed as a limit approximation (i.e. $\lim_{\epsilon\to 0}\hat{\pi}_{ABC,\epsilon}(.|y_{exp})$), as with classical ABC, but rather as an estimation of the exact distribution, since paths are accepted exclusively if their distance to the satisfiability region is zero.

Simple ABC with satisfiability distance. With Algorithm 3, we propose a modified version of the simple ABC Algorithm 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 distance automaton $\mathcal{A}_\varphi$ corresponding to a reachability formula φ. It works as follows: at each iteration, a parameter θ' is drawn from the prior $\pi(.)$, a path σ' is sampled from the MPM $\mathcal{M}_{\theta'}$ and the distance $d(\sigma',\varphi)$ is computed by synchronisation of σ' with automaton $\mathcal{A}_\varphi$; θ' is accepted if the distance from φ is $d(\sigma',\varphi)=0$ (i.e. if $\sigma'\models\varphi$ by Proposition 3.1).

Algorithm 3 Automaton-ABC with $\mathcal{A}_\varphi$ automaton

Require: $(\mathcal{M})_{\theta\in\Theta}$ a pMPM, $\pi(.)$ prior, $\mathcal{A}_\varphi$ 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

Proposition 3.3 links Algorithm 3 with the satisfaction probability function (Definition 2.5).

Proposition 3.3. *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 $(\theta_i)_{1\leq i\leq N}$ sampled by the Algorithm 3 are drawn from a density function $\pi_{\varphi-ABC}$:*

$$\pi_{\varphi-ABC}(\theta_i) = Pr(\varphi|\mathcal{M}_{\theta_i})\frac{\pi(\theta_i)}{K}$$

where $Pr(\varphi|\mathcal{M}_\theta)$ is the probability satisfaction function of Definition 2.5 and $K\in\mathbb{R}_{\geq 0}$ is a positive constant.

Proof. Let $C^\varphi_{\mathcal{M}_\theta} = \{\sigma\in Path_{\mathcal{M}_\theta}/\sigma\models\varphi\}$. Then $Pr_{\mathcal{M}_\theta}(C^\varphi_{\mathcal{M}_\theta}) = Pr(\varphi|\mathcal{M}_\theta)$. ABC is a reformulation of the

accept-reject algorithm. Thus, the samples $(\theta_i, \sigma_i)_{1\leq i\leq N}$ from Algorithm 3 are drawn from a density $\pi_{\varphi-ABC}$:

$$\pi_{\varphi-ABC}(\theta_i, \sigma_i) \propto \underbrace{\mathbb{1}_{d(.,\varphi)=0}(\sigma_i)}_{\sigma_i \models \varphi} \underbrace{p_{\mathcal{M}_{\theta_i}}(\sigma_i)}_{\sigma_i \sim \mathcal{M}_{\theta_i}} \underbrace{\pi(\theta_i)}_{\text{prior}}$$

where $p_{\mathcal{M}_{\theta_i}}$ is the density related to the MPM $\mathcal{M}_{\theta_i}$ with regard to a measure $\overline{\mu}$.

As $\sigma \in C^{\varphi}_{\mathcal{M}_\theta} \Leftrightarrow d(\sigma, \varphi) = 0$, $\mathbb{1}_{d(.,\varphi)=0}(\sigma_i) = \mathbb{1}_{C^{\varphi}_{\mathcal{M}_\theta}}(\sigma_i)$. One can obtain the marginal distribution of θ by integration over the whole set of paths $Path_{\mathcal{M}_\theta}$:

$$\begin{aligned}\pi_{\varphi-ABC}(\theta) &\propto \int_{\sigma\in Path_{\mathcal{M}_\theta}} \mathbb{1}_{C^{\varphi}_{\mathcal{M}_\theta}}(\sigma) p_{\mathcal{M}_\theta}(\sigma)\pi(\theta)d\overline{\mu} \\ &\propto \pi(\theta) \int_{\sigma\in C^{\varphi}_{\mathcal{M}_\theta}} p_{\mathcal{M}_\theta}(\sigma)d\overline{\mu} \\ &\propto Pr_{\mathcal{M}_\theta}(C^{\varphi}_{\mathcal{M}_\theta})\pi(\theta)\end{aligned}$$

As $Pr_{\mathcal{M}_\theta}(C^{\varphi}_{\mathcal{M}_\theta}) = Pr(\varphi|\mathcal{M}_\theta)$, we can conlude that $\pi_{\varphi-ABC}(\theta) = Pr(\varphi|\mathcal{M}_\theta)\frac{\pi(\theta)}{K}$. ■

Algorithm 4 Automaton-ABC Sequential Monte Carlo with $\mathcal{A}_\varphi$ automaton

Require: $(\mathcal{M})_{\theta\in\Theta}$ a pMPM,, $\pi(.)$ prior, $\mathcal{A}_\varphi$ distance automaton for MITL formula φ, N : number of particles, $\alpha \in (0, 1)$, K kernel distribution

Ensure: $(\omega_j)_{1\leq j\leq N}$, $(\theta_j)_{1\leq j\leq N}$ weighted samples drawn from $\pi_{\varphi-ABC}$

$(\theta_j^{(1)})_{1\leq j\leq N} \sim \pi(.)$
$\forall j \in 1, \ldots, N, d_j \sim (\mathcal{M}_{\theta_j^{(1)}} \times \mathcal{A}_\varphi)$
$\epsilon \leftarrow quantile(\alpha, d_j)$
$(\omega_j)^{(1)}_{1\leq j\leq N} \leftarrow \frac{1}{N}$
$i \leftarrow 2$
while $\epsilon > 0$ **do**
  for $j = 1 : N$ **do**
    repeat
      Take θ'_j from $(\theta_j^{(i-1)})_{1\leq j\leq N}$ with prob. $(\omega_j^{(i-1)})_{1\leq j\leq N}$
      $\theta_j^{(i)} \sim K(.|\theta'_j)$
      $d_j \sim (\mathcal{M}_{\theta_j^{(i)}} \times \mathcal{A}_\varphi)$
    until $d_j \leq \epsilon$
    $\omega_j \leftarrow \frac{\pi\left(\theta_j^{(i)}\right)}{\sum_{j'=1}^{N} \omega_{j'}^{(i-1)} K(\theta_j^{(i)}|\theta_{j'}^{(i-1)})}$
  end for
  Normalise $(\omega_j^{(i)})_j$
  $\epsilon \leftarrow quantile(\alpha, (d_j)_{1\leq j\leq N})$
  $i \leftarrow i + 1$
end while

This result transforms the regression of a smooth function into the regression of a probability density function. First, each parameter sampled from the $\pi_{\varphi-ABC}$ gives information, because it produced a simulation that verifies φ. Also, as $\pi_{\varphi-ABC}$ is a probability density function, the relative position of each parameter to the other sampled parameters gives much information about the satisfaction probability function. The denser in sampled parameters a subset of parameters space, the higher the satisfaction probability function over the subset.

ABC-SMC with satisfiability distance. Following the same approach, we formulate a comparable version of the ABC-SMC Algorithm 4 with distance automaton, resulting in a reduced *runtime* (w.r.t. Algorithm 3) yet complying with Proposition 3.3.

The functioning/motivation behind Algorithm 4 may be summarised by the following remarks. If with Algorithm 3, we do not exploit the real-value of the measured distance (i.e. we only accept/reject paths depending on whether their distance is zero, i.e. if they satisfy φ.), with Algorithm 4, we take advantage of the distance value to rank paths and accept the parameters whose corresponding paths are closer (i.e. better ranked) to the satisfiability regions than others (even if they do not necessarily satisfy φ). This can lead to a faster convergence of the algorithm corresponding to a faster exploration of the parameter space.

Algorithm 4 has the same inputs as Algorithm 3, plus a kernel distribution K and a hyper-parameter $\alpha \in]0, 1[$ representing how fast the tolerance ϵ decreases along with the iterations.

It works as follows. Initially, N parameters/particles $(\theta_j^{(1)})_{1\leq j\leq N}$ are drawn from the prior $\pi(.)$, and the first tolerance level ϵ to reach equals the α-quantile of the distances d_j, resulting from the synchronised simulations $\mathcal{M}_{\theta_j^{(1)}} \times \mathcal{A}_\varphi$. Then, at each iteration i, each parameter $\theta_j^{(i)}$ ($j \in \{1, \ldots, N\}$) is moved by a kernel distribution K, and the synchronised simulation $\mathcal{M}_{\theta_j^{(i)}} \times \mathcal{A}_\varphi$ is performed. This procedure is done until the resulting distance d_j is below the current tolerance level ϵ, which means the parameter $\theta_j^{(i)}$ is kept. After doing so for the N parameters, we compute a new tolerance level ϵ that equals the α-quantile of the distances d_j. These iterations are repeated until the last tolerance level $\epsilon = 0$ is reached. The introduction of several steps with positive decreasing tolerances leads to an efficient exploration of the parameter space driven by $\mathcal{A}_\varphi$.

3.4. Estimation of the satisfaction probability function

Based on the samples $(\theta^{(i)})_{1\leq i\leq M}$ of the Algorithm 4, we can estimate the satisfaction probability function

thanks to Proposition 3.3. This procedure is twofold: estimation of the ABC posterior density and estimation of the constant K.

Estimation of the $\varphi - ABC$ posterior distribution. We estimate our ABC posterior based on the samples $(\theta_i)_{1\leq i\leq N}$ with kernel density estimation (Section 2.5). Two kernels are used: Gaussian and beta [52]. 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 [50].

Estimation of K. K is estimated by a single-point estimation of $Pr(\varphi|\mathcal{M}_{\theta^*}))$ and $\pi_{\varphi-ABC}(\theta^*)$. θ^* should be chosen wisely: verifying φ should not be rare, and θ^* should be in a region where $\pi_{\varphi-ABC}$ can be well approximated (a region of high posterior probability). Then, $Pr(\varphi|\mathcal{M}_{\theta^*})$ can be estimated with statistical model checkers. One can choose several $\theta*$, estimate the constants, and compute the mean to get a more stable kernel density estimation.

4. Application

We applied the automaton-ABC method to tackle the estimation of satisfaction probability function on three models of biological systems: the enzymatic reaction network (Michaelis-Menten kinetics), a model of viral infection and the SIR chemical reaction network.

4.1. Enzymatic reaction system

4.1.1. Model

We consider the model of Enzymatic Reaction system (Michaelis-Mentens kinetics [54]) described by Equation(6), in which a *substrate* species S is converted into a *product* P through the mediation of an *enzyme* E. The dynamics depend on the kinetic rates that induce a parameter vector $\theta = \{k_1, k_2, k_3\}$. We thus consider the underlying parametric MPM $(\mathcal{M})_{\theta\in[0,100]^3}$. The initial state is $(E_0, S_0, ES_0, P_0) = (100, 100, 0, 0)$.

$$\begin{aligned} R_1 &: E + S \xrightarrow{k_1} ES \\ R_2 &: ES \xrightarrow{k_2} E + S \\ R_3 &: ES \xrightarrow{k_3} E + P \end{aligned} \tag{6}$$

Figure 5 shows two (4-dimensional) paths sampled from the MPM model $\mathcal{M}_\theta$ of the enzymatic reaction system with parameters $\theta = (1, 1, 1)$ (top) and $\theta = (0.1, 1, 0.1)$ (bottom). The dynamics of the ER system (Figure 5) is such that the totality of the substrate (initially $S_0 = 100$) is converted into the product at speed dependent on parameters θ. With $\theta = (1, 1, 1)$, the totality of S is converted before $time = 5$, whereas with a tenfold speed reduction in the formation of the ES complex and synthesis of P (i.e. $\theta = (0.1, 1, 0.1)$), we have that only about 30% of S has been converted at $time = 5$.

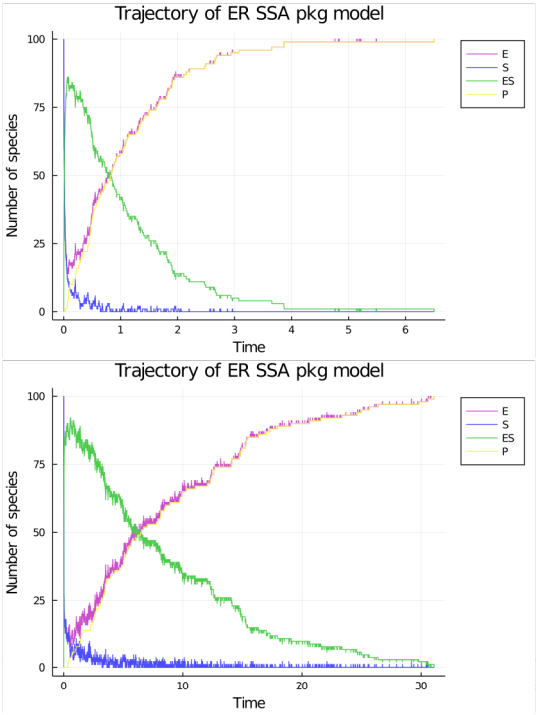

Figure 5: Paths of the ER system with $\theta_{\text{top}} = (1, 1, 1)$, $\theta_{\text{bottom}} = (0.1, 1, 0.1)$.

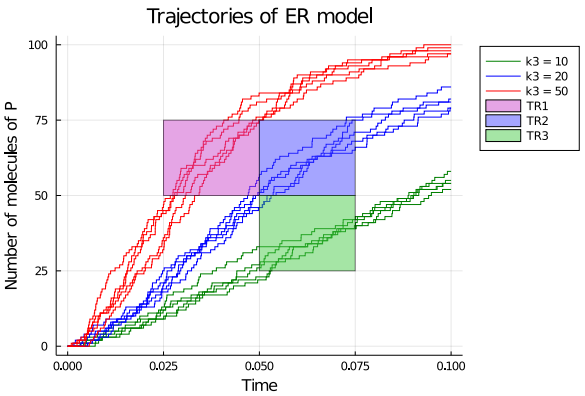

Figure 6: Relationship between paths of ER system for species P corresponding to parameter configurations with fixed $k_1 = k_2 = 1$ and different values of $k_3 \in \{10, 20, 50\}$ and three different regions TR1, TR2 and TR3.

4.1.2. Preliminary tests of distance automata

Before actually applying the distance automata of Section 3.2 to the ABC framework of Section 3.3, we have executed several experiments aimed at testing whether the distance measured by automata $\mathcal{A}_F$, $\mathcal{A}_G$ and $\mathcal{A}_{G\wedge F}$, in isolation, that is, not plugged within ABC algorithms, give reasonable readings. The results are shown in Figure 7. To this aim, we have used the statistical model checker Cosmos [43], i.e. we have developed the MPM model of the ER system (in terms of a generalised stochastic Petri net model, the input modelling formalism for Cosmos) as well the distance automata with Cosmos.

More specifically, for these tests, we have considered different configurations of the ER model and observed whether the distance (from specific regions) measured through the automata was in line with the dynamics exhibited by several paths sampled from each configuration. For example, Figure 6 shows batches of paths of the ER model for species P, corresponding to the parameter sets $\theta_1 : (1, 1, 50)$, $\theta_2 = (1, 1, 20)$ and $\theta_3 = (1, 1, 10)$ (i.e. only k_3 varies). It appears that paths for $\mathcal{M}_{\theta_1}$ (red) are likely to traverse TR_1, those for $\mathcal{M}_{\theta_2}$ (blue) to traverse $TR2$ and those for $\mathcal{M}_{\theta_3}$ (green) to traverse $TR3$. We have therefore considered the following time-bounded reachability formulae, each of which is associated to a corresponding time-bounded region:

- $\varphi_1 : \mathbf{F}^{[0.025,0.05]}(50 \leq P \leq 75)$ associated with region named $TR1$,
- $\varphi_2 : \mathbf{F}^{[0.05,0.075]}(50 \leq P \leq 75)$ associated with region named $TR2$,
- $\varphi_3 : \mathbf{F}^{[0.05,0.075]}(25 \leq P \leq 50)$ associated with region named $TR3$,
- $\varphi_4 : \mathbf{F}^{[8.0,10.0]}(5 \leq P \leq 15)$ associated with region named $TR4$,
- $\varphi_5 : \mathbf{G}^{[0.0,0.8]}(50 \leq E \leq 100)$ associated with region named $TR5$,
- $\varphi_6 : \mathbf{G}^{[0.0,0.8]}(50 \leq E \leq 100) \wedge \mathbf{F}^{[0.8,0.9]}(30 \leq P \leq 100)$ associated with region named $TR6$.

We point out that formulae $\varphi_1, \varphi_2, \varphi_3$ (which correspond to region $TR1$, $TR2$, $TR3$ of Figure 6) are examples of formulae for which there is a fairly large subset of values for $k_3 \in [0, 100]$ (with $k_1 = k_2 = 1$) which yields a positive satisfaction probability. On the other hand, $\varphi_4, \varphi_5, \varphi_6$ are formulae where only a small subset of $k_1, k_2 \in [0, 100]^2$ (with $k_3 = 1$) yields a positive satisfaction probability.

Such intuition is confirmed by the plots showed in the first row of Figure 7, which depicts the average value of the distance of paths from time regions $TR1$, $TR2$ and $TR3$ measured with Cosmos, as a function of k_3, using specific instances of $\mathcal{A}_F$, i.e. $\mathcal{A}_{\varphi_1}$, $\mathcal{A}_{\varphi_2}$ and $\mathcal{A}_{\varphi_3}$. We observe that, for example, the measured distance from region $TR1$ monotonically decreases as k_3 increases and cancels for $k_3 \geq 30$, while the distance from region $TR3$ is zero when $10 \leq k_3 \leq 15$, whereas it grows as k_3 increases.

4.1.3. Satisfaction probability function estimation

We apply automaton-ABC Algorithm 4 to the ER parametric MPM defined over $[0, 100]^3$. $TR1$, $TR2$ and $TR3$ corresponds to one-dimensional experiments: only k_3 varies over $[0, 100]$ ($k_1 = k_2 = 1.0$ are fixed), a uniform prior $\pi(.) \sim \mathcal{U}(0, 100)$ is set. $TR4$, $TR5$ and $TR6$ corresponds to two-dimensional experiments: k_1 and k_2 varies over $[0, 100]$ ($k_3 = 1.0$ is fixed), a uniform prior $\pi(.) \sim \mathcal{U}(0, 100)$ over each parameter is set.

Figure 8 illustrates the evaluation of the posterior distribution $\pi_{\varphi-ABC}$ and the estimation of the satisfaction function probability w.r.t. parameter k_3 (1D experiments) obtained by application of the automaton-ABC method (Algorithm 3 and 4) to a few examples of $\mathbf{F}$ (eventual) formulae.

The estimated function of $TR1$ exhibits a rather uniform profile, with a 95% credibility interval that φ_1 is satisfied for $k_3 \in [20, 100]$ (approximately), which is in agreement with the average distance measure (Figure 7, first row). When the average distance is zero, the estimated probability by both Prism model checking and automaton-ABC is one. The estimated functions for $TR2$ and $TR3$, instead, result in narrower 95% credibility intervals with $k_3 \in [15, 50]$ ($TR2$) resp. $k_3 \in [5, 25]$ ($TR3$), again in line with average measured distance (Figure 7, first row).

Figure 9 depicts the results of the 2D experiments on the ER system with examples of $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{G} \wedge \mathbf{F}$ formulae. The triangular profile of the joint posterior in experiments $TR4$ and $TR5$ (computed with $k_3 = 1$ and $\pi_{k_1}(.), \pi_{k_2}(.) \sim U(0, 100)$) indicates that only very low values of k_1 ($k_1 \leq 0.015$ for $TR4$, $k_1 \leq 1$ for $TR5$) combined with rather high values of k_2 (i.e. $k_2 \in [40, 100]$ for $TR4$, $k_2 \in [50, 100]$ for $TR5$) result in paths entering $TR4$, resp. never leaving $TR5$, which means that the algorithm managed to catch the correlation between the parameters. This is intuitively correct in both cases.

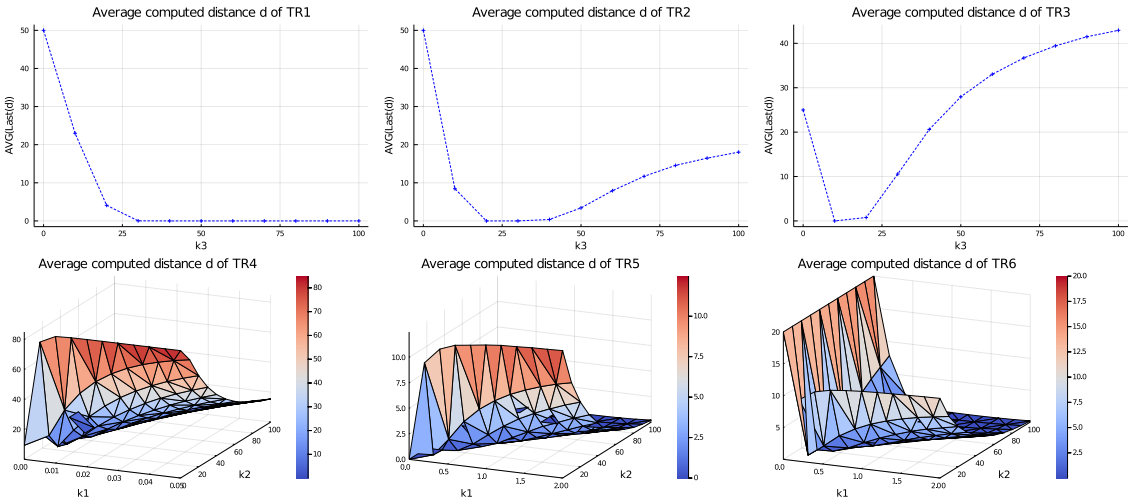

Figure 7: Average distances of the automata based on the six formulae φ_i, $i \in \{1, \ldots, 6\}$, computed by Cosmos with approximation of 0.1 and 99% level of confidence. First row: $\mathcal{A}_{\varphi_1}, \mathcal{A}_{\varphi_2} and \mathcal{A}_{\varphi_3}$. Second row: $\mathcal{A}_{\varphi_4}, \mathcal{A}_{\varphi_5} and \mathcal{A}_{\varphi_6}$.

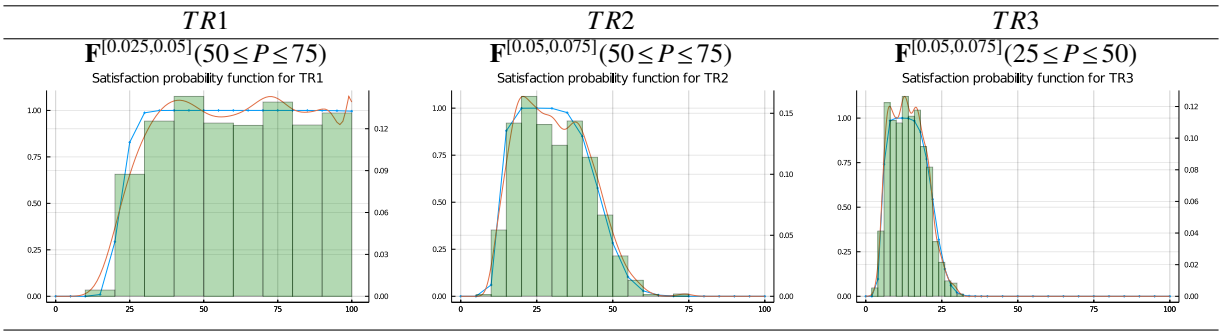

Figure 8: Weighted histogram of automaton-ABC posterior of each experiment with 1000 particles. In each experiment, $k_1 = k_2 = 1$ and $\pi_{k_3}(.) \sim U(0, 100)$. In blue: the satisfaction probability function estimated with Prism model checker on a selection of points using the numerical engine of Prism. In red : the satisfaction function estimated through kernel density estimation method based on automaton ABC samples.

In fact, $TR4$ corresponds to a very low synthesis of P, which is not compatible with fast creation of the ES complex (i.e. only very small k_1 are not ruled out), and even the compensation effect obtained by fast decomplexation (i.e. large k_2) will not suffice for paths to stay in $TR4$.

One can notice that the estimated satisfaction probability function of region $TR4$ (most-left bottom picture of Figure 9) has a lot of probability mass around $(0, 0)$. At first glance, one could conclude in a problem of bias of the kernel density estimator since the satisfaction probability function estimated by Statistical Model Checking (second row, first column picture) does not show the same shape in this area. However, if we run Statistical MC with a refined grid around zero (Figure 10), one can surprisingly notice that the probability values are high around $(0, 0)$. This behaviour was not expected, and automaton-ABC algorithm allowed us to discover this small area of high satisfaction probability that was not caught by Statistical Model Checking with the original grid of 20 points per axis.

Similarly to experiment $TR4$, $TR5$ limits the speed of the initial decrease of E (with initially $E_0 = 100$) to 50 within $t \leq 0.8$, which again is compatible only with slow ES complexation and cannot be compensated by fast decomplexation.

The $TR6$ experiment caught an even more important correlation between the two parameters. In fact, the posterior for $TR6$ is contained in the one obtained for $TR5$, which is expected because if a path verifies $TR6$, then it also verifies $TR5$.

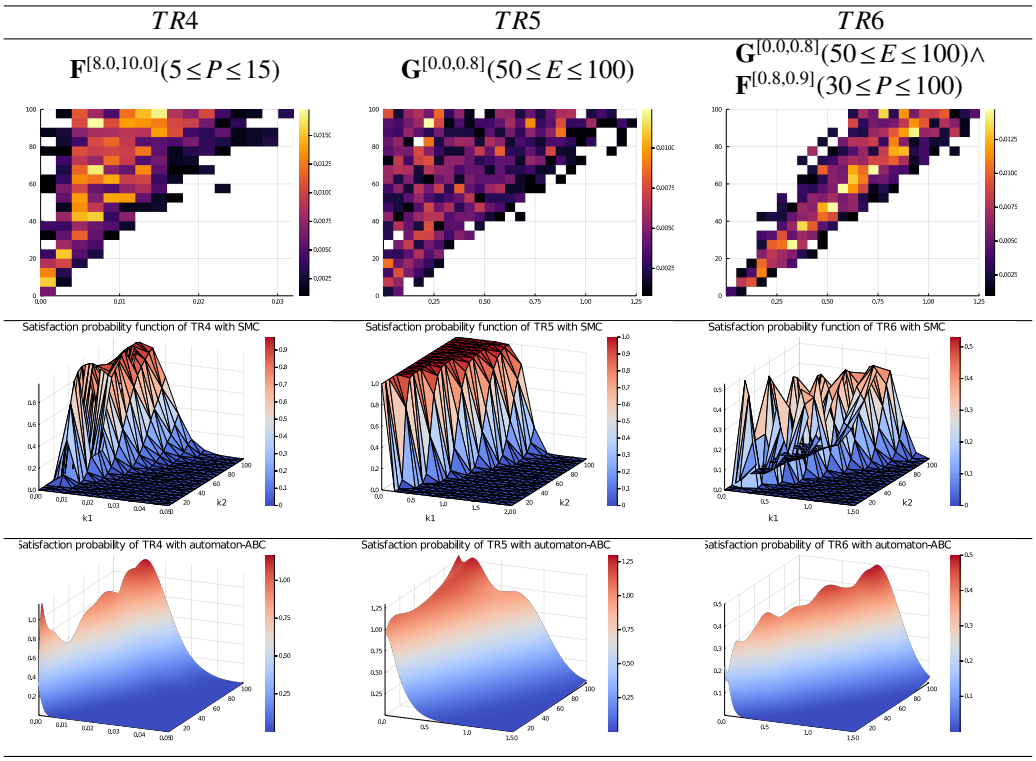

Figure 9: Results of 2D experiments of ER system with 1000 particles ($\pi_{k_1}(.), \pi_{k_2}(.) \sim U(0, 100)$, $k_3 = 1$).
Top: the 2D weighted histogram of the automaton-ABC posterior.
Middle: estimation of the satisfaction probability function with Prism by Statistical model checking (99% confidence interval with approximation 0.01).
Bottom: kernel density estimation of the satisfaction probability function.

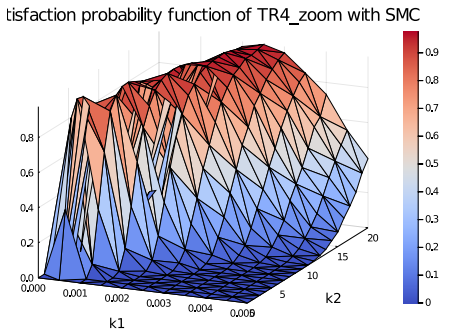

Figure 10: Statistical Model Checking of the ER model with $TR4$ over $[0.0, 0.0005] \times [0.0, 20.0]$. 10 points for the first axis and 20 points for the second axis.

4.1.4. Remarks

Results have been obtained by running Algorithm 4 with sample size $N = 1000$. There are no notable differences in performance between Algorithm 3 and 4 for 1D experiments on $TR1$, $TR2$, $TR3$ because of the large size of the resulting distributions. However, simple automaton-ABC Algorithm 3 is not worth considering for $TR4$ and $TR5$. Given the large size of the support for the considered priors ($[0, 100]\times[0, 100]$), we remark that the probability of sampling (a pair of) parameters in the resulting ABC posterior distribution is about $\frac{\frac{90\times 0.03}{2}}{100*100} \approx 10^{-4}$. This leads to 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 from the obtained distribution could produce paths that do not satisfy φ. By adding several transitional steps with the sequential version (Algorithm 4), the problem becomes treatable. The results for $TR4$ required about $2 \cdot 10^5$ simulations of the model, which is the highest number of simulations in all experiments.

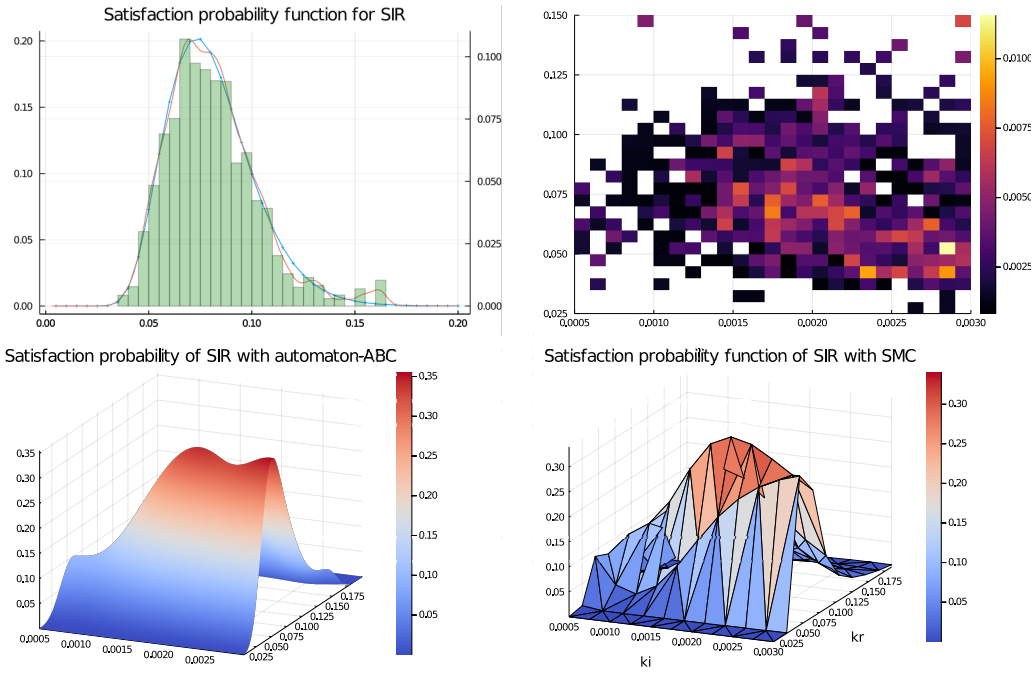

Figure 11: Results for the SIR model with $\varphi = \mathbf{G}^{[0,100]}(I > 0) \wedge \mathbf{F}^{[100,120]}(I = 0)$. On the top left figure: automaton-ABC posterior weighted histogram with 1000 particles for the 1D experiment; in blue: the true satisfaction probability function computed with Prism model checker using the numerical engine of Prism over 40 points; in red: the estimated satisfaction function with kernel density estimation method. The three other figures correspond to the 2D experiment. On the top right figure: the 2D histogram of the automaton-ABC posterior. On the bottom left figure: the kernel density estimation of the satisfaction probability function. On the bottom right figure: the estimation of the satisfaction probability function by Prism model checker (numerical engine).

4.2. SIR

We consider the classical SIR compartmental model [26] introduced in Example 2.1. We tested our algorithm on a single formula $\varphi = \mathbf{G}^{[0,100]}(I > 0) \wedge \mathbf{F}^{[100,120]}(I = 0)$ with one 1D experiment and one 2D experiment. This formula means the considered epidemic is active within the time $[0, 100]$ but disappears during the time $[100, 120]$. In the 1D experiment, we fix $k_i = 0.0012$ and we take $k_r \sim U(0.005, 0.2)$. In the 2D experiment, both parameters vary: $k_i \sim U(5 \cdot 10^{-4}, 0.003)$ and $k_r \sim U(0.005, 0.2)$. Figure 11 reports results for such experiments, including the comparison of the probability satisfaction function obtained through the ABC-automaton method with that obtained through Prism model checker (numerical method for both 1D and 2D experiments). One can see that the satisfaction probability function is well reconstructed in both cases.

Whereas the success is expected in the 1D experiment because the histogram suggests a Gaussian-like shape of the density, the result for the 2D experiment is more remarkable because it is hard to guess the density shape based on the 2D histogram. However, our algorithm managed to reproduce the same complex shape and values given by the Model Checking estimates.

4.3. Intracellular viral infection

We consider a model of cell viral infection [55] described by Equations (7).

$$\begin{aligned} N + T &\xrightarrow{k_1 X_T c_n} G + T & T &\xrightarrow{k_4 X_T} \varnothing \\ N + G &\xrightarrow{k_2 X_G c_n} T & S &\xrightarrow{k_5 X_S} \varnothing \\ N + A + T &\xrightarrow{k_3 X_T c_n c_a} S + T & G + S &\xrightarrow{k_6 X_G X_S} V \end{aligned} \tag{7}$$

N represents the nucleotides and A the amino acids. G is the genomic nucleic acids, T the template nucleic acids, S the viral structural protein and V the secreted virus. In this model, we assume nucleotides and amino acids have constant concentrations c_n and c_a: we suppose these species are in large number.

Satisfaction probability function estimation. A 2D experiment is considered with the following logical property: $\varphi = \mathbf{G}^{[0,50]} G \leq 10 \wedge \mathbf{F}^{[50,200]} G > 100$. This property expresses that the species G remains stable and low

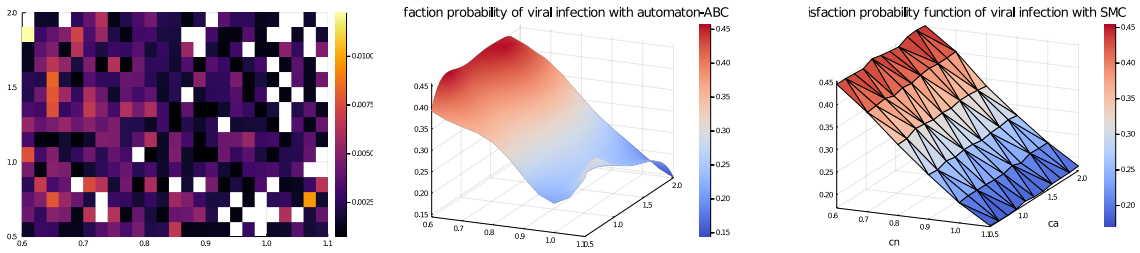

Figure 12: Results for the intracellular viral infection model with $\varphi = \mathbf{G}^{[0,50]} G \leq 10 \wedge \mathbf{F}^{[50,200]} G > 100$. Left : the 2D histogram of the automaton-ABC posterior. Center: kernel density estimation of the satisfaction probability function. Right: estimation of the satisfaction probability function by Monte-Carlo simulations (based on a 99% confidence level and approximation 0.01).

within time $[0, 50]$, and then a burst of speed in the creation of G occurs within $[50, 100]$, which is needed material to the creation of the virus V.

We vary the nucleotides and amino acids concentrations: $c_n \sim \mathcal{U}(0.6, 1.1)$ and $c_a \sim \mathcal{U}(0.5, 2.0)$. Figure 12 reports the results of the experiment. Considering the end time of the formula, each simulation of this model is more computationally expensive than the other considered models. But our methodology still allows a proper run of the experiment, and we get a good approximation of the satisfaction function (about $7 \cdot 10^3$ simulations).

4.4. About implementation of automaton-ABC method

The implementation of automaton-ABC methods leads to a Julia package[10]. It includes both simulation and synchronised simulation of MPM, support for CRN representation of MPMs and automaton-ABC related methods.

ABC related algorithms are distributed and the experiments were performed using HPC resources from the "Mésocentre" computing center of CentraleSupélec and École Normale Supérieure Paris-Saclay supported by CNRS and Région Île-de-France (http://mesocentre.centralesupelec.fr/).

Tables 1 and 2 shows performance results for the whole set of experiments. We omit Kernel Density Estimation since it only depends on the number of particles N and the choice of the kernel. The computational time of Least-Squares Cross Validations can be expensive when the kernel is a multivariate beta kernel with a high number of particles ($N \geq 1000$). However, Least Squares Cross-Validation estimates are easily distributable.

When the number of jobs is 120, the run was distributed on the Mesocentre HPC cluster. Otherwise, it was run on a Dell XPS 9370 with CPU Intel i7 8550U @ 1.8GHz x 8 cores.

Exp	ER TR1	ER TR2	ER TR3	SIR 1D
Num. of jobs	1	1	1	1
Num. of sim.	2263	4896	7197	25404
Time (sec)	7.5	10.9	8.7	7.1

Table 1: Performance results for the one-dimensional experiments of automaton-ABC.

Our tool for automaton-ABC is quite efficient because the run of an experiment is rarely higher than one minute. For example, the experiment about the region TR4 of the ER model performs 256000 simulations within a minute over the cluster (without counting the other computations of the ABC algorithm), knowing that each simulation that reaches $TR4$ is about 2000 steps of Stochastic Simulation Algorithm [10].

The SIR 2D experiment has higher computational time than the 1D experiment, even if it is run with 120 jobs. This is explained by the fact that creating and dealing with many jobs has a higher computational cost when the execution time per job is low. It is the case here: less than 7 seconds of computation has to be distributed over 120 jobs, which is not helpful. The viral infection model has a higher computational cost per simulation because it is a much more complex model than the two others.

This computational time has to be put in perspective to classical Statistical Model Checking methods. For example, in experiment $TR4$, the Statistical MC run over a grid of 200 points of a much smaller set than $[0, 100] \times [0, 100]$. It lasted more than one hour, and we saw this grid missed a region of high probability for φ_4 formula.

[10] available at `https://gitlab-research.centralesupelec.fr/2017bentrioum/tcs2021`

Exp	ER TR4	ER TR5	ER TR6
Num. of jobs	120	120	120
Num. of sim.	256641	32367	47649
Time (sec)	66.18	29.2	31.9

Exp	SIR 2D	Viral inf.
Num. of jobs	120	120
Num. of sim.	17284	6125
Time (sec)	13.1	70.1

Table 2: Performance results for the two-dimensional experiments of automaton-ABC.

	R1	R2	R3
Smoothed MC	27000	37200	32400
Automaton-ABC	2263	4896	7197

Table 3: Number of simulations before termination for both algorithms. For automaton-ABC: number of $N = 1000$ particles. For Smoothed MC: each point of the dataset is estimated with 600 trajectories (default value).

5. Discussion

Comparison with Smoothed Model Checking. We discuss some results of Smoothed MC algorithm for which a Python version is given in [23]. The main idea of Smoothed MC algorithm is to estimate the satisfaction function $\theta \rightarrow Pr(\varphi|\mathcal{M}_\theta)$ over an interval $I \subset \Theta$ with Gaussian Processes (GP) in a Bayesian framework. Initially, the function is estimated over d points of I using classical stochastic model checking: this establishes a data set D. Given a GP prior and a likelihood of the observations that are, by nature, *Binomial*, one can then compute a posterior $p(\theta|D)$ that is also a GP thanks to the Bayes rule (the constant of normalisation is computed with the Expectation-Propagation algorithm). At each iteration i, a new $\theta_{d+i} \in I$ is added in D and $Pr(\varphi|\theta_{d+i})$ is estimated through classical stochastic model checking to increase the training set D and therefore have a better accuracy until a convergence criterion is met. For the sake of comparison, we reproduced the 1D experiments for the ER model (Section 4.1) using the Smoothed MC tool [23]. Figure 13 depicts plots of the satisfaction probabilities obtained with Smoothed MC, which show a good agreement with those obtained with the ABC-automaton approach (Figure 8). Table 3 reports the number of trajectories generated by both algorithms, showing a clear advantage for the ABC-automaton approach. In this table, we do not include any simulation for the estimation of $Pr(\varphi|\mathcal{M}_{\theta*})$ for $\theta* \in \Theta$, which is required for the estimation of the constant K. On the contrary, this table does not show the cost of normalisation constant estimation in the Smoothed MC posterior by Expectation-Propagation. Also, a default value of 600 trajectories is set for each satisfiability probability estimation by statistical model checking, which should be higher if one wants a high confidence level and an approximation of 0.01.

In return, the ABC method does not have the same statistical guarantees as Smoothed MC because it does not assume any specific form for the ABC density π_{ABC}: we minimise the Least Squares Cross-Validation criterion to have the better trade-off between bias and variance over the $N = 1000$ particles in kernel density estimation. To run 2D experiments, we have adapted the available Smoothed MC code, but preliminary tests seem to indicate a prohibitive computational time. Indeed, convergence for 2D experiments such as R4 required many more trajectories simulation ($\sim 2 \cdot 10^5$). The end time of simulations is much higher than the 1D experiments ($t_2 = 10.0 \gg 0.075$). The 2D experiments on the ER system show that our method gives an efficient way to identify the region of the parameter space where the satisfaction function is positive. Once such exploration is performed, one can either use our kernel density estimation based method as it is done in this paper (on which a large literature exists), or regression methods over the identified region, such as Smoothed MC that trade higher computational cost for better statistical guarantees. We further remark that since the main computational cost of the ABC approach is the simulation of trajectories, our method is well suited for distributed computing, which is not the case for all regression methods (e.g. Smoothed MC is, by nature, sequential).

Comparison with robustness-based approaches. The framework we introduce in this paper naturally compares with approaches for assessing the *robustness* of a temporal logic property w.r.t. (deterministic) continuous and hybrid models [41, 56]. Intuitively the notion of robustness has been proposed so to overcome the limits of methods for establishing the Boolean satisfaction of a formula (i.e. model checking) by introduction of a *scoring function* that allow for quantifying how "strongly" a trace σ (hence a model) satisfy a formula φ. More precisely the *robustness* of σ w.r.t. to φ is a real-valued function that expresses how far σ is from dissatisfying (positive robustness) or satisfying (negative robustness) φ. Initially introduced only w.r.t. the *space perspective*, spatial-robustness [57] has then been extended to the *time perspective* yielding

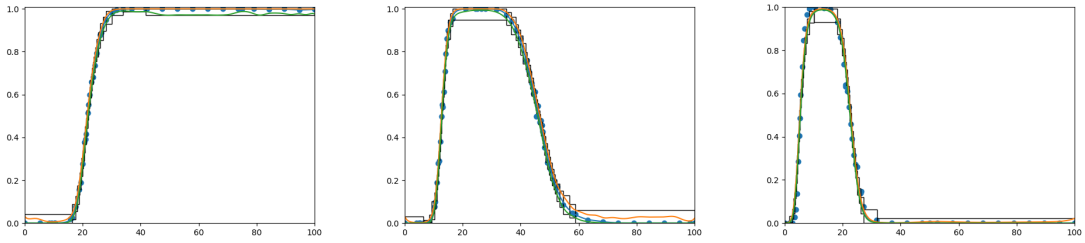

Figure 13: Estimation of the satisfaction probability functions in experiments R1, R2, R3 for the ER system reported in Figure 8 with Smoothed MC algorithm. In blue: the estimated function; in green: the lower bound; in orange: the upper bound.

the notions of *temporal-robustness* as well as the combined *space/time-robustness* [41]. Such robustness measures have then been plugged into dedicated simulation-based procedures for identifying the subspace of a non-probabilistic model's parameters for which a formula is guaranteed to hold (e.g. the BREACH tool [56]).
In our framework, on the other hand, the scoring of traces is done through so-called *satisfiability-distance* measures, e.g. (3), (4), (5), i.e. non-negative functions that express how far σ is from satisfying φ by taking into account the combined spatio-temporal dimension (and yielding 0 if $\sigma \models \varphi$): therefore our *satisfiablity-distance* semantically correspond to negative *trace-robustness*. The main limitation of our approach is that the definition of *satisiability-distances* we gave is limited to a fragment of the full STL syntax, whereas, conversely *robustness measures* are defined (recursively) on the entire STL language. It would be certainly worth, as future developments, to consider whether the *robustness measures* definition could be adapted, so to replace *satisiability-distances*, in our framework for exploring the parameter space of MPMs. In this respect it is worth pointing out that existing *robustness-based* frameworks [41, 56], being addressed to *deterministic* (i.e. non-probabilistic) models, cannot straight away be applied to probabilistic systems, as parameter search for stochastic systems requires to plug a trace scoring measure within a procedure for estimating the probability satisfaction function of φ. .

6. Conclusion

We developed a novel ABC-based framework to address probabilistic verification of temporal logic formulae against parametric MPMs. Our method allows an efficient exploration of the parameter space thanks to a formal definition of distance between the model's trajectories and the satisfiability region associated with the considered formula. Such distances are measured with linear hybrid automata.

We have shown that the estimation of the satisfaction probability function for a formula φ can be achieved thanks to a sound relation with the resulting ABC posterior distribution of the automaton-ABC. The estimation of the probability of satisfying φ boils down to a density estimation problem, which opens up to future improvements and tools from an active research area in statistics. We tested this novel approach through many case studies, which showed promising results. Our method can also be used as a heuristic for estimating the satisfaction probability function because it explores the parameter space w.r.t a logical property in an efficient manner and then can be combined with other regression methods over a subset of the parameter space. Some aspects remain to be considered to extend this work, including i) support for automatic generation of the LHA corresponding to a given formula (which is currently done manually), including the case of formulae for non-elementary regions; ii) the extension to a less constrained set of formulae (currently the method only considers a fragment of MITL temporal logic formulae); iii) the integration of the automaton-ABC framework within the Cosmos statistical model checking platform.

References

[1] T. Toni, D. Welch, N. Strelkowa, A. Ipsen, M. P. Stumpf, Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems, Journal of the Royal Society Interface 6 (31) (2009) 187–202. doi:10.1098/rsif.2008.0172.

[2] O. Ratmann, C. Andrieu, C. Wiuf, S. Richardson, Model criticism based on likelihood-free inference, with an application to protein network evolution, Proceedings of the National Academy of Sciences (2009). doi:10.1073/pnas.0807882106.

[3] K. Koutroumpas, P. Ballarini, I. Votsi, P. H. Cournède, Bayesian parameter estimation for the Wnt pathway: An infinite mixture models approach, Vol. 32, 2016, pp. 781–789. doi:10.1093/bioinformatics/btw471.

[4] O. Lenive, P. D. Kirk, M. P. Stumpf, Inferring extrinsic noise from single-cell gene expression data using approximate Bayesian computation, BMC Systems Biology (2016). doi:10.1186/s12918-016-0324-x.

[5] V. Plagnol, S. Tavaré, Approximate bayesian computation and mcmc, in: Monte Carlo and Quasi-Monte Carlo Methods 2002, Springer, 2004, pp. 99–113.

[6] M. Bentriou, P. Ballarini, P.-H. Cournède, Reachability design through approximate bayesian computation, in: International Conference on Computational Methods in Systems Biology, Springer, 2019, pp. 207–223.

[7] H. Kitano, Foundations of Systems Biology, MIT Press, 2002.

[8] D. J. Wilkinson, Stochastic modelling for quantitative description of heterogeneous biological systems, Nature Reviews Genetics 10 (2) (2009) 122–133. doi:10.1038/nrg2509.
URL https://doi.org/10.1038/nrg2509

[9] A. Ribeiro, R. Zhu, 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) (2006) 1630–1639. doi:10.1089/cmb.2006.13.1630.
URL http://dx.doi.org/10.1089/cmb.2006.13.1630

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

[11] K. R. Sanft, S. Wu, M. Roh, J. Fu, R. K. Lim, L. R. Petzold, StochKit2: software for discrete stochastic simulation of biochemical systems with events, Bioinformatics 27 (17) (2011) 2457–2458. arXiv:https://academic.oup.com/bioinformatics/article-pdf/27/17/2457/600221/btr401.pdf, doi:10.1093/bioinformatics/btr401.
URL https://doi.org/10.1093/bioinformatics/btr401

[12] M. Kwiatkowska, G. Norman, D. Parker, Stochastic model checking, in: Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation, Vol. 4486 of LNCS, Springer, 2007, pp. 220–270.

[13] C. Baier, B. Haverkort, H. Hermanns, J.-P. Katoen, Model-Checking Algorithms for Continuous-Time Markov Chains., Software Engineering, IEEE Transactions on 29 (2003) 524–541. doi:10.1109/TSE.2003.1205180.

[14] K. Sen, M. Viswanathan, G. Agha, On statistical model checking of stochastic systems, in: Proc. CAV'05, 2005.

[15] H. Younes, R. Simmons, Statistical probabilistic model checking with a focus on time-bounded properties, Inf. Comput. 204 (9) (2006).

[16] C. Baier, J.-P. Katoen, Principles of Model Checking (Representation and Mind Series), The MIT Press, 2008.

[17] L. Brim, M. Češka, S. Dražan, D. Šafránek, Exploring parameter space of stochastic biochemical systems using quantitative model checking, in: N. Sharygina, H. Veith (Eds.), Computer Aided Verification, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 107–123.

[18] A. Aziz, K. Sanwal, V. Singhal, R. Brayton, Verifying continuous time Markov chains, in: R. Alur, T. A. Henzinger (Eds.), Computer Aided Verification, Springer Berlin Heidelberg, Berlin, Heidelberg, 1996, pp. 269–276.

[19] T. Han, J. Katoen, A. Mereacre, Approximate parameter synthesis for probabilistic time-bounded reachability, in: 2008 Real-Time Systems Symposium, 2008, pp. 173–182.

[20] M. Češka, F. Dannenberg, M. Kwiatkowska, N. Paoletti, Precise parameter synthesis for stochastic biochemical systems, in: P. Mendes, J. O. Dada, K. Smallbone (Eds.), Computational Methods in Systems Biology, Springer International Publishing, Cham, 2014, pp. 86–98.

[21] G. W. Molyneux, V. B. Wijesuriya, A. Abate, Bayesian verification of chemical reaction networks (2020). arXiv:2004.11321.

[22] L. Bortolussi, D. Milios, G. Sanguinetti, Smoothed model checking for uncertain Continuous-Time Markov Chains, Inf. Comput. 247 (2016) 235–253. doi:10.1016/j.ic.2016.01.004.
URL https://doi.org/10.1016/j.ic.2016.01.004

[23] L. Bortolussi, S. Silvetti, Bayesian statistical parameter synthesis for linear temporal properties of stochastic models, in: D. Beyer, M. Huisman (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, Springer International Publishing, Cham, 2018, pp. 396–413.

[24] V. Kulkarni, Modeling and Analysis of Stochastic Systems, Third Edition, Chapman & Hall/CRC Texts in Statistical Science, CRC Press, 2016.
URL https://books.google.it/books?id=VgGRDQAAQBAJ

[25] L. Bortolussi, F. Cairoli, Bayesian abstraction of Markov population models, in: International Conference on Quantitative Evaluation of Systems (QEST2019), Springer, 2019, pp. 259–276.

[26] W. O. Kermack, A. G. McKendrick, A Contribution to the Mathematical Theory of Epidemics, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 115 (772) (1927) 700–721. doi:10.1098/rspa.1927.0118.

[27] O. Maler, D. Nickovic, Monitoring temporal properties of continuous signals, in: Y. Lakhnech, S. Yovine (Eds.), Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 152–166.

[28] E. M. Clarke, O. Grumberg, D. A. Peled, Model Checking, MIT Press, 1999.

[29] C. Baier, J. Katoen, Principles of model checking, MIT Press, 2008.

[30] A. Pnueli, The temporal logic of programs, in: 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977, IEEE Computer Society, 1977, pp. 46–57. doi:10.1109/SFCS.1977.32.
URL https://doi.org/10.1109/SFCS.1977.32

[31] E. M. Clarke, E. A. Emerson, A. P. Sistla, Automatic verification of finite state concurrent systems using temporal logic specifications: A practical approach, in: J. R. Wright, L. Landweber, A. J. Demers, T. Teitelbaum (Eds.), Conference Record of the Tenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas, USA, January 1983, ACM Press, 1983, pp. 117–126. doi:10.1145/567067.567080.
URL https://doi.org/10.1145/567067.567080

[32] R. Alur, T. A. Henzinger, Logics and models of real time: A survey, in: J. W. de Bakker, C. Huizing, W. P. de Roever, G. Rozenberg (Eds.), Real-Time: Theory in Practice, Springer Berlin Heidelberg, Berlin, Heidelberg, 1992, pp. 74–106.

[33] H. Hansson, B. Jonsson, A logic for reasoning about time and reliability, Formal Aspects of Computing 6 (1995) 512–535.
URL http://www.es.mdh.se/publications/13-

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

[35] M. Kwiatkowska, G. Norman, D. Parker, {PRISM} 4.0: Verification of Probabilistic Real-time Systems, in: G. Gopalakrishnan, S. Qadeer (Eds.), Proc. 23rd International Conference on Computer Aided Verification (CAV'11), Vol. 6806 of LNCS, Springer, 2011, pp. 585–591.

[36] C. Dehnert, S. Junges, J.-P. Katoen, M. Volk, A storm is coming: A modern probabilistic model checker, in: Proc. 29th International Conference on Computer Aided Verification (CAV'17), 2017.

[37] H. Younes, Ymer: A statistical model checker, in: Proc. CAV'05, LNCS 3576, 2005.

[38] A. Legay, S. Sedwards, L. Traonouez, Plasma lab: A modular statistical model checking platform, in: ISoLA (1), Vol. 9952 of Lecture Notes in Computer Science, 2016, pp. 77–93.

[39] P. Ballarini, H. Djafri, M. Duflot, S. Haddad, 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), IEEE Computer Society Press, 2011, pp. 143–144.

[40] K. Sen, M. Viswanathan, G. Agha, Vesta: A statistical model-checker and analyzer for probabilistic systems, in: Second International Conference on the Quantitative Evaluation of Systems (QEST'05), 2005, pp. 251–252. doi:10.1109/QEST.2005.42.

[41] A. Donzé, O. 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, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 92–106.

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

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

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

[45] J.-M. Marin, P. Pudlo, C. P. Robert, R. J. Ryder, Approximate bayesian computational methods, Statistics and Computing 22 (6) (2012) 1167–1180. doi:10.1007/s11222-011-9288-2.
URL https://doi.org/10.1007/s11222-011-9288-2

[46] S. A. Sisson, Y. Fan, M. Beaumont, Handbook of approximate Bayesian computation, Chapman and Hall/CRC, 2018.

[47] M. A Nunes, D. J Balding, On optimal selection of summary statistics for approximate bayesian computation, Statistical applications in genetics and molecular biology 9 (2010) Article34. doi:10.2202/1544-6115.1576.

[48] M. A. Beaumont, J.-M. Cornuet, J.-M. Marin, C. P. Robert, Adaptive approximate bayesian computation, Biometrika 96 (4) (2009) 983–990.
URL http://www.jstor.org/stable/27798882

[49] P. Del Moral, A. Doucet, A. Jasra, Sequential monte carlo samplers, Journal of the Royal Statistical Society B 68 (3) (2006) 411–436. doi:10.1111/j.1467-9868.2006.00553.x.
URL http://www.blackwell-synergy.com/doi/abs/10.1111/j.1467-9868.2006.00553.x;http://onlinelibrary.wiley.com/doi/10.1111/j.1467-9868.2006.00553.x/full

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

[51] J. Chacón, T. Duong, Multivariate Kernel Smoothing and its Applications, 2018. doi:10.1201/9780429485572.

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

[53] T. Bouezmarni, J. V. Rombouts, Nonparametric density estimation for multivariate bounded data, Journal of Statistical Planning and Inference 140 (1) (2010) 139 – 152. doi:https://doi.org/10.1016/j.jspi.2009.07.013.
URL http://www.sciencedirect.com/science/article/pii/S0378375809002249

[54] L. Michaelis, M. Menten, K. Johnson, R. Goody, The Original Michaelis Constant: Translation of the 1913 Michaelis-Menten Paper, Biochemistry 50 (2011) 8264–8269. doi:10.1021/bi201284u.

[55] E. Haseltine, J. Rawlings, Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics, The Journal of Chemical Physics 117 (10 2002). doi:10.1063/1.1505860.

[56] A. Donzé, T. Ferrère, O. Maler, Efficient robust monitoring for stl, in: N. Sharygina, H. Veith (Eds.), Computer Aided Verification, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 264–279.

[57] G. E. Fainekos, G. J. Pappas, Robustness of temporal logic specifications, in: K. Havelund, M. Núñez, G. Roşu, B. Wolff (Eds.), Formal Approaches to Software Testing and Runtime Verification, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 178–192.

Appendix A.

Proposition 3.2. Let X_O be a species of an MPM model $\mathcal{M}$, $\mathcal{A}_F$ be the distance LHA automaton 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:

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

where $\mathcal{A}_F(\sigma).d$ denotes the value stored in variable d of automata $\mathcal{A}_F$ when $\mathcal{A}_F$ has synchronised with σ.

Proof. We recall that (s, l, v) denotes a generic state of the product process $\mathcal{M}\times\mathcal{A}_F$, where s is a state of $\mathcal{M}$, l is a location of $\mathcal{A}_F$ and v' a valuations of the variables of $\mathcal{A}_F$. In the remainder we use the notation $(s, l, v) \xrightarrow{*} (s', l', v')$ to indicate that state (s', l', v') of $\mathcal{M} \times \mathcal{A}_F$ is reachable from (s, l, v), i.e. $(s, l, v) \xrightarrow{*} (s', l', v')$ means that there exists a finite sequence of (synchronised/autonomous) transitions that takes $\mathcal{M} \times \mathcal{A}_F$ from (s, l, v) to (s', l', v').

The proof boils down to showing that the synchronisation of an arbitrary path $\sigma \in Path_{\mathcal{M}}$ with $\mathcal{A}_F$ leads the product process $\mathcal{M} \times \mathcal{A}_F$ to reach, from the initial state (s_0, l_0, v^0), a state $(\sigma@t^*, l_2, v^*)$, (i.e. l_2 being the only accepting location of $\mathcal{A}_F$), where $t^* \in \mathbb{R}_{\geq 0}$ is the time instant when $(\sigma@t^*, l_2, v^*)$ is reached and $v^*(d)$ is equal to $d(\sigma, \mathbf{F}^{[t_1,t_2]} x_1 \leq x_O \leq x_2)$ as in (3).

Given an arbitrary path $\sigma \in Path_{\mathcal{M}}$ and its projection w.r.t. the observed species X_O, i.e. σ_O, we denote[11] $t_{l_1} = t_{last}(t_1)$, resp. $t_{l_2} = t_{last}(t_2)$, the time instant of the last jump contained in σ_O before t_1, resp. t_2 (see examples in Figure A.14). We consider the following partition of the set of paths of $\mathcal{M}$, i.e. $Path_{\mathcal{M}} = Path_{\mathcal{M}}^{0=t_{l_1}=t_{l_2}} \cup Path_{\mathcal{M}}^{0=t_{l_1}<t_{l_2}} \cup Path_{\mathcal{M}}^{0<t_{l_1}=t_{l_2}} \cup Path_{\mathcal{M}}^{0<t_{l_1}<t_{l_2}}$ where $Path_{\mathcal{M}}^{0=t_{l_1}=t_{l_2}}$ is the set of paths whose projection σ_O contain no jumps within t_2, $Path_{\mathcal{M}}^{0=t_{l_1}<t_{l_2}}$ are the paths that contain no jumps within t_1 but at least one jump in $[t_1, t_2]$, $Path_{\mathcal{M}}^{0<t_{l_1}=t_{l_2}}$ the paths that contain at least one jump within t_1 and no jumps in $[t_1, t_2]$ and $Path_{\mathcal{M}}^{0<t_{l_1}<t_{l_2}}$ the paths that contain at least one jump within t_1 and at least one jumps in $[t_1, t_2]$. Examples of paths characterising such a partition of $Path_{\mathcal{M}}$ are given in Figure A.14.

Let $(s, l, [t, n, n', d]])$ denote a generic state of the product process $\mathcal{M} \times \mathcal{A}_F$, where $s \in S$ is a state of $\mathcal{M}$, $l \in L$ is a location of $\mathcal{A}_F$ and $[t, n, n', d] \in \mathbb{R}^4$ are the values of the four variables of $\mathcal{A}_F$. In the remainder we denote μ the propositional formula $(x_1 \leq x_O \leq x_2)$, s_t the state of an arbitrary path σ_O is in at time $t \in \mathbb{R}_{\geq 0}$, i.e. $s_t \equiv \sigma_O@t$, $d_0 = min(\sqrt{t_1^2 + (s_{0\_O} - x_2)^2}), \sqrt{t_2^2 + (s_{0\_O} - x_1)^2})$ the distance between the initial state s_0 of σ and the region $[x_1, x_2] \times [t_1, t_2]$ and $d_m(t) = min(|s_t - x_1|, |s_t - x_2|)$ is the minimal distance from $[x_1, x_2]$ for the segment of σ_O delimited by time interval $[t_1, t]$ with $t \in [t_1, t_2]$.

1) $\sigma \in Path_{\mathcal{M}}^{0=t_{l_1}=t_{l_2}}$. In this case σ_O is constant (at least) until t_2. We distinguish between 2 cases:
a) $t_1 = 0$ then either the condition μ is satisfied in the initial state s_0 i.e. $x_1 \leq s_0 \leq x_2$ and hence the synchronisation of σ with $\mathcal{M}$ yields the following unique path on $\mathcal{M} \times \mathcal{A}_F$: $(s_0, l_0, [0, 0, 0, 0]) \xrightarrow[\sharp]{0} (s_0, l_1, [0, s_0, 0, \infty]) \xrightarrow[\sharp]{0} (s_0, l_2, [0, s_0, 0, 0])$ and therefore $\mathcal{A}_F(\sigma).d = 0 = d(\sigma, \mathbf{F}^{[0,t_2]} x_1 \leq x_O \leq x_2)$, or μ is not satisfied in the initial state, i.e. $x_1 > s_0 \vee s_0 > x_2$ in which case the synchronisation of σ with $\mathcal{M}$ yields the following unique path on $\mathcal{M} \times \mathcal{A}_F$: $(s_0, l_0, [0, 0, 0, 0]) \xrightarrow[\sharp]{0} (s_0, l_1, [0, s_0, 0, \infty]) \xrightarrow[\sharp]{0} (s_0, l_3, [0, s_0, s_0, min(|s_0 - x_1|, |s_0 - x_2|)]) \xrightarrow[\sharp]{0} (s_0, l_2, [t_2, s_0, s_0, min(|s_0 - x_1|, |s_0 - x_2|)])$
b) $t_1 > 0$ then either i) $s_0 \models \mu$ hence the synchronisation yields the following unique path on $\mathcal{M} \times \mathcal{A}_F$: $(s_0, l_0, [0, 0, 0, 0]) \xrightarrow[\sharp]{0} (s_0, l_1, [0, s_0, 0, \infty]) \xrightarrow[\sharp]{0} (s_0, l_3, [0, s_0, s_0, 0]) \xrightarrow[\sharp]{0} (s_0, l_2, [t_2, s_0, s_0, 0])$ and therefore $\mathcal{A}_F(\sigma).d = 0 = d(\sigma, \mathbf{F}^{[0,t_2]} x_1 \leq x_O \leq x_2)$, or ii) $s_0 \not\models \mu$ hence the synchronisation yields the following unique path on $\mathcal{M} \times \mathcal{A}_F$: $(s_0, l_0, [0, 0, 0, 0]) \xrightarrow[\sharp]{0} (s_0, l_1, [0, s_0, 0, \infty]) \xrightarrow[\sharp]{0} (s_0, l_3, [0, s_0, s_0, d_0]) \xrightarrow[\sharp]{0} (s_0, l_2, [t_2, s_0, s_0, d_0])$ and therefore $\mathcal{A}_F(\sigma).d = d_0 = d(\sigma, \mathbf{F}^{[0,t_2]} x_1 \leq x_O \leq x_2)$.

2) $\sigma \in \mathbf{Path}_{\mathcal{M}}^{\mathbf{0=t_{l_1}<t_{l_2}}}$. In this case σ_O is constant until t_1 and contains at least 1 jump in $[t_1, t_2]$ and $t_{l_2} \in [t_1, t_2]$ is the time of the last jump before t_2. We distinguish between 2 cases:
a) $s_0 \models \mu$, in this case the synchronisation of σ with $\mathcal{A}_F$ yields the following unique path $(s_0, l_0, [0, 0, 0, 0]) \xrightarrow[\sharp]{0} (s_0, l_1, [0, s_0, 0, \infty]) \xrightarrow[\sharp]{0} (s_0, l_2, [0, s_0, s_0, 0]) \xrightarrow[r]{t'} (s_{t'}, l_1, [0, s_{t'}, s_0, 0]) \xrightarrow[\sharp]{0} (s_{t'}, l_2, [t', s_{t'}, s_0, 0])$ where

[11] based on (3).

$t' \in [t_1, t_2]$ is the time of the first jump that occurs on σ_O within $[t_1, t_2]$ and r is corresponding reaction occurred at t'; therefore $\mathcal{A}_F(\sigma).d = 0 = d(\sigma, \mathbf{F}^{[0,t_2]} x_1 \leq x_O \leq x_2)$.
b) $s_0 \not\models \mu$, in this case we distinguish between 2 further cases: i) $\exists t' \in [t_1, t_2]$ such that $s'_t \models \mu$, in this case the synchronisation of σ with $\mathcal{A}_F$ yields a unique path $(s_0, l_0, [0,0,0,0]) \xrightarrow[\sharp]{0}$
$(s_0, l_1, [0, s_0, 0, \infty]) \xrightarrow[\sharp]{0} (s_0, l_3, [0, s_0, s_0, d_0]) \xrightarrow{*}$
$(s_{t'}, l_1, [t', s_{t'}, s_0, min(d_m(t'), d_0)]) \xrightarrow[\sharp]{0}$
$(s_{t'}, l_2, [t', s_{t'}, s_{t'}, 0]$ or ii) $\forall t' \in [t_1, t_2], s'_t \not\models \mu$ and in this case the synchronisation of σ with $\mathcal{A}_F$ yields a unique path $(s_0, l_0, [0,0,0,0]) \xrightarrow[\sharp]{0}$
$(s_0, l_1, [0, s_0, 0, \infty]) \xrightarrow[\sharp]{0} (s_0, l_3, [0, s_0, s_0, d_0]) \xrightarrow{*}$
$(s_{t'}, l_1, [t_{l_2}, s_{t'}, s_0, min(d_m(t'), d_0)]) \xrightarrow[\sharp]{0}$
$(s_{t'}, l_3, [t_{l_2}, s_{t'}, s_{t'}, min(d_m(t_{l_2}), d_0)] \xrightarrow[\sharp]{t_2 - t_{l_2}}$
$(s_{t'}, l_2, [t_2, s_{t'}, s_{t'}, min(d_m(t_{l_2}), d_0)]$ where $t' \in [t_1, t_{l_2}]$ is the time instant of the last but one jump before t_2. Therefore $\mathcal{A}_F(\sigma).d = min(d_m(t_{l_2}), d_0) = d(\sigma, \mathbf{F}^{[0,t_2]} x_1 \leq x_O \leq x_2)$.

3) $\sigma \in \mathbf{Path}_{\mathcal{M}}^{\mathbf{0<t_{l_1}=t_{l_2}}}$
Similar to previous cases.

4) $\sigma \in Path_{\mathcal{M}}^{0<t_{l_1}<t_{l_2}}$
Similar to previous cases. ■

Appendix B. Automaton $\mathcal{A}_{G \wedge F}$

Figure B.15 depicts automaton $\mathcal{A}_{G \wedge F}$ corresponding to the conjunction of a G formula with and F formula such that the time-bounding interval of the G formula temporally precedes that of the F formula. Automaton $\mathcal{A}_{G \wedge F}$ is simply obtained by concatenation of $\mathcal{A}_G$ and $\mathcal{A}_F$.

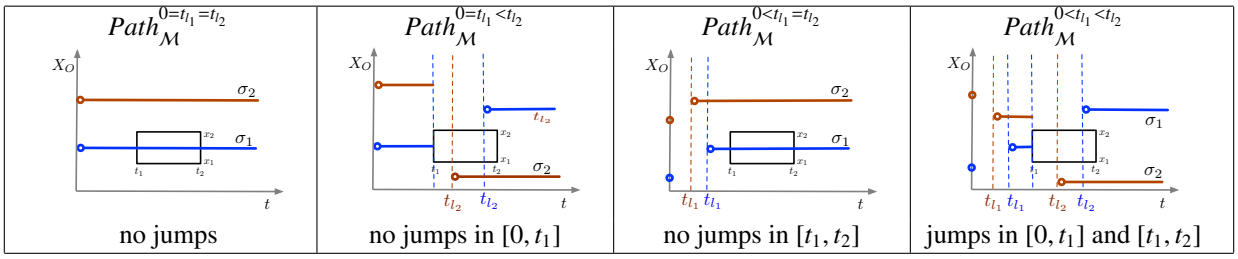

Figure A.14: Examples of paths belonging to the different subsets of the partition $Path_{\mathcal{M}} = Path_{\mathcal{M}}^{0=t_{l_1}=t_{l_2}} \cup Path_{\mathcal{M}}^{0=t_{l_1}<t_{l_2}} \cup Path_{\mathcal{M}}^{0<t_{l_1}=t_{l_2}} \cup Path_{\mathcal{M}}^{0<t_{l_1}<t_{l_2}}$. Vertical dashed lines delimit an unspecified part of a path.

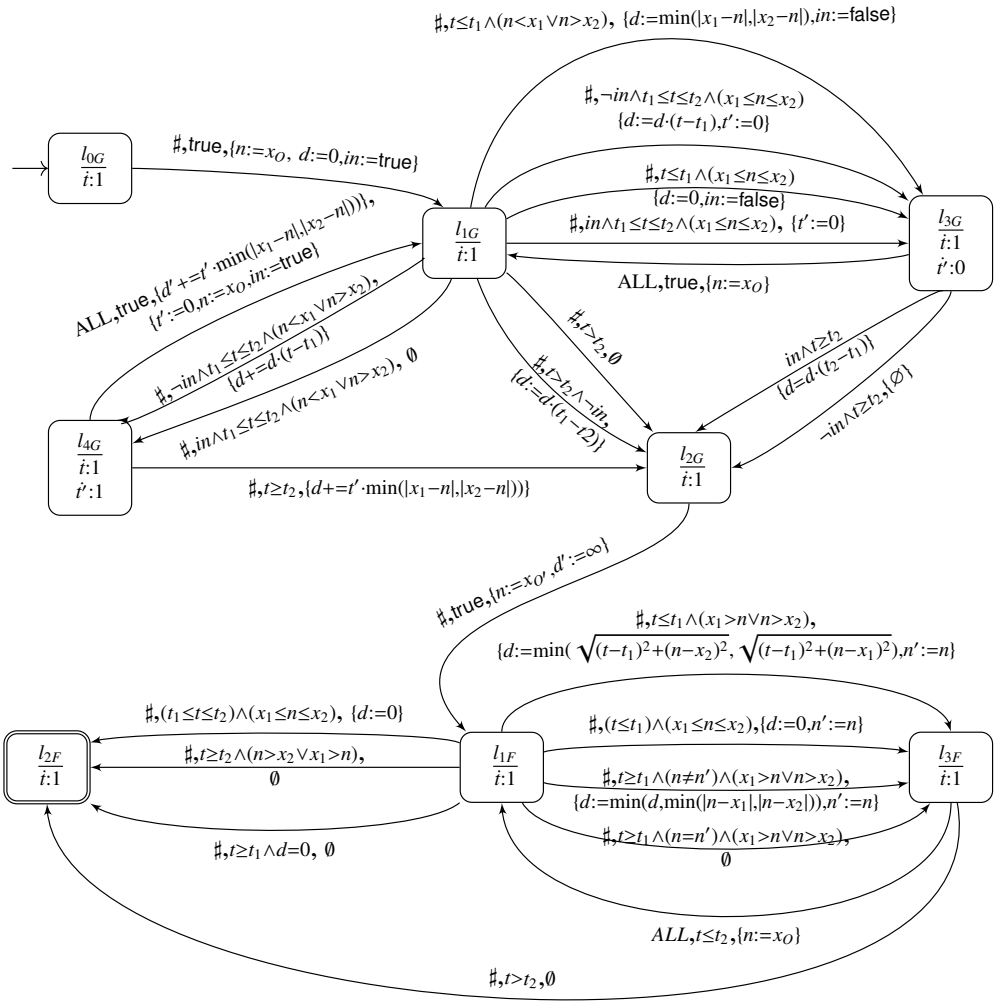

Figure B.15: LHA distance automaton $\mathcal{A}_{G\wedge F}$ for formula $\mathbf{G}^{[t_1,t_2]}(x_1 \le x_O \le x_2) \wedge \mathbf{F}^{[t_3,t_4]}(x_3 \le x_{O'} \le x_4)$, with $t_2 \le t_3$ (i.e. the G region precedes the F region), and where x_O, resp. $x_{O'}$, denotes the population of species O, resp. O'