BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

NOVEL METHODS FOR SEMI-QUANTITATIVE ANALYSIS OF BIOCHEMICAL SYSTEMS

NOVÉ METODY PRO SEMIKVANTITATIVNÍ ANALÝZU BIOCHEMICKÝCH SYSTÉMŮ

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR AUTOR PRÁCE	**JAN BÍL**
SUPERVISOR VEDOUCÍ PRÁCE	**RNDr. MILAN ČEŠKA, Ph.D.**

BRNO 2021

Department of Intelligent Systems (DITS) Academic year 2020/2021

Bachelor's Thesis Specification

24079

Student:	**Bíl Jan**
Programme:	Information Technology
Title:	**Novel Methods for Semi-Quantitative Analysis of Biochemical Systems**
Category:	Formal Verification

Assignment:

1. Study the current computational methods for automated analysis of chemical reaction networks including semi-quantitative techniques.
2. Evaluate the performance and applicability of these techniques on practically relevant case-studies and identify their limitations.
3. Design possible improvements and extensions of these methods including analysis against temporal and functional specifications.
4. Implement the improvements and extensions within the existing tool SeQuaiA.
5. Carry out a detailed evaluation of the implemented methods including an extension of the existing benchmarks.

Recommended literature:

- Češka M., Křetínský J. (2019) Semi-quantitative Abstraction and Analysis of Chemical Reaction Networks. Computer Aided Verification. CAV 2019. Lecture Notes in Computer Science, vol 11561. Springer, 2019.
- Češka M., Chau C., Křetínský J. (2020) SeQuaiA: A Scalable Tool for Semi-Quantitative Analysis of Chemical Reaction Networks. Computer Aided Verification. CAV 2020. Lecture Notes in Computer Science, vol 12224. Springer, 2020.

Requirements for the first semester:

- Items 1, 2 and partially item 3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Češka Milan, RNDr., Ph.D.**
Head of Department:	Hanáček Petr, doc. Dr. Ing.
Beginning of work:	November 1, 2020
Submission deadline:	May 12, 2021
Approval date:	November 11, 2020

Abstract

This thesis aims on providing novel methods for analysis of stochastic bio-chemical systems. It introduces a new population abstraction based on dirac semi-Markov processes. It turned out, that this abstraction is more precise than Continuous time Markov chain abstraction. On this abstraction, analysis methods are provided. Algorithm for transient analysis over this abstraction is described and also novel timed temporal logic formulas, that allow to express interesting biological properties are presented. Further, model-checking algorithm for these formulas is proposed and implemented. Preliminary experiments showing potential of this approach are also described.

Abstrakt

Cílem této práce je poskytnout nové metody pro analýzu stochastických biochemických systémů. V práci je představena nová třída populační abstrakce založené na Diracových semi-Markovových procesech. Na této abstrakci je popsán algoritmus pro výpočet transientní analýzy a také jsou prezentovány nové časované logické formule umožňující ověření vlastností zajímavých pro biology. Dále je představen a implementován model-checking algoritmus pro tyto formule. Rovněž jsou popsány předběžné experimenty ukazující potenciál tohoto přístupu.

Keywords

Bio-chemical models, Continuous time Markov chains, Chemical reaction networks, Model-checking

Klíčová slova

Bio-chemické systémy, Markovovy řetězce se spojitým časem, Sítě chemických reakcí, Model-checking

Reference

BÍL, Jan. *Novel Methods for Semi-Quantitative Analysis of Biochemical Systems.* Brno, 2021. Bachelor’s thesis. Brno University of Technology, Faculty of Information Technology. Supervisor RNDr. Milan Češka, Ph.D.

Rozšířený abstrakt

Analýza komplexních bio-chemických systémů je výpočetně náročný úkol. Tyto systémy jsou často popisovány pomocí sítí chemických reakcí. Tato síť chemických reakcí vede na Markovův řetězec se spojitým časem, který umíme analyzovat. Tento přístup má však několik problémů. Těmi nejzásadnějšími jsou *exploze stavového prostoru* a velký rozdíl mezi raty přechodů systému.

Existující metody pro analýzu sítě chemických reakcí můžeme rozdělit do dvou kategorií: numerický přístup a přístup založený na simulacích. Principem numerických metod je výpočetně analyzovat tyto systémy. Jelikož přímá numerická analýza komplexnějších systémů je výpočetně téměř nemožná, existují různé metody, které výpočet urychlují. Principem simulačních metod je spuštění simulačních běhů. To je typicky rychlejší než numerická analýza, ale je potřeba spustit velké množství simulací, aby se odhalilo i málo pravděpodobné chování.

Nový *semi-kvantitativní* přístup pro řešení zmíněných problémů byl nedávno představen v [7, 8]. Tento přístup je založen na dvou hlavních krocích. 1. Ze sítě chemických reakcí je vytvořena abstrakce. Tato abstrakce spojuje stavy různých populací a druhů do intervalů, čímž se získá konečný stavový prostor. Dále jsou uvažovány sekvence přechodů a některé přechody jsou urychleny. Výsledkem je Markovův řetězec se spojitým časem, popisující chování systému se zredukovaným stavovým prostorem. 2. Nad touto abstrakcí je provedena semi-kvantitativní analýza, která se zaměřuje jen na pravděpodobné chování systému.

V této práci je představena nová abstrakce, která využívá tyto semi-kvantitativní postupy a lépe popisuje chování bio-chemických systémů. V takovém systému čas odchodu ze stavu závisí na přechodu, který je použitý. Toho však není možné dosáhnout v Markovovém řetězci se spojitým časem. Proto je použita speciální třída semi-Markovovského procesu, kterou označujeme jako Dirakův semi-Markovův proces (DSMP). V DSMP abstrakci má každý přechod pravděpodobnost a čas, který tento přechod trvá. Porovnáním simulací nové DSMP abstrakce a abstrakce v podobě Markovova řetězce se spojitým časem se simulací sítě chemických reakcí je ukázáno, že nová abstrakce popisuje chování systémů opravdu lépe.

Pro tuto novou abstrakci je představen algoritmus a implementace transientní analýzy. Transientní analýza poskytnne pro zadaný čas pravděpodobnostní distribuci. Klíčová část této práce je ověřování vlastností vyjádřených pomocí temporální logiky. Aby bylo možné vyjádřit biologicky zajímavé vlastnosti, představujeme novou formuli zřetězených until operátorů, které umožňuje vyjádřit některé z takových vlastnosti. Pro vyjádření některých vlastností, jako je například oscilace, může být tato formule příliš omezující. Proto umožňujeme i ověření nového časově omezeného $\exists$ operátoru, kterým je možné ověřit přítomnost oscilace.

V experimentální části je ukázáno, že některé použité modely je možné analyzovat za pomoci snížené přesnosti i bez použití semi-kvantitativní analýzy. Pro jiné modely, které obsahují velký počet stavů, je však potřeba použít zmíněnou semi-kvantitativní analýzu. Dále jsou ukázány výsledky použití nových formulí.

Novel Methods for Semi-Quantitative Analysis of Biochemical Systems

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of RNDr. Milan Češka, Ph.D The supplementary information was provided by Štefan Matiček, Calvin Chau and Martin Helfrich. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

. . . .
Jan Bíl
May 18, 2021

Acknowledgements

I would like to thank my supervisor RNDr. Milan Češka, Ph.D. for his encouragement and continuous support throughout the work on this project.

Contents

Chapter 1

Introduction

Biochemical systems describes chemical processes in living organisms. As an example, we can think about processes between our cells or viruses or bacteria and many more. Its study has a great potential. Understanding those processes might help improving our lives. For instance modified viruses will become means for supplying our cells with healthy proteins or cure for various diseases will be found. Unfortunately such systems are very complicated and it is difficult to analyze them[24].

Chemical reaction networks (CRN) is language widely used for modelling and analysis of those real world biochemical systems [9]. CRN contains set of reactions and their rates (intuitively speed). The time evolution of CRNs is governed by Master Chemical Equation that leads to potentially infinite discrete space Continuous time Markov chain (CTMC). It describes how the probability of molecular counts of each chemical species evolves in time. Analysis of such system is computationally very intensive task and have several problems.
1. Analysis is computationally difficult, because many biochemical systems often leads to complex dynamics, which includes: *state-space explosion* (number of species can exponentially grow, possibly even to infinity due to unbounded populations), *stochasticity* (races between reactions), *stiffness* (rates of different magnitudes) and *multimodality* (qualitatively different behaviours such as extinction of predators only, or also of preys in the predator-prey models)[13, 22].
2. Rates of reactions are typically not exactly known. Often only order of magnitude is given or rate is completely unknown, which makes analysis more difficult.
3. To see behaviour of the system, analysis is not needed to be precise numerically, but qualitatively. For example, that initial growth of species is followed by its extinction. Unfortunately it is hard to get qualitative analysis without the numerical one.
4. Biologist and engineers also often need to know explanation of the behaviour of the system. As set of system simulations/trajectories or population distributions is not sufficient enough, the ability to provide an accurate explanation for the temporal or steady-state behaviour is another major challenge for the existing techniques. [7, 8]

There exists two main approaches how to analyze CRNs. Simulation based and numerical based techniques. Simulation based approach produces simulates behaviour of the system. It is typically faster then computation of the behaviour, but to reveal less probable behaviour, a lot of simulation must be made. Numerical approach numerically computes behaviour, but for more complicated systems it is rather impossible.
For both approaches various abstractions and approximations exists. First of all, for CRNs with large populations of species, fluid approximation techniques can be used. These are also known as mean field approximations (leading to an ODE characterisation) that can be

extended to higher-order moments and equipped by Gaussian process in Linear Noise Approximation as well as in the form of various hybrid models combining a discrete stochastic process for low population species and a continuous process for large population species. Another existing solution is state space reduction technique. It either truncates low probability states of underlying CTMC or aggregates states to set of states. Last is multi-scale simulation technique. It uses partitioning scheme to speed up the standard simulation algorithm (SSA) [12].

Recently, approach which uses semi-quantitative reasoning was proposed [8]. It uses some techniques to face these problems and to give wanted results. To handle computational complexity and to obtain explanation of behaviour of the system, it uses more of the qualitative approach, instead of pure quantitatively precise approach. This *semi-quantitative* approach provides good look at the behaviour and also some rough timing information as well as information about probability classification of each behaviour.
This semi-quantitative reasoning was used in tool called SeQuaia (Semi-Quantitative analysis). The tool consists of two main levels. Firstly, an abstraction of the systems into semi-quantitative models is done. Then, semi-quantitative analysis over the models is performed. Direct analysis of Continuous time Markov chain (CTMC) derived from CRN is difficult, so desired model is made with some techniques. The state space is abstracted by population model of each species following classic may abstraction described in [14]. That means the population is divided into finitely many intervals. Also, sequences of transitions are considered, which helps to solve the problem of non-deterministic self loops. As a result, self-looping transitions are accelerated, which means taken multiple times. This acceleration helps adequately capture the transitions between population levels. Accelerated transitions are key idea to semi-quantitative abstraction.

In this thesis, focus is on improvement of this approach using novel population abstraction. CTMC does not capture behaviour of the system properly, because both, transition time and probability depends on transitions rate. In fact, exit time heavily depends on the transition that is used. That means, that transition probability and exit time needs to be separated. This is very natural as single abstract transition capture sequence of reactions. Since in CTMC this separation cannot be easily achieved, we propose to use special class of semi-markov process, further denoted as dirac-semi-Markov process (DSMP) having a degenerated distribution of the waiting times – the probability that the waiting time is equal to the expected waiting time is one. Thus, every transition has probability that the transition is taken and waiting time modelling the expected time needed to perform the transition equipped.

Key contribution of this work is novel algorithms and implementation for analysing DSMP. Standard transient analysis, that provides probability distribution of the system for the given time instant, can be computed. Except for transient distribution, also properties given by temporal logic are verified. In order to express properties useful and easily understandable for biologists, chained until formulation is introduced and can be also verified. Chained until is too strict for some properties, which then cannot be verified properly. For example quick oscillation of population is problematic to detect. Because of that, bounded exists operator is introduced, which allows to verify even this kind of properties.

In experimental part, it is shown, that for some models, approximation of the algorithm for computation of the transient analysis can be used with lowered precision providing solid results with much lower computation time. This approximation is based on pre-computing reactions happening in the future with every executed set of reaction. This set is given by precision parameter.

Main contribution of this work is novel model-checking algorithm for DSMP and timed temporal properties. It consists of 3 parts: 1) Stepping algorithm, which allow us to get transient distribution. 2) model checking algorithm for PCTL using untils operator allowing to quantify probability of given behaviour. 3) less restrictive chain of bounded exists operators allowing to quantify properties like oscillation of population. Last two points are the main focus of my work.

Also, experiments over 3 models were made. These experiments shows performance evaluation as well as potential of introduced novel model-checking formulas.

Chapter 2

Preliminaries

In this chapter, preliminary information about CRNs are provided. Also Continuous Time Markov Chains and Continuous Stochastic Logic are discussed as it is basic model obtained from CRN and logic used for numerical analysis of CTMCs and CRNs.

2.1 Chemical reaction networks

Chemical reaction networks (CRNs) is a language used as a formalism for describing chemical systems. CRNs consists of several chemical reactions. Example of such reaction might be familiar reaction:
$2H_2 + O_2 \xrightarrow{\text{k}} H_2O$
The quantities on the left side are called reactants, quantities on the right side of the arrow are called products. Reactants and products are collectively referred to as an species of the reaction. k denotes rate of the reaction [9].

Formally, chemical reaction network $\mathcal{N} = (\Lambda, \mathcal{R})$ is a pair of finite sets. Elements of Λ are species and elements of $\mathcal{R}$ are reactions over them. Species in Λ reacts according to reactions in $\mathcal{R}$. Reaction $\tau \in \mathcal{R}$ is a triple $\tau = (r_\tau, p_\tau, k_\tau)$ where $r_\tau \in \mathbb{N}^{|\Lambda|}$ is the reactant complex, $p_\tau \in \mathbb{N}^{|\Lambda|}$ is the product complex and $k_\tau \in \mathbb{R}_{>0}$ is coefficient associated with the rate of the reaction. Given a reaction $\tau_1 = ([1,1,0],[0,0,2],k_1)$, it is often referred as $\tau_1 : \lambda_1 + \lambda_2 \xrightarrow{k_1} 2\lambda_3$.
$\mathbb{N}^\Lambda$ are discrete states over $\mathcal{N}$. These states might be also referred as configurations. They describe counts of each of the molecular species $\in \Lambda$ in $\mathcal{N}$. Network can move between states, whenever transition happens. However transition can take place only if all reactants are available with in state moved from. Then reactants of transition are removed and products are added, that is how successor state looks. Formally, we can write it as following:
For a state $\mathbf{c} \in \mathbb{N}^\Lambda$ and a reaction α over Λ, we say that $\alpha = (\mathbf{r}, \mathbf{p})$ is applicable to $\mathbf{c}$ if $\mathbf{r} \leq \mathbf{c}$. (This means, enough reactants are available in the state and we can apply transition α). If α is applicable to $\mathbf{c}$, then the result of applying α to $\mathbf{c}$, denoted by $\alpha\ (\mathbf{c})$, is $\mathbf{c}' = \mathbf{c} - \mathbf{r} + \mathbf{p}$. In this case we also write $\mathbf{c} \Rightarrow_\alpha \mathbf{c}'$. We can define *reachable state space* as all states that can be reached by applying sequence of transitions to initial state.

Example 2.1:
How CRNs work can be demonstrated on model called SimpleGene defined as:

$$\begin{aligned}
&\mathcal{N} = (\Lambda, \mathcal{R}) \text{ with} \\
&\Lambda = \{D_{on}, D_{of}, P\} \text{ and} \\
&\mathcal{R} = \{D_{on} \xrightarrow{10} P + D_{on}, P \xrightarrow{0.1} \emptyset, D_{on} + P \xrightarrow{0.001} D_{off}\}\}
\end{aligned}$$

This example shows simple CRN with 3 species and 3 reactions. First reaction produces proteins, second reaction demonstrates degradation of proteins and the last reaction blocks active DNA, which then no longer produces proteins. Consider initial state $\mathbf{c} = D_{on}$. From such state only transition $\alpha = D_{on} \rightarrow P + D_{on}$ of $\mathcal{R}$ is applicable. Whenever this transition is applied we have $\mathbf{c} \Rightarrow_{\alpha} \mathbf{c}'$ where $\mathbf{c}' = D_{on} + P$. Note that from state $\mathbf{c}'$ every transition from $\mathcal{R}$ is applicable.

So far, only which reaction is applicable was discussed. Now, we are going to talk more about behaviour of the system. Firstly, chemical kinetics is discussed. Chemical kinetics describes rates of the reactions. Kinetic behaviour is conventionally studied first by determining how is the reaction rate affected by external factors like temperature or concentrations of the species. In this work, mass action kinetics is assumed. It is kinetic scheme, which says that the rate of a chemical reaction is proportional to the product of the concentrations of the reacting chemical species. That means, rate of the reaction is multiplied with concentration of reactants of current state. Temporal behaviour of the CRN is described as a Markov process, represented with set of Ordinary Differential Equations (ODE), which is referred to as Chemical Master Equation (CME). CME describes the CRN as stochastic process. Reason for stochasticity in CRNs is that monomolecular reactions always involves quantum mechanics and bimolecular reactions requires collisions, which are so sensitive to initial conditions, that they appear essentially randomly. Stochastic kinetics can differ a lot from deterministic models, because a system might follow very different scenarios with non-zero likelihoods. [4, 7, 9, 25]

There exists two main approaches, how to compute evolution of the system in time: simulation-based and numerical-based techniques. An idea behind simulation-based techniques is to simulate behaviour of the system. Such simulation produces single realization, but not the proper analysis. Although simulation is typically faster than numerical analysis, many simulations must be performed to obtain a good accuracy and reveal even a less probable behaviours and it can be very time consuming. Simulation technique uses Stochastic Simulation Algorithm (SSA). SSA generates time to the next reaction and index of that reaction (which reaction will be performed). Time τ to the next reaction is computed as an exponential random variable and index j of reaction is an integer random variable. For generating random numbers, the unit-interval uniform random number generator is used. It produces pseudo-random numbers from the uniform distribution. [12]

Numerical-based approach typically transforms CRN to Continuous Time Markov Chain (CTMC). Such transformation is accurate representation of the CRN, but can possibly lead to infinite state space and even when the state space is finite, analysis is not scalable due to state-space explosion. More about analysis of CTMC will be discussed in the next section.

Neither of the approaches can provide required solution easily. Various abstractions and approximation were introduced to reduce computational time of analysis of CRNs.
For numerical based approach, various reduction techniques for stochastic models exists. Widely studied reduction method is state aggregation based on lumping [6] or (bi-)simulation

[3] equivalence. Also several approximate aggregation schemes leveraging the structural properties of CRNs were proposed [11, 17, 26]. Another method of state-space reduction is based on removing states, which can be reached only with very small probability.
For simulation-based approach, partitioning schemes for species and reactions have been proposed, in order to speed up SSA in multi-scale systems.[13, 20, 21]

2.2 Continuous Time Markov Chain

Markov chains are state transition systems, where choice of successor state is determined by probabilistic choice. That means, successor of state s is chosen by probability distribution. Markov chains also satisfies Markov property, which means behaviour of the system demands only on the current state and not the path system took before. This is also known as memoryless property. Discrete Time Markov Chain (DTMC) models system with discrete time steps. Markov chains modelling continuous time behaviour are called Continuous Time Markov Chains (CTMCs).[3]

We start with some terms needed to understand how CTMC works. Random continuous variable is random variable, where possible values comprise single interval or union of disjoint intervals on the number line . As it is interval, there is infinity possible values, we can take. Knowing that, we can say that probability taking exact single value is equal to zero. Positive probability lies somewhere in intervals. It can be shown by probability density function (PDF). PDF shows probability that random variable falls into particular range of values e.i. interval. This can be computed by integrating PDF in this range. Integrate over whole PDF is equal to 1. Continuous random variable X is exponential with parameter $\lambda > 0$ if its PDF is given by:

$$f_X(t) = \begin{cases} \lambda \cdot e^{-\lambda \cdot t} & \text{if t>0} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

If we want to know what is the probability, that random variable X will take value less or equal to variable x, we can use cumulative distribution function (CDF). CDF $F_X(x)$ means exactly that. We can get CDF of exponential variable X by integrating its PDF:

$$F_X(t) = \int_0^t \lambda \cdot e^{-\lambda x}\, dx = [-e^{-\lambda \cdot x}]_0^t = 1 - e^{-\lambda \cdot t} \tag{2.2}$$

Note that exponential distribution is the only distribution, which is memoryless. We can show that

$$Pr(X > t_1 + t_2 | X > t_1) = Pr(X > t_2)$$

Due to this feature, CTMC satisfies Markov property. In CTMCs, exit times for individual state is given by exponential distribution.

Now, we can move to formal definition of CTMC:
CTMC C is a tuple (S, S_{init}, R, L), where S is finite set of states, $S_{init} \in S$ is the initial state, $R : S \times S \to \mathbb{R}_{\geq 0}$ is transition rate matrix and $L : S \to 2^{AP}$ is a function labelling the states with atomic propositions.

Transition rate matrix assigns rate to every pair of states. This rate is parameter λ to the exponential distribution. That means for every rate $R(s, s')$ probability, that transition is triggered before time t is $1 - e^{-R(s,s')\cdot t}$. Whenever there is multiple transitions

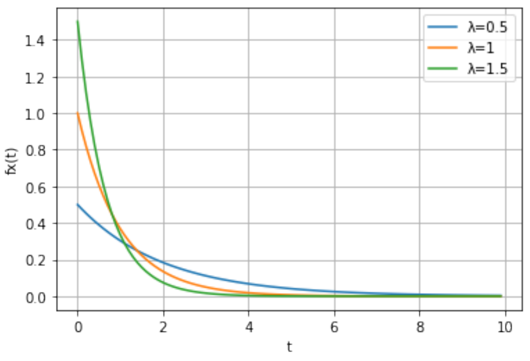


Figure 2.1: Probability distribution function for exponential random variables

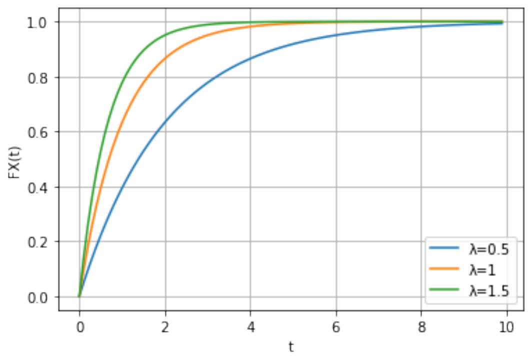


Figure 2.2: Cumulative distribution function for exponential random variables

from one state, first transition triggered determines the next state. This is called the race condition. If we want to know the probability of leaving the state s before t time units, we simply sum all transition rates from state s. This is called the exit rate. Then, exit rate is used as parameter to exponential distribution, so probability of leaving state s in time [0,t] is equal to $1 - e^{-E(s)\cdot t}$ If exit rate of some state is equal to zero, such state is called absorbing. Embedded Discrete Time Markov Chain can also be derived from CTMC. Embedded DTMC is independent to time and only probabilities of transitions are covered. Embedded DTMC derived from CTMC C is defined as:
$emb(C) = (S, S_{init}, \mathbf{P}^{emb(C)}, L)$, where S, S_{init}, L is the same as in C and $\mathbf{P}^{emb(C)} : S \times S \to [0,1]$ is transition probaility matrix and is defined as:

$$\mathbf{P}^{emb(C)}(s,s') = \begin{cases} R(s,s')/E(s) & \text{if } E(s) > 0 \\ 1 & \text{if } E(s) = 0 \text{ and } s = s' \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

Example 2.2:
Let‘s have an example of CTMC shown below in the Figure 2.3. Note that this CTMC is derived from CRN shown in Example 2.1. In the figure, reactions are denoted as p for production, d for degradation and b for blocking. Also, D_{on} and D_{off} are shown as discrete states, in fact, they are states [1,0,x] and [0,1,x]. As one can see, this is CTMC with infinite number of states.

State-space of such CTMC derived from CRN is given by combination of every population of each species, reachable from initial CRN state (in this case D_{on}). Transitions are derived using mass-action kinetics, so transition rates are given by multiplication of reaction rates and population of reactants. For instance, degradation rate of state $[D_{on}, 1]$ is $1 \cdot 0.1 = 0.1$, and for state $[D_{on}, 51]$ it is $51 \cdot 0.1 = 5.1$.

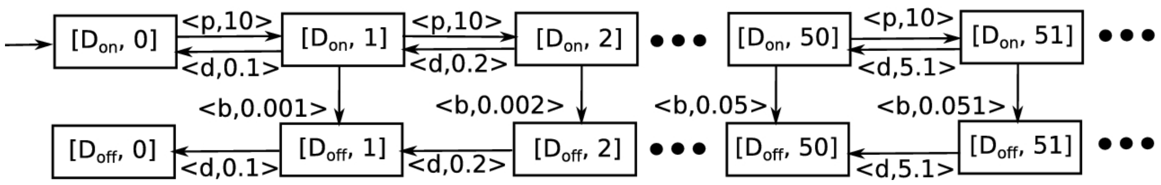


Figure 2.3: Example of infinite state CTMC derived from CRN.

For the purpose of demonstrating key terms, consider that there is bound for number of proteins as in Figure 2.4.

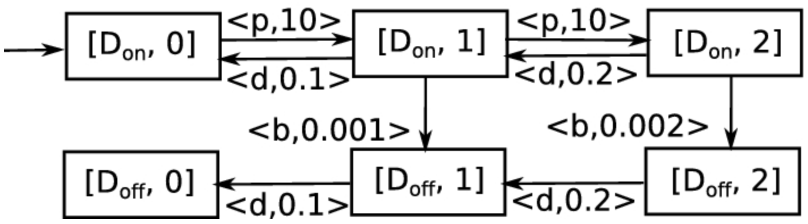


Figure 2.4: Example of CTMC

This CTMC has just six states: $[D_{on},0],[D_{on},1],[D_{on},2],[D_{off},0],[D_{off},1],[D_{off},2]$. Transition matrix $\mathbf{R}$ is shown below. We can compute exit rates for every state: $[D_{on},0]=10$, $[D_{on},1]=10.101$, $[D_{on},2]=0.202$ and so on. With those, embedded DTMC, where transition matrix $\mathbf{P^{emb(C)}}$ represents probabilities going to another state not considering time can be computed.

$$\mathbf{R}=\begin{pmatrix} 0 & 10 & 0 & 0 & 0 & 0 \\ 0.1 & 0 & 10 & 0 & 0.001 & 0 \\ 0 & 0.2 & 0 & 0 & 0 & 0.002 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0 \end{pmatrix}$$

Path in CTMC is sequence of states and time intervals. Time intervals represent time spent in the previous state. We can define infinite path as:

$$s_0, t_0, s_1, t_1, s_2, t_2...,$$

where $\mathbf{R}(s_i, s_{i+1}) > 0$ and $t_i \in \mathbb{R}_{>0}$ for all $i \in \mathbb{N}$. Rate between state and its successor must be greater than zero because otherwise it means there is no transition to successor and probability of such path is always equal to zero. Also time interval must be greater then 0, because probability of leaving the state in no time is again always zero. Finite path is defined as:

$$s_0, t_0, s_1, t_1, ..., t_{k-1}, s_k,$$

where $\mathbf{R}(s_i, s_{i+1}) > 0$ and $t_i \in \mathbb{R}_{>0}$ for all $i < k$. s_k is absorbing.

Cylinder set Cyl(ω) is the set of infinite and finite paths, where ω is the common prefix of those paths. We can obtain probability space by those cylinder sets:

$$\begin{aligned} & Pr_s(Cyl(s) = 1) \text{ and} \\ & Pr_s(Cyl(s, l, s_1, l_1, ..., l_{n-1}, s_n, l', s') \\ =& Pr_s(Cyl(s, l, s_1, l_1, ..., l_{n-1}, s_n) \cdot P^{emb(C)}(s, s') \cdot (e^{-E(s_n)\cdot inf\, l'} - e^{-E(s_n)\cdot sup\, l'}), \end{aligned}$$

where $P^{emb(C)}(s_n, s')$ means the probability of reaching state s' from state s_n using the embedded DTMC. $e^{-E(s_n)\cdot inf\, l'} - e^{-E(s_n)\cdot sup l'}$ is probability, that the time spent in state s_n

is within the interval of l'. (inf and sup stands for infimum and supremum, i.e. minimum and maximum of time interval).

Example 2.3:
Consider CTMC in Figure 2.4. from Example 2.2 and Cylinder:
$Cyl([D_{on}, 0], (0, 1], [D_{on}, 1])$
We are interested in computing probability of this path:

$$\begin{aligned}
&Pr_{[D_{on},0]}(Cyl([D_{on}, 0], (0, 1], [D_{on}, 1])) \\
=&Pr_{[D_{on},0]}(Cyl([D_{on}, 0])) \cdot P^{emb(C)}([D_{on}, 0][D_{on}, 1]) \cdot (e^{-E([D_{on},0])\cdot 0} - e^{-E([D_{on},0])\cdot 1}) \\
=&1 \cdot 1 \cdot (e^{-10\cdot 0} - e^{-10\cdot 1}) \\
=&1 - e^{-10} \\
\approx&0.9999546
\end{aligned}$$

When talking about analysis of CTMC, interest is often in transient and steady state behaviour. Transient behaviour describes, how the state of the model at particular time looks like e.i. it gives us probability distribution at given time. Steady state behaviour describes state of the model similarly in a long-run.

2.2.1 Transient and steady state analysis

Transient probability $p(t) = \mathbb{P}[X(t) = s|X(0) = s_0]$ describes probability, that after time $t \geq 0$ residing state is $s \in S$. This can be obtained by various methods. One way might be getting all cylinder sets leading to s with time length t and integrating through them. This approach is computationally infeasible. Another ways are solving a system of differential equation or as a matrix exponential and therefore evaluated as a power series. These otions are possibly computational unstable. Probabilities are instead computed using uniformised DTMC. Rough idea of uniforization of CTMC is transforming into discrete time system, but in contrast with embedded DTMC, uniformization considers largest exit rate, so inter-state transitions are normalized by largest exit rate. Also self loops for states, where probability of leaving transitions is not summing up to 1, are introduced.
Formally, Uniformised DTMC $\mathbf{P}^{unif}$ is:

$$\mathbf{P}^{unif}(s, s') = \begin{cases} \mathbf{R}(s, s')/q & \text{if } s \neq s\prime \\ 1 - E(s)/q & \text{if } s = s' \end{cases} \tag{2.4}$$

where $q \geq max\{E(s)|s \in S\}$ is the uniformization rate. Such uniformized DTMC can track probability distribution in given number of jumps. Transient probability after k steps is denoted as $\mathbf{u_k} = \mathbf{p_0} \cdot (\mathbf{P}^{unif})^k$. We can also compute probability of performing k jumps in time t. It is so, because discrete jumps are independent and exponentially delayed with rate q. Probability of k jumps occurring at time t is random variable with Poisson distribution: $\mathbb{P}[k \text{ jumps}] = e^{-qt}\frac{(qt)^k}{k!} =: \Psi_{qt}(k)$. When probability distribution after k steps is multiplied by probability of k steps occurring at time t, result is part of the distribution at time t. To receive whole distribution, it is needed to sum for every possible k:

$$\mathbf{p}_t = \sum_{k=0}^{\infty} \Psi_{qt}(k) \cdot \mathbf{u}_k$$

It is needed to sum over the interval from zero to infinity, because even for very large k, there is probability greater then zero, that k jumps is performed, but it might be very small. Summation of infinity elements is indeed problematical. Instead, under approximation via partial sum can be used:

$$\mathbf{s_k} = \sum_{i=0}^{k} \Psi_{qt}(i) \cdot \mathbf{u}_i < \mathbf{p}_t$$

There is also a better solution, which uses fact that probability of performing k steps at time t is handled by Poisson distribution and so interval of values k, where $1 - \epsilon$ of probability lies can be computed. ϵ is a given error. So we get truncation bounds L and R and approximation of $\mathbf{p_t}$:

$$\mathbf{s_{L,R}} = \sum_{k=L}^{R} \Psi_{qt}(k) \cdot \mathbf{u}_k < \mathbf{p}_t$$

can be performed. Then, probability missing in this approximation is less or equal to λ. Truncation points L and R can be obtained by Fox&Glynn‘s scheme, when given error λ. As mentioned, steady state behaviour can be also analysed. Steady state $P_\infty(s) = lim_{t\to\infty} \mathbf{p}_t(s)$, this limit exists for every finite CTMC and intuitively, it represents percentage of time spent in each state in a long run. [16, 3, 23]

2.3 Population abstraction

As mentioned earlier, one of the problems with analyzing CRNs is that CRN often leads to a large state state space CTMC, possibly even infinite for unbounded populations (as shown in Figure 2.3). Rough idea is to merge multiple states into intervals of populations. It is extended with treating sequences of actions, called acceleration, which was introduced in [8, 7]

More concretely, given CRN $\mathcal{R}$, every species $\lambda \in \Lambda$ is divided into finite intervals reflecting rough size of population. Also, intervals keep track whether the transitions are enabled. That means, λ includes interval $\{0\}$ (for species where in reactant is only 0 or 1) and also also intervals for every integer less then maximum number that can be found in reactants part of reactions before λ. The intervals define the abstract states.
Transitions from abstract states are defined according to may abstraction This is solved in order with ideas of classic may abstraction introduced in [14]. Since the abstract states keep track whether are transitions enabled, transitions from abstract states are given by every successor reachable from concrete states contained in the abstract state. Rate of those reaction is smallest interval including every transition rate from concrete enabled transitions. Example of this abstraction for the CTMC shown in Figure 2.3 is shown in Figure 2.5 As one can see, there are 4 intervals. 0 is automatically included, then [1,20], [21,50] and [51,1000].

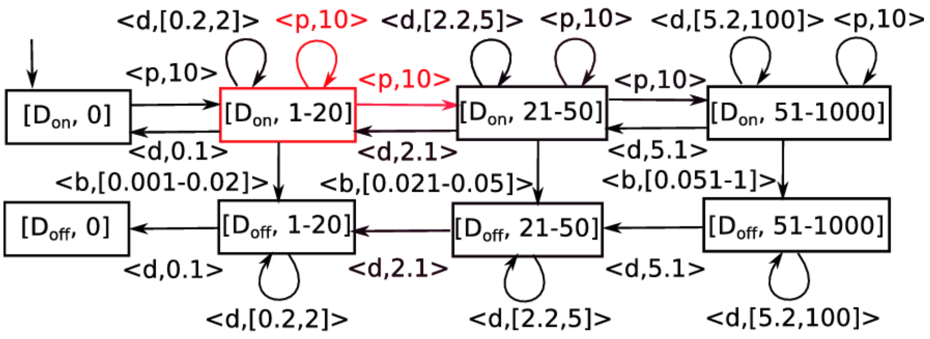


Figure 2.5: Example of population abstraction

This abstraction is problematic as it includes high-level of non-determinism. Non-determinism is connected with the abstract population sizes and so abstract transitions to different abstract states happens only non-deterministically. That means, it cannot be determined which transition is the most probable one and so probable behaviour of the system also cannot be determined. It is marked as red in the example above.
To solve this non-deterministic self loops and transitions leading to higher or lower abstract states, these transitions are dropped. Instead, sequence of actions leading to another abstract state is considered and simultaneously self-loops are taken multiple times. As result, one transition, which represents how the system moves to abstract state with higher population interval is produced. It *accelerates* their effect and so it reflects change to another different abstract state happening in real system. This idea is called acceleration.

When acceleration is applied to system shown in Figure 2.5, it can be seen, that non-determinism is no longer included and accelerated transitions took place (denoted with A). Red transition indicates accelerated transition, that replaced non-deterministic transitions also marked red earlier. (Figure 2.6)

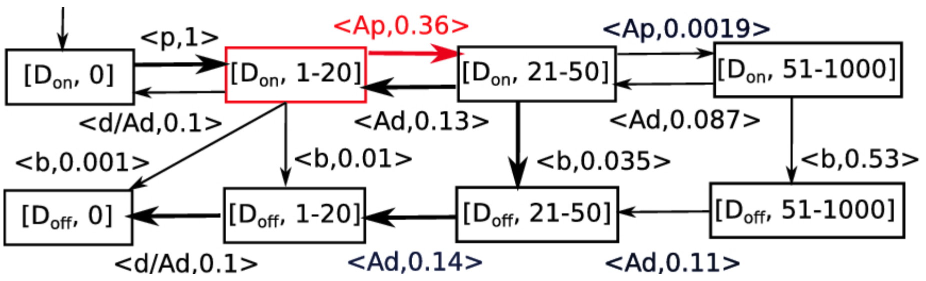


Figure 2.6: Example of applied acceleration to abstraction.

Further is described, that CTMC abstraction can be improved with using new Dirac Semi-Markov Process.

Chapter 3

Dirac Semi-Markov Process for population abstraction

In this chapter, novel Dirac Semi-Markov Process abstraction for CRNs is introduced. Also, it is shown, why it reflects behaviour of the system better then CTMC abstraction.

3.1 Motivation

In the CTMC abstraction, every transition has assigned just one rate and so expected time of transition is independent to its successor, but it does not capture behaviour of population model of CRN well. In CRNs expected time highly depends on the transition. It can be demonstrated on the following example.

Example 3.1 Let N be a CRN over species $S = \{A, B\}$ with initial state $2A + 1000B$ and the following two reactions: $A \xrightarrow{0.04} \emptyset$ and $B \xrightarrow{0.001} B + B$. Consider initial abstract state $\{(a, b) \mid 1 \leq a \leq 3 \wedge 975 < b < 1025\}$. Interest is in which direction and when will system leave the initial abstract state. It is shown in Figure 3.1. It is clear, that time in which is the abstract state left is not exponentially distributed as in CTMC abstraction, but it is rather bell-shaped with very few transitions, that are much faster than expected. Further, time until the system leaves depends on the actual successor (which transition will be taken). Time until the system leaves the state in direction A is 15 time units and for direction B it is 24 time units. This cannot be modeled by CTMC, where exit time of leaving abstract state is 20.4 and it is independent of the successor. That is why new abstraction is used. It has to allow modelling exit times depending on the actual successor.

In order to enable times of transitions to depend on the successor, Dirac Semi-Markov Process (DSMP) is introduced, where probabilities and expected leaving times are separated and also transition times have no variance.

Formally, Dirac Semi-Markov Process is a tuple $D = (S, s_0, \mathbf{P}, \mathbf{Q})$, where S is a finite set of *states*, $s_0 \in S$ is an *initial state*, $\mathbf{P}\colon S \times S \to [0, 1]$ is a *transition probability matrix*, and $\mathbf{Q}\colon S \times S \to \mathbb{Q}_{\geq 0}$ is a matrix assigning a *conditional expected waiting-time* for each transition. The state s is *absorbing* if $\mathbf{P}(s, s) = 1$.
Path ω in D is defined as non-empty sequence of states and times: $s_0, t_0, s_1, t_1, \ldots$, where

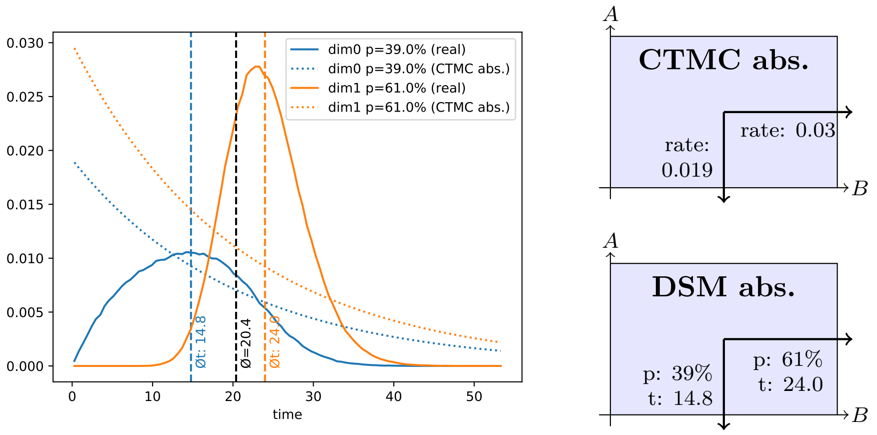


Figure 3.1: (left) Fraction of runs that leave the abstract state of Example 3.1 for a given point in time. Blue runs leave in dimension A, orange runs in dimension B. Full lines show actual behaviour of system. Dashed lines show expected time for leaving in a dimension. The dotted lines show the behaviour of the CTMC abstraction. (top right) Initial abstract state in CTMC abstraction where transitions only have a rate. (bottom right) Initial abstract state in improved DSM abstraction where transitions occur with probability “p” and take exactly time “t”.

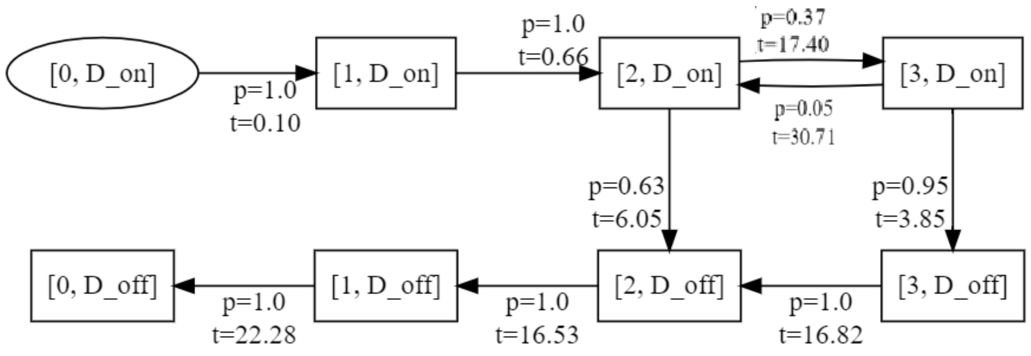


Figure 3.2: Example of DSMP population abstraction.

$\mathbf{P}(s_i, s_{i+1}) > 0$ and $\mathbf{Q}(s_i, s_{i+1}) = t_i$. i-th state of the path can be denoted as $\omega(i)$ for $i \geq 0$ and s_i is meant. State, where path is located in $t \in \mathbb{Q}_{\geq 0}$ is denoted as $\omega@t$. It is ωj, where j is the smallest index for which $\sum_{i=0}^{j} t_i \leq t$. Set of all paths starting in state s is referred to as: $Path_s$. Probability of such set of paths can be computed and is denoted as Pr_s over $Path_s$. Transient probability at time $t \in \mathbb{Q}_{\geq}0$ π_t^D assigns probability in interval $[0, 1]$ to every state and sum of probabilities is equal to 1. It is defined as $\pi_t^D(s) = Pr_{s_0}\{\omega \in Path_{s_0} | w@t = s\}$. The steady-state is then defined as $\pi_\infty^D(s) = \lim_{t\to\infty} \pi_t^D(s)$.

Example of the DTMC abstraction of SimpleGene model is shown in Figure 3.2. Intervals for protein population are: {0}, [1,10], [11, 100], [101, 1000]. These intervals are denoted as abstract states 0, 1, 2 and 3. The other species are again denoted as discrete states D_{on} or D_{off} instead of [x, 1,0] or [x,0,1]. Note, that every transition is assigned with its probability and time that it takes transition to be fired.
Let’s describe now, how such system behaves. At the beginning systems starts in the initial

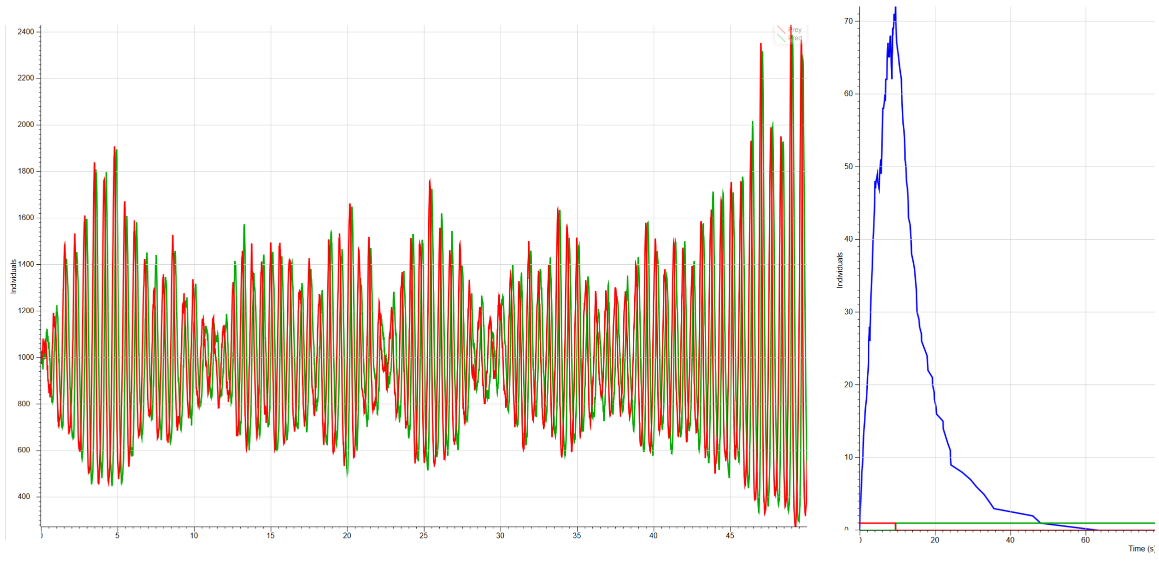


Figure 3.3: (left) Simulation of CRN of PredatorPrey model (right) Simulation of CRN of SimpleGene model. Blue line is population of proteins

state $[0, D_{on}]$. Then, two transitions with probability 1 are fired and all of the probability mass is in state $[2, D_{on}]$ at time 0.76 (sum of exit times of fired transitions.). From this state, with probability 0.37, transition leading to state $[3, D_{on}]$ is taken. If this happens, time in this state would be 18.16 time units. and and with probability 0.63, blocking transition leading to state $[2, D_{off}]$ is fired. Transition takes 6.05 time units, so time there would be 6.81.

3.2 Comparison of DSMP x CTMC abstraction

In this section, we experimentally demonstrate that, that DSMP population abstraction behaves more similarly to actual CRN than CTMC abstraction using simulations. We compare the typical simulations of the original CRN with CTMC and DSMP-based abstraction using the same number of popultation levels. We discuss that already this very rought comparsion demonstrates that DSMP-based abstraction is more precise than CTMC-based abstraction. For every introduced example, simulation illustrating the most typical behaviour is shown.

We will show first PredatorPrey model, where it is visible, that new abstraction is fundamentally more precise than CTMC abstraction. In Figure 3.3 left simulation run over CRN model is shown. It shows the most typical behaviour of the system. Firstly, it is compared to the typical simulation run over CTMC population abstraction (Figure 3.4 left). Then, the same comparison is made with DSMP population abstraction in Figure 3.4 right. One can see, growth and reduction of oscillation amplitude in CRN simulation run. This behaviour is also visible in DSMP abstraction simulation, but it does not appear in CTMC abstaction simulation. Also, number of oscillations in the CTMC model is lower compared to the other two simulations. These indicates, that DSMP is more accurate abstraction.

Its behaviour can be seen in Figure 3.3 right, where simulation of this model is shown. As we can see, typical behaviour of this model is quick generating of proteins and then, after DNA is blocked, protein population instantly starts to fall quickly. When compared to Figure 3.5, it is visible, that DSMP abstraction (right) is more accurate compared to

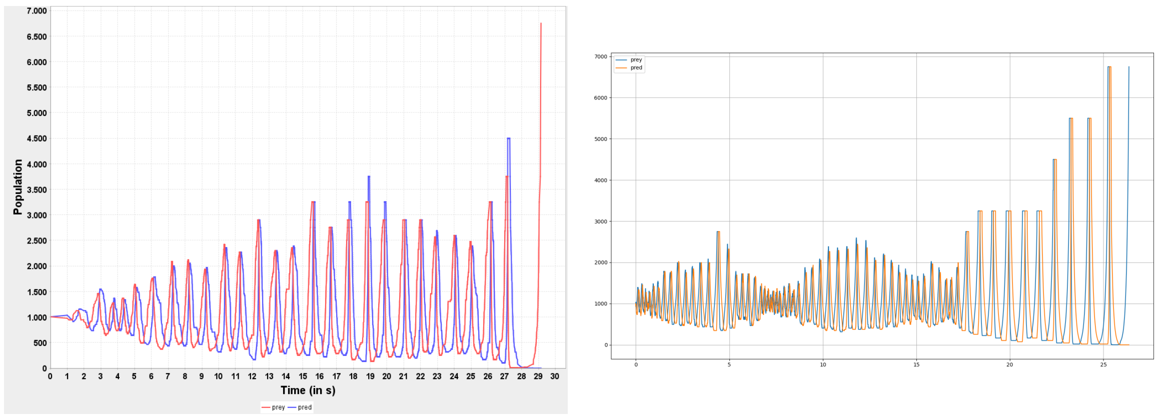


Figure 3.4: (left) Simulation of PredatorPrey model over CTMC population abstraction for 100 levels (intervals). (right) The same simulation as for left, but for DSMP population abstraction.

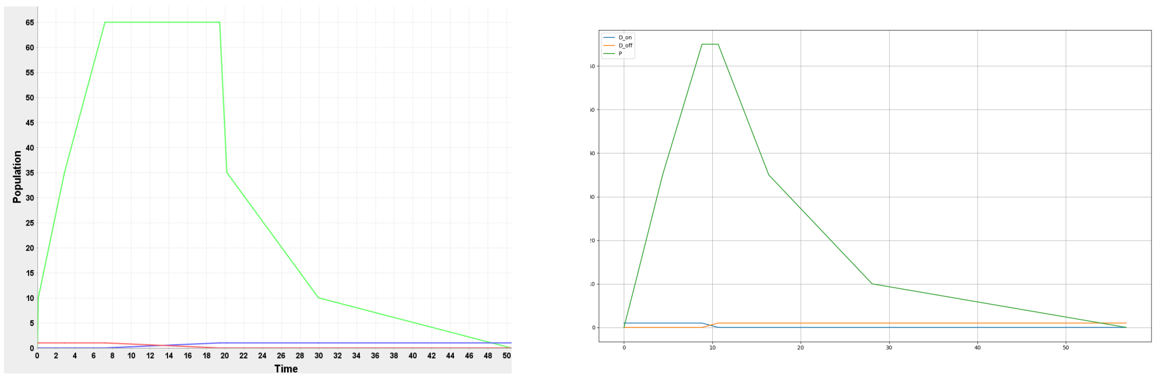


Figure 3.5: (left) Simulation of SimpleGene model over CTMC population abstraction. (right) The same simulation as for left, but for DSMP population abstraction.
Green line is population of protein.

CTMC abstraction (left), because CTMC abstraction stays in the peak moment for quite some time unlike in the simulation of CRN. DSMP abstraction simulation grows fast and after DNA is blocked protein population starts to fall with almost no delay.

Chapter 4

Specification Language

Principle of model-checking is to check, whether the model satisfies desired property. To verify this, two main ingredients are needed. One of those is to have an appropriate model, which was discussed earlier.

The second ingredient is the temporal logic formula, which allows to express desired property. When talking about behaviour of biological models, interest is often in one of the following properties: reachability properties, temporal ordering of events, variable correlations, (multi)stability properties, monotonic trends and oscillation properties. Some examples of these properties are shown using Linear Time Logic:

- Reachability property expresses reachability of concentration level of species.
  $\mathbf{F}(10 < A < 20)$ expresses that A reaches concentration level between 10 and 20 during the model dynamics.
- Stability properties holds for every state in the path. Example of this property might be $\mathbf{G}(A \geq 2)$, which means, that A concentration is always above 2.
- Multi-stability queries for existence of several different stable states.
  $[(A \leq 5) \Rightarrow \mathbf{G}(A \leq 5)] \wedge [(A > 5) \Rightarrow \mathbf{G}(A > 5)]$
  In this formula, two different stable states are included: A is bellow concentration 5 and A is above this concentration level.

4.1 Logics for biologically relevant properties

Temporal logics are convenient way to formalize and verify properties like properties mentioned earlier. They can be generally divided into two basic logical formalisms: Linear Time Logic (LTL) [19] and branching time Computational Tree Logic (CTL) [10].

To express important quantitative properties like time aspect, stochasticity or energy costs, various extensions for mentioned logics exists. We can divide these extensions into two categories: deterministic and stochastic logics.

Deterministic logics often extends logics with notion of time. CTL time extension is called TCTL [1]. It adds clock constraints to the language. Dense time extension of LTL is Metric Interval Logic (MITL) [2]. It is restricted Metric Temporal Logic (MTL) [15]. It is based on timed until operator. Similar operator is used in this work and is described in detail later. Another logic is Signal Temporal Logic (STL) [18]. It combines dense time of MITL and the numerical predicates over real numbers.

Regarding stochastic logics, they specifies probability and performance measures over the Markov chains. Discrete time Markov chains (DTMCs) are covered with Probabilistic Computation Tree Logic (PCTL) and Probabilistic Liner Temporal Logic (PLTL) , which is probabilistic extension to CTL and LTL respectively, adding probabilistic operator P to those logics. Regarding continuous time logic is called Continuous Stochastic Logic (CSL). In this work, focus is on this logic as it works for stochastic continuous time models. [5]

4.1.1 Continuous Stochastic Logic

CSL is extension of the non-probabilistic temporal logic CTL. Key additions to CTL are probabilitic operator P and steady state operator S.
Syntax of CTL is:

$$\begin{aligned}\Phi &::= true \mid a \mid \Phi \wedge \Phi \mid P_{\sim p}[\phi] \mid S_{\sim p}[\Phi]\\ \phi &::= X\Phi \mid \Phi U_I \Phi,\end{aligned}$$

where a is an atomic proposition, I interval of $\mathbb{R}_{\geq 0}$, $p \in [0,1]$ and $\sim \in \{<,>,\leq,\geq\}$. Φ is a state formula and ϕ is a path formula. In fact, CSL formula is always state formula and path formula occurs only inside the P operator. $P_{\sim p}[\phi]$ indicates that probability, that path formula ϕ from a state is satisfied meets the bound $\sim p$. $\Phi U^I \Psi$ is bounded until formula, which is satisfied, when Φ stands until time instant t, which is in interval I, when Ψ is satisfied. 'Unbounded' until can also be derived by considering $I = [0, \infty)$. Steady-state operator $S_{\sim p}$ describes probability being in state meets the bound $\sim p$. $s \models \Phi$ indicates, that CSL formula Φ is satisfied in a state s and denote by $Sat(\Phi)$ the set $\{s \in S \mid S \models \Phi\}$. Also, we write $\omega \models \phi$ when formula ϕ is satisfied by path ω. Also, temporal operators $\diamond$ and $\square$ can be used. These operators stands for „eventually“ for $\diamond$ and „always“ for $\square$.

4.1.2 CSL model checking

Model checking of that CSL is based on reducing the problem to computing backward transient analysis. When given CSL formula and CTMC, firstly CSL formula is reduced to simple formulas and these are model checked. Transient probability cannot be computed directly on original CTMC, because it cannot verify or quantify probability of these formulas. To obtain wanted result, CTMC needs to be transformed. An idea of this transformation is to recognize states from which is formula always satisfied and states, from which is CSL formula always violated. We can recognize these states, because they are or are not labeled with atomic preposition or set of atomic prepositions. These states are then typically made absorbing and transient distribution is computed over this transformed CTMC. Then results of all of the simple formulas given together. [16, 3, 23]

4.2 Multiple Until

In this work, main focus is on the temporal ordering of atomic events and oscillation properties. Temporal ordering of atomic events is based on until operator, where $\varphi_1 \; U \; \varphi_2$ means that φ_1 holds until some eventual state, where φ_2 holds. Example of usage of this formula might be a $(P \leq 10) \; U \; [(10 < P \leq 30) \; U \; (P > 30)]$. This formula expresses that population of proteins (P) is initially bellow 10 until it grows to interval between 10 and 30 until it finally reaches population above 30. Motivation for using temporal ordering of events to be able to express property such as bell shape property:

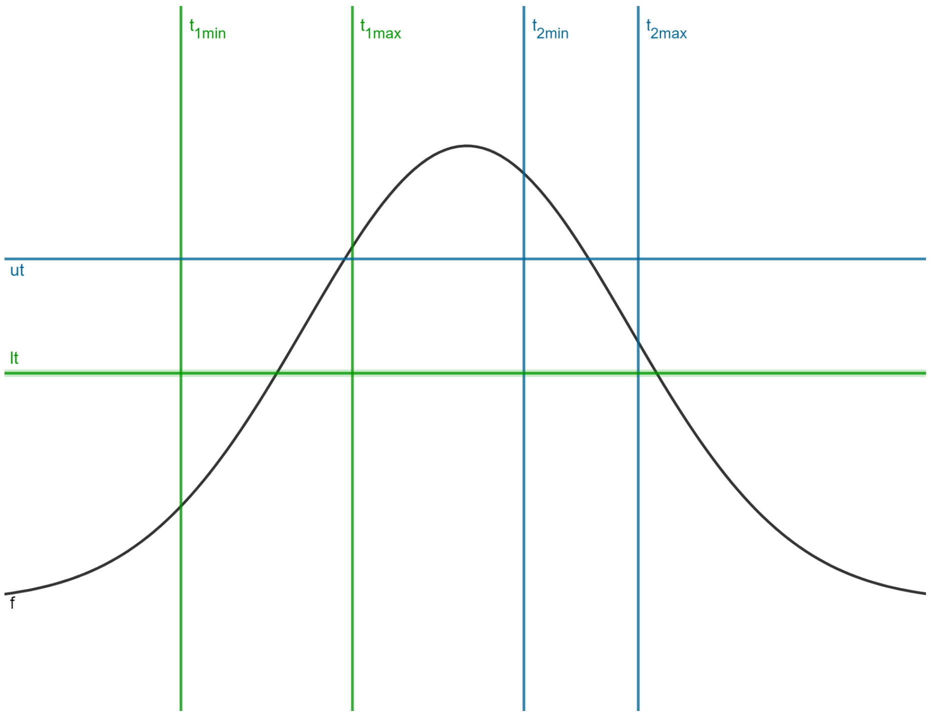


Figure 4.1: Example of bell shape property

Bell shape property over given species is given by intervals t_1, t_2 and two thresholds lt, ut. Bell shape is satisfied by the path if and only if:

- Initially, population is under the threshold lt.
- Within the interval t_1, population goes above lt.
- Within the interval t_2, population goes below ut.

Example of bell shape and how is it defined is demonstrated in Figure 4.1. This property appears very often in biological systems, because it describes production followed by degradation of species. Quantifying chance of such behaviour can help analyzing those systems.

To be able to express properties similarly to LTL formula above in probabilistic models with reasonable computation time, extension to CSL is introduced. This extension uses multiple bounded until operators. It is given in the following form.

$$\varphi = \Phi_1 U^{I_1} \Phi_2 U^{I_2} \dots U^{I_{k-1}} \Phi_k \text{ for k} \geq 2$$

, where Φ_i is Boolean combination of atomical propositions over the set of states S and I_i is time interval. For simplification, overlapping of time intervals is not allowed, so $inf I_i \geq sup I_{i-1}$. Formula is satisfied if formula Φ_1 stands, until time instant $t_1 \in I_1$, where Φ_2 stands. Φ_2 then have to hold again to time instant $t_2 \in I_2$, from where Φ_3 holds and so on until last interval, where if exists time instant, where Φ_k is satisfied, it is enough, similarly to until operator in CSL. Formally, satisfaction relation $\models$ over paths ω is defined as:

$$\omega \models \varphi \Leftrightarrow \exists 0 \leq t_1 \leq t_2 \dots \leq t_k - 1 \text{ such that}$$
$$\omega(t_{k-1}) \models \Phi_k \wedge \forall 0 < i < k : t_i \in I_i \wedge \forall t' \in [t_{i-1}, t_i) : \omega(t_{k-1}) \models \Phi_i,$$

where t_0 denotes 0.
Using an introduced formula, bell shape can be expressed as:

$$\varphi = (P < lt)\ U^{t_1}\ (P > lt)\ U^{I_2}\ (P < ut)$$

4.3 Oscillation property

Another property discussed in this work is oscillation property. Oscillation typically means, that population oscillates between higher and lower values. Instance of LTL formula expressing permanent oscillation of population A around population 1000:

$$(\mathbf{G}[(A \leq 1000) \Rightarrow \mathbf{F}(A > 1000)]) \wedge (\mathbf{G}[(A > 1000) \Rightarrow \mathbf{F}(A \leq 1000)]),$$

where **G** means Globally and **F** means Future. „Globally“ operator means, that every state in the path must satisfy formula that follows the operator. Future operator is satisfy, when eventually somewhere in the path, exists state that satisfies formula. $\Rightarrow$ is intuitive, so whenever left side holds, right side also must hold to satisfy whole formula. So, stated formula can be interpreted as: All the time, if A is bellow 1000, eventually A must grow over 1000 And similarly All the time, if A is over 1000, eventually A must drop under 1000.

Stated LTL formula is too restrictive for biological oscillation. Also, model-checking of such LTL formulas is computationally very intensive. To solve this, „bounded $\exists$ operator“ is introduced as an extension to CSL. It is denoted as : $\exists^I \Phi$, where Φ is again Boolean combination of atomic prepositions and I is time interval. This formula is satisfied if there exists state at time t in path ω, that satisfies Φ. This formula is basically **F** operator restricted with time interval. Multiple of these can be combined to express desired property as following:

$$\varphi = \exists^{I_1}\Phi_1\ \exists^{I_2}\Phi_2 \ldots \exists^{I^k}\Phi_k$$

Again, overlapping of time intervals is not allowed. This formula is satisfied if there exists time instant $t_i \in I_i$, where Φ_i is satisfied for every $i \leq k$.

Oscillation using this formula can be expressed as:

$$\exists^{0,10}(A \leq 1000)\ \exists^{10,20}(A > 1000)\ \exists^{20,30}(A \leq 1000)$$

This can be interpreted as: in time interval [0,10], system hits state that has population A below 1000 once, somewhere in time interval [10,20], population grows over 1000 and in [20,30] there exists state that has A population again bellow 1000. That means if time intervals are long enough, it surely catches oscillation of the system.

This specification language has 3 important features:
- the semantics is reasonably simple such that the biologist can understand it and use it
- can capture important biologically relevant patterns including timed-reachability, monotonic trends, bell-shape patterns or bounded oscillation
- verification of the specification is computationally tractable.

Chapter 5

Model checking of Dirac Semi-Markov processes

In this chapter, algorithm for computing transient distribution as well as chain of untils is further described.

5.1 Stepping algorithm

Recall, that transient distribution provides probability distribution in the given time. This is fundamental model checking information and is used for further analysis like model checking of more complicated properties expressed by logic specification formulas.

Discrete Time Markov Chains have transitions equipped only with probabilities. Computing its transient distribution is easy. At given number of discrete steps n, transient distribution π_n is given by $\pi_n = \pi_0 \cdot \mathbf{P}^n$, where π_0 is initial distribution and $\mathbf{P}$ is probability matrix. However recall that in Dirac Semi-Markov Process, transitions are equipped except from probability also with exit time to fire this transition. This makes it more difficult to compute transient distribution as it is needed to track time. Rough idea of algorithm to compute transient distribution is to remember for how long is the probability mass waiting in the state. Then, when transition waited long enough i.e. exit time of transition, transition is fired.

More concretely, current distribution together with number of steps for which did particular probability mass waited is stored in tables. Tables are iteratively computed, since it keeps data for different times. Time step is given by greatest common divisor of all the transitions together with inspected transient time. It is because it needs to catch every transition to inspected transient time. There is upper bound of the number of waiting times, which is longest exit time of the transition divided by time step and this number is size of the tables. Transient probability for a state is than obtained by summing column of the final table with that state. To always move the correct amount of mass, also another history tables are computed, which stores the probability mass entering each state in a particular step together with the number of steps performed since it entered the state.
Formally for DSMP $D = (S, s_0, \mathbf{P}, \mathbf{Q})$, output is function: $\Pi_t^D(s_0) : S \to [0, 1]$ giving transient probabilities to every state in S at time t.
Let step size Δ be defined as:

$$\Delta = \gcd(\{y | \exists x : \mathbf{Q}(x) = y\} \cup \{t\})$$

Upper bound of the number of waiting times used for the size of the tables:

$$\Gamma = \max_{s,s' \in S}(\mathbf{Q}(s,s'))/\Delta$$

Also, two tables are used:

- Table $\tau : \{1, ..., \mathit{stepRange}\} \times S \to [0,1]$ to store the current probability distribution together with the number of steps for which did particular probability mass remains in the given state
- Table $\alpha : \{1, ..., \mathit{stepRange}\} \times S \to [0,1]$ to store the probability mass entering each state in a particular step together with the number of steps performed since it entered the state. This is necessary to always move the correct amount of probability mass when performing a transition in table τ.

These tables are recursively computed as follows:

$$\tau_0 = \alpha_0 = \{(1, s, 0) \mid s \in S \wedge s \neq s_{init}\} \cup \{(1, s_{init}, 1)\}$$

$$\begin{aligned}\tau_{i+1} =& \{(\mathbf{x}+1, s, p-p') \mid (\mathbf{x}, s, p) \in \tau_i \wedge \mathbf{x} \leq \Gamma \wedge \\ & p' = \sum_{s' \in Succ_{\mathbf{x}}(s)} \mathsf{P}(s,s')\alpha_i(\mathbf{x}, s)\} \cup \\ & \{(1, s, p) \mid s \in S \wedge p = \sum_{\mathbf{x} \in \{1,...,\Gamma\}} \sum_{s' \in Pred_{\mathbf{x}}(s)} \mathsf{P}(s', s)\alpha_i(\mathbf{x}, s')\}\end{aligned}$$

$$\begin{aligned}\alpha_{i+1} =& \{(\mathbf{x}+1, s, p) \mid (\mathbf{x}, s, p) \in \alpha_i \wedge \mathbf{x} \leq \Gamma\} \cup \\ & \{(1, s, p) \mid s \in S \wedge p = \sum_{\mathbf{x} \in \{1,...,\Gamma\}} \sum_{s' \in Pred_{\mathbf{x}}(s)} \mathsf{P}(s', s)\alpha_i(\mathbf{x}, s')\},\end{aligned}$$

where $Succ_s$ and $Pred_s$ are functions, which return successors and predecessors of state.... s ,respectively, available via a transition with waiting time $r \cdot \Delta$. Finally, we define the output probability distribution:

$$\mathbf{\Pi}_t^D(s_0) = \{(s,p) \mid p = \sum_{(\mathbf{x},s,p') \in \tau_{t/\Delta}} p'\}$$

Example 5.1 This algorithm can be demonstrated on this simple example shown in Figure 5.1. We have three state DSMP there. Circles inside the states represents probability mass. Coloured transitions in τ represents enabled transitions about to be fired. Coloured transitions in α are enabled transitions together with transitions already taken. Numbers inside the probability mass circles are times for how long probability mass waited in the current state. This demonstrates that until all the transitions for given state are taken, original probability mass is kept to be able to calculate the correct values in τ. We go through this example step by step. At the beginning, in τ all the probability mass is in the initial state. One colored transition is enabled. α_0 table is equal as no transitions were taken yet. In τ_1 we can see, that enabled transition in τ_0 was fired, so probability mass moved to another state. There are two enabled transitions now. In α_1 we can see that since not all transitions were taken yet from initial state probability mass, it stays there. Also except from enabled transitions, first taken transition is coloured. To construct τ_2 it is needed to look at α_1 and move probability mass from enabled transitions. So one can see, blue transition in τ_1 was fired and 0.4 of probability mass in α_1 was moved. Note that since not all transitions were fired from initial state yet, in α_2 are still coloured transitions already taken. If we wanted to know probability distribution at time 2Δ, we need to sum up all probability masses in states in τ_2.

This basic idea is further developed in Implementation chapter, where approximation for this algorithm, allowing to compute transient distribution in less time with reasonable precision for some models, is introduced.

5.2 Chain of untils

In this section, model checking of DSMP using chain of untils will be discussed. It is important part of this work as it allows to check some biologically interesting properties like bell-shape or oscillation of the population of species. Also, it uses simple notation, so it can be used and understood by biologists.
As described before chain of untils is sequence of Boolean combination of atomical propositions and bounded until operator:

$$\varphi = \Phi_1 U^{I_1} \Phi_2 U^{I_2} \dots U^{I_{k-1}} \Phi_k \text{ for k} \geq 2$$

Example 5.2
Recall when is such formula satisfied on this example:
$\varphi = \Phi_1 U^{I_1} \Phi_2 U^{I_2} \Phi_3$, where $I_1 = [1,3]$, $I_2 = [5,7]$. We go through conditions, which must be met for path ω to be satisfied. First of all, every state included in ω under time 1, which is left corner point of I_1, must satisfy Φ_1. Next, consider time instant $t_1 \in I_1$. For ω to satisfy φ, there must exist t_1, that at time before t_1 every state satisfies Φ_1 and at time interval $[t_1, 3]$ (3 is right interval of I_1) every state satisfies Φ_2. In interval [3,5] (3 is right corner point of I_1 and 5 is left corner point of I_2), Φ_2 must be satisfied in every state. For last until operator, there must exist state in ω, that satisfies Φ_3 in interval I_2. Instance of path satisfying φ can look like this:

$$\Phi_1, \Phi_1|^1 \Phi_1 \wedge \Phi_2, \Phi_1|^{t_1} \Phi_2, \Phi_2|^3 \Phi_2, \Phi_2, \Phi_3|^{\leq 5},$$

where $|^t$ means time instant.

5.2.1 Formal algorithm of chain of untils

Now we can move to algorithm of chain of untils $\varphi = \Phi_1\ U^{I_1}\ \Phi_2\ U^{I_2}\ \dots\ U^{I_{k-1}}\ \Phi_k$, where $I_i = [b_i, e_i]$
Notation used in the algorithm

- $D[\Phi]$ denotes DSMP D where states $s \in S$ satisfying Φ are absorbing
- $\pi_t^D[\Phi](s) = \pi_t^D(s)$ if $s \vDash \Phi$ otherwise $\pi_t^D[\Phi](s) = 0$

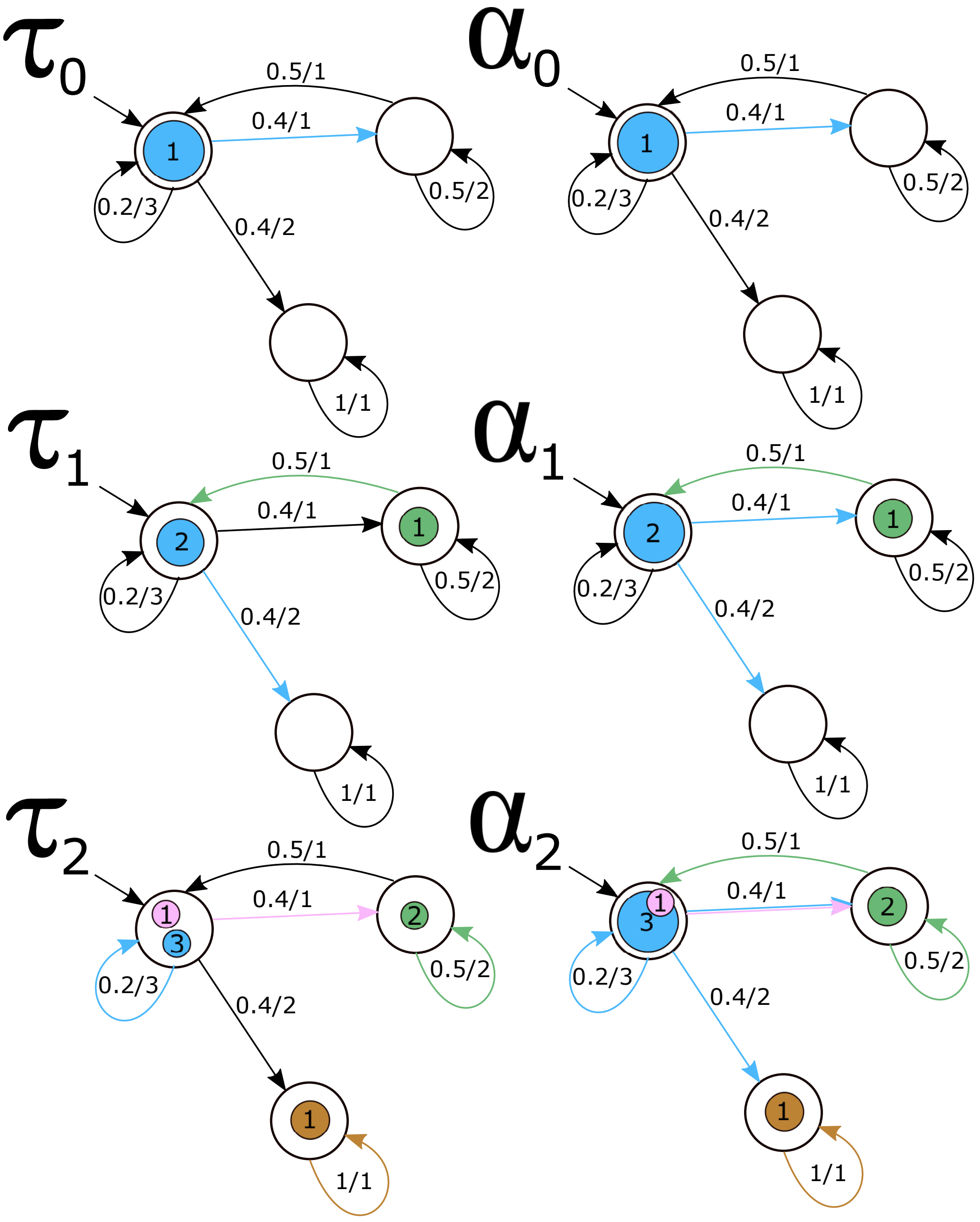


Figure 5.1: An example of the stepping algorithm

Algorithm 1: Model checking algorithm for multiple untils

```
Input  : A DSMP $D$ and formula $\varphi = \Phi_1 \; U^{I_1} \; \Phi_2 \; U^{I_2} \; \dots \; U^{I_{k-1}} \; \Phi_k$.
Output: The probability that $s_0 \vDash \varphi$
$D_1 \leftarrow D[\neg\Phi_1]$
run $\pi_{b_1}^{D_1}$ from $s_0$
$\mu_1 = \pi_{b_1}^{D_1}[\Phi_1]$
for $i \in \{2 \dots k-1\}$ do
    $D_i \leftarrow \mathtt{build}(D, \Phi_{i-1}, \Phi_i)$, $t \leftarrow e_{i-1} - b_{i-1}$
    compute $\pi_t^{D_1}$ from initial distribution $\mu_{i-1}$
    $\mu_i(s) = \pi_t^{D_1}[\phi_2](s') + \pi_t^{D_1}[\phi_2](s'')$
    $D_1 \leftarrow D[\neg\Phi_i]$, $t \leftarrow b_i - e_{i-1}$
    compute $\pi_t^{D_1}$ from initial distribution $\mu_i$
    $\mu_i = \pi_t^{D_1}[\Phi_i]$
$D_i \leftarrow D[\neg\Phi_{i-1} \wedge \Phi_i]$, $t \leftarrow e_{i-1} - b_{i-1}$
run $\pi_t^{D_1}$ from initial distribution $\mu_{i-1}$
return $\sum_{s \in S} \pi_t^{D_i}[\Phi_i]$
```

Algorithm 1 firstly needs to remove probability mass, that violated Φ_1 at $[0, t_1]$. This is made in similarly to model-checking CSL formulas. Transient analysis over transformed DSMP, where states satisfying $\neg\Phi_1$ are absorbing is computed. Then, only probability mass in states satisfying Φ_1 is considered. This is covered in the first 3 lines giving a probability distribution μ_1

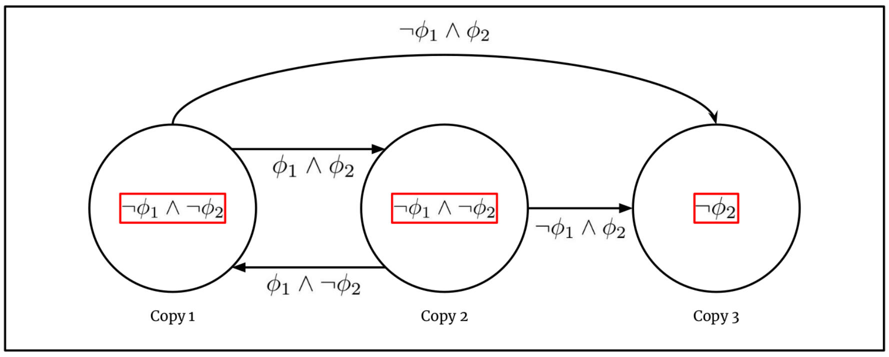
Then, for every until operator we get DSMP with copies using a `build` function, that builds DSMP D_i demonstrated in Figure 5.2. Then, transient analysis over D_i is performed from initial distribution μ_{i-1} for time $b_{i-1} - e_{i-1}$. This traps probability mass, that violates formula in states satisfying $\neg\Phi_i$, so this probability is discarded in $\mu_i(s)$. Also, Φ_i must hold between e_{i-1} and b_i. This is solved again by transforming D, so states satisfying Φ_i are absorbing. Transient analysis is performed over this transformed DSMP for time e_{i-1} and b_i from initial distribution μ_i and probability mass in states made absorbing is then discarded.

After this loop, transient analysis over DSMP D_i is performed with initial distribution μ_{i-1} for time $b_{k-1} - e_{k-1}$. D_i has states satisfying $\neg\Phi_{k-2} \wedge \Phi_{k-1}$ absorbing. This time, only probability mass in absorbing states satisfying Φ_{k-1} is considered and returned as the result.

5.2.2 Demonstration of chain of untils algorithm on example

When given a chain of untils formula $\varphi = \Phi_1 \; U^{I_1} \; \Phi_2 \; U^{I_2} \; \Phi_1$, the algorithm for model-checking this formula is is decomposed to a few steps. These steps are described using the same notation as in the algorithm.

- **First step**
  In this step, not satisfying paths are discarded at time interval $[0, b_1]$. Recall, that in this interval, every state in satisfying path must satisfy Φ_1. In this step, every state not satisfying Φ_1 is made absorbing and with this DSMC is computed transient distribution for time b_1. This way, all the unwanted probability remains in absorbing states and is discarded.



$\neg\phi_1 \wedge \phi_2$
$\phi_1 \wedge \phi_2$
$\neg\phi_1 \wedge \neg\phi_2$
$\neg\phi_1 \wedge \neg\phi_2$
$\neg\phi_1 \wedge \phi_2$
$\neg\phi_2$
$\phi_1 \wedge \neg\phi_2$
Copy 1
Copy 2
Copy 3

Figure 5.2: The second step chain. The states satisfying the formulae in red are made absorbing in the respective copies. The transitions are labeled with the formula that is satisfied by the successor state, i.e. whenever a successor state satisfies the formula on the transition, a jump to another copy is performed.

- **Second step**
  This step removes probability violating formula at time interval I_1. In this interval Φ_1 must hold until some time instant, from where every state must satisfy Φ_2. To verify, that path satisfies this, 3 copies of the original chain and modify some transitions to target to another copy state. This is demonstrated in Figure 5.2. Idea is, that in Copy 1 is probability mass, which satisfies $\Phi_1 \wedge \neg\Phi_2$. Copy 2 is place for probability mass satisfying both $\Phi_1 \wedge \Phi_2$ and in Copy 3 is probability which satisfies Φ_2, but already was in state, which does not satisfy Φ_1. Also states in Copy 1 and Copy 2 satisfying $\neg\Phi_1 \wedge \neg\Phi_2$ are made absorbing and states in Copy 3 satisfying $\neg\Phi_2$ are made absorbing. Probability mass caught in those states is then discarded. Probability distribution from first step is initial distribution to this step and transient distribution for time $e_1 - b_1$ is computed and probability in states satisfying $\neg\Phi_2$ is discarded.
  So, to sum up, probability mass, which satisfies and always satisfied Φ_1 is in Copies 1 or 2 depending on whether current state satisfies Φ_2. Once $\neg\Phi_1$ is seen, probability mass is either moved to Copy 3, if its state currently satisfies Φ_2 or is caught in an absorbing state and discarded later otherwise.

- **Third step**
  Third step is similar to first step. It starts from distribution given by second step and again states not satisfying needed preposition are made absorbing and after transient probability for time between time intervals is computed.

  When formula contains more until operators, second and third steps are performed until time instant b_i, where i is the index of last time interval.

- **Last step**
  This step works similarly to first and third step, but not all probability mass in absorbing states is discarded. States satisfying $\neg\Phi_{i-1} \vee \Phi_i$, where Φ_i is last preposition in φ, in this example $\neg\Phi_2 \vee \Phi_3$, are made absorbing, transient distribution is computed and probability in states not satisfying Φ_i is discarded.

Example 5.3:
In Figure 5.3 left is shown simple example of DSMP on which constructing of copies in

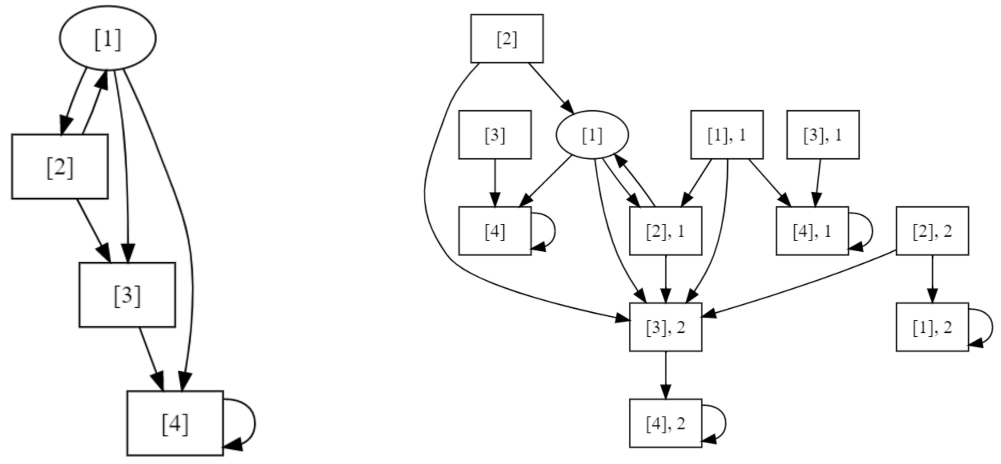


Figure 5.3: (left) Example of DSMP. (right) Demonstration of making copies and transition between them in Second step of checking chain of untils from example in left. Copies 2 and 3 are denoted with numbers 1 and 2. State [1] satisfies $\Phi_1 \wedge \neg\Phi_2$, state [2] satisfies $\Phi_1 \wedge \Phi_2$, state [3] satisfies $\neg\Phi_1 \wedge \Phi_2$ and finally state [4] satisfies $\neg\Phi_1 \wedge \neg\Phi_2$

the second step of chain of untils model checking algorithm can be shown (right). To demonstrate that, probabilities and exit times of transitions can be ignored. Consider, that states state [1] satisfies $\Phi_1 \wedge \neg\Phi_2$, state [2] satisfies $\Phi_1 \wedge \Phi_2$, state [3] satisfies $\neg\Phi_1 \wedge \Phi_2$ and finally state [4] satisfies $\neg\Phi_1 \wedge \neg\Phi_2$. In right figure, states in copies 2 and 3 are denoted with 1 and 2 and states in copy 1 remains without additional label. As one can see, transitions from copy 1 leading to state [2] is redirected to copy 2. also transitions leading to state [3] are redirected to copy 3. State [4] remains absorbing and probability there is discarded. Note that state [1] in copy 3 is made absorbing as well, since it satisfies $\neg\Phi_2$

5.2.3 Bounded exists operator

As mentioned earlier, also „bounded exists“ operator is introduced. Roughly, it is a weak variant of the multiple until operator allowing to capture some behaviour better. Recall, that path satisfies $\exists^{t_1}\Phi$, if in interval t_1 there is at least one state that satisfies Φ. It is allowed to combine this with another „bounded exists“ formulas or with chain of untils formula. That means it is needed to keep track of probability mass that already satisfied formula and so solution, where only satisfying states are made absorbing cannot be used. Similarly to second step of chain of untils algorithm, a copy of the system is added. Copy 2 serves for catching probability that already satisfied formula. For that reason, probability, that starts in state satisfying Φ is moved into the appropriate copy 2 state. Also, transitions leading into state satisfying Φ are redirected to appropriate copy 2 state. This way, probability mass, that already „saw“ state satisfying Φ is in copy 2 states and rest is still in the copy 1. At the end, probability mass in the copy 1 is discarded.

Chapter 6

Implementation

In this chapter we discuss how described features are implemented. Also we describe encountered problems and its solution.

The developed implementation is part of the tool SeQuaia [7]. This tool is developed within research cooperation between Verifit group and group from TU Berlin. This tool is written with Java language and includes important features I extended within my implementation. These are data structures for a state-transition system derived from CRN and methods implemented over it. My main contribution is implementation of verification of multiple until and bounded exist operators, that uses transient analysis and so it was needed to add some features to its class as well as to abstraction class I worked with.

Key data for functionality of this tool are states and transitions of the abstraction. These are stored in the generic transition system class called Automaton. States are there stored as Set of states and transitions are stored as Map, where keys are Pair of states (source and target state) and values are of type Quantities, which stores probability and exit time of the reaction. Also, there is a Map of successors of the state, where key is the source state and as a value there is set of target states and Quantities of the transition.

In Figure 6.1 there is diagram of the architecture of the tool. From the input CRN, abstraction is computed. Abstraction is a class that extends abstract class Automaton. Abstraction is input for a Transient analysis, which computes transient distribution for a given time. In Verification class, original abstraction is stored together with instance of transient analysis.

6.1 Stepping algorithm

Stepping algorithm computes transient analysis and is described earlier. To compute transient distribution, apart from time it is needed to submit two parameters. These are maximal expected time and step size. These parameters are important for tables, where probability mass is stored. They determine size of the two tables from algorithm, which is computed as $maxExpectedTime/stepSize$. Problem with this approach is that computation time for stiff systems is slow, because of large table size, even with a few transitions actually happening.

To solve that event-based approximation is used. It pre-computes reactions happening in the future (events) and whenever transition or set of transitions is fired, new set of events is computed i.e. probability mass, that was just moved to another state has not yet planned transitions. These are computed and added to already planned events. For some systems,

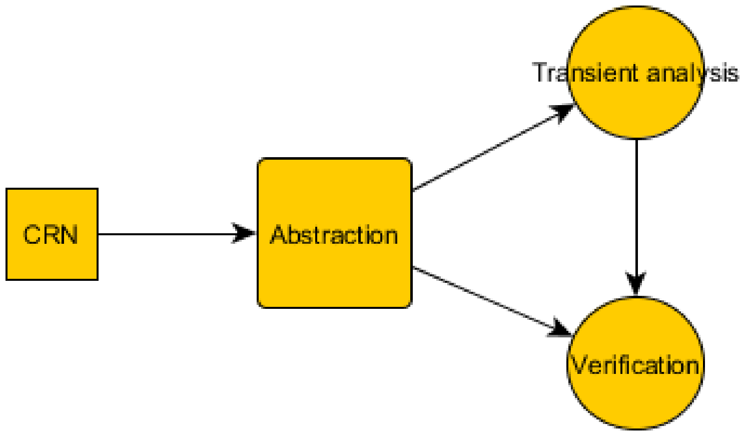


Figure 6.1: Diagram of the architecture of the tool.

this approach is much faster as it does not need to iterate through tables of large sizes. Note, that this is approximation as the method computing transient analysis this way takes as an argument precision parameter. This parameter is considered, whenever some event is fired. It is taken time t of that event and every event that is smaller then t + precision parameter is fired simultaneously. That means larger the parameter is, precision is smaller, but computation faster. This is shown in the experimental part.

6.2 Chain of untils

Recall, that algorithm for quantifying chain of untils formula is roughly based on different steps, where some states are made absorbing or copies of the system are introduced and transitions between them modified and then transient analysis is performed. Modified distribution obtained from this transient analysis is then used as initial distribution to the next step.

At the beginning of the computation, new instance of the original abstraction is made. This instance is modified and transient analysis is run over it. Since the original abstraction was stored, again, new instance of the original abstraction is made and transient analysis over this modified abstraction is performed. This is made several times, until whole formulae is quantified. This corresponds to individual steps of the algorithm. Note, that all of the transient analysis computations are made over one instant of the transientAnalysis class. This is so, because it needs to remember transitions waiting to be fired. Except from abstraction, probability distribution computed by transient analysis also needs to be modified between steps.

Abstraction needs to be modified when performing single step of the algorithm. The main challenge when implementing it was to create copies of the original states. This is made by making exact state space as in original system with the same transitions. That means, new states are added to the set of abstraction states. Created states are marked as copies and also identifier to recognize different copies is assigned. Original states are mapped to copy states, so it is possible to get appropriate copy state. This is important when redirecting transitions to copy states. This way, copies are introduced to the system and then, transitions between them are modified, transient probability is computed and unwanted probability mass is discarded.

The other parts were easier to apply, as only certain states marked with certain atomic preposition are made absorbing, transient analysis is performed and unwanted probability mass is discarded.

Tricky thing to implement was that input DSMP and probability distribution needs to be changed, because some probability mass might be discarded. Solution for event-based approach is described as it is faster then stepping algorithm and it differs a bit. DSMP can be changed there in process and future planned events are based on the new DSMP. Problem is with already planned events, because they might be leading now to different state (copy state). These planned events need to be changed. Whenever probability mass in certain state is discarded, it is needed to also discard events, which has this state as the source state, because there is no longer probability mass to move.

Another implemented feature is moving probability mass from a certain state. This needs to be performed, because some probability mass might need to already begin in another copy. In this case, events, that have such state as source state are changed. Typically, it is needed to change source state as well as target state, because another copy state transition probably leads to different state than original state.

Similar thing needs to be performed after computing transient distribution with copies. There are typically planned events, that has source state or/and target state in some copy. Problem with this is that in the next computation, these states are no longer in the system. That means, probability mass in those copies states are moved to original states and transitions are modified, so source and target states are right.

6.3 Bounded Exists operator

Implementation of verifying bounded exists operator uses most of the described features in chain of untils. It is very similar as it needs to make copy of the original system, some probability mass might need to begin in another state, some states needs to be made absorbing and then copies states are after computation no longer in the system.

6.4 Testing

To verify whether the implementation works as we wanted, several tests were performed. At first, unit tests were triggered off. These tests verified, that individual parts of the code works as desired. Mainly, methods for making states absorbing, creating of the copies and deleting and adding transitions from/to the abstraction were tested. Then, it was needed to check if the transitions between copies are made correctly. This was tested for some not trivial systems using the method toDotty, that visualize states and transitions. Also, the code was debugged and tested on simple models, where behaviour can be tracked.

Chapter 7

Experiments

In this chapter, behaviour of stepping algorithm and model checking of simple CSL formulas as well as model checking of chain of until formulas is tested. The goal is to discover, whether the event-based stepping algorithm is fast enough and how computation time is affected by the precision parameter. Also what is the impact of lower precision on result. Also, comparison between model checking of some CSL formula and chain of untils formula is described in terms of computation time and again, how is this affected by change of precision parameter. This is important to determine, whether proposed algorithms can be used for more complicated models. In another section, results of introduced specification language model-checking interesting biological properties are introduced.

7.1 Used models

For this experimental part SimpleGene model was used. This model was already described in Example 2.1, recall, that it is given by:
$\Lambda = \{D_{on}, D_{of}, P\}$ and
$\mathcal{R} = \{D_{on} \xrightarrow{10} P + D_{on}, P \xrightarrow{0.1} \emptyset, D_{on} + P \xrightarrow{0.001} D_{off}\}$ and starting with D_{on}
On this model, bell shape behaviour can be verified as production and degradation of proteins takes place with very high probability(see Figure 3.3 right).

In order to get more complicated abstractions with a lot of states, 2 models with interesting behaviour were used: PredatorPrey and Goutsias models.

PredatorPrey model includes 2 populations, Predator and Prey population:
$\Lambda = \{Pred, Prey\}$
and 3 reactions:
$\mathcal{R} = \{Prey \xrightarrow{10} Prey + Prey, Prey + Pred \xrightarrow{0.01} Pred + Pred, Pred \xrightarrow{10} \emptyset\}$.
Starting state is 1000Pred + 1000Prey.
An idea about how the model behaves can be shown using DSD simulation tool (Figure 3.3 left). One can see oscillations of populations until one of the populations dies out (in the picture Prey population died out and then Predator population as well). This model also has quadratic number of states to number of levels, so it is easy to control number of states and also get high number of states. since reaction rates are not of very different magnitudes this model is not stiff. Since PredatorPrey models has oscillation behaviour, it can be verified with bounded until formula.

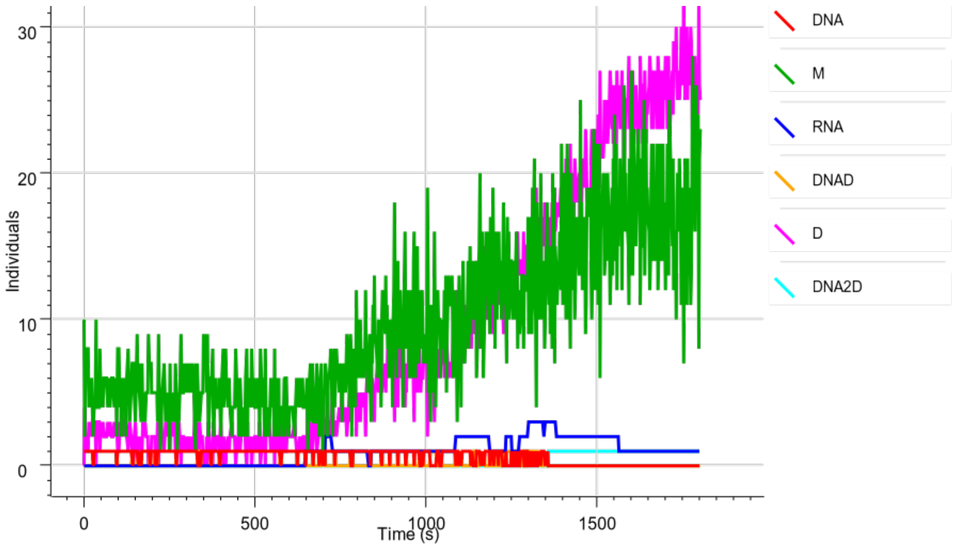


Figure 7.1: Simulation of Goutsias model using DSD tool.

Goutsias model has 3 large populations and leads to an interesting behaviour even in a long runs. It is given by populations
$\Lambda = \{\mathrm{M}, \mathrm{D}, \mathrm{RNA}, \mathrm{DNAD}, \mathrm{DNA2D}, \mathrm{DNA}\}$ with reactions:
$\mathcal{R} = \{\mathrm{RNA} \xrightarrow{0.043} \mathrm{RNA} + \mathrm{M},\ \mathrm{DNAD} \xrightarrow{0.0072} \mathrm{DNAD} + \mathrm{RNA},\ \mathrm{M} \xrightarrow{0.0007} \emptyset,$
$\mathrm{RNA} \xrightarrow{0.004} \emptyset,\ \mathrm{DNA} + \mathrm{D} \xrightarrow{0.02} \mathrm{DNAD},\ \mathrm{DNDA} + \mathrm{D} \xrightarrow{0.0002} \mathrm{DNA2D},$
$\mathrm{DNA2D} \xrightarrow{0.000000000009} \mathrm{DNAD} + \mathrm{D},\ \mathrm{DNAD} \xrightarrow{0.48} \mathrm{DNA} + \mathrm{D},$
$\mathrm{M} + \mathrm{M} \xrightarrow{0.083} \mathrm{D},\ \mathrm{D} \xrightarrow{0.5} \mathrm{M} + \mathrm{M}\}$
and starting state is DNAD + 10M.
Again, simulation of this model is shown in simulation from DSD tool (Figure 7.1). Goutsias model leads to an interesting behaviour even in a longer runs and also is interesting to model-checking of some properties for RNA population. Note, that this model is very stiff, because of that one very slow reaction, so it can represent behaviour on stiff models.

7.2 Performance evaluation

In this section, we evaluate performance of different algorithms. We are mainly interested in usability of event based approximation and how copies introduced in model-checking algorithm of multiple untils and bounded exists formulas impact computation time.

7.2.1 Event-based approximation

In this part, experiments over event-based approximation of stepping algorithm are described. For every experiment firstly, reference solution using highest possible precision was computed and then, time precision parameter was gradually increased, which means precision was lowered. Highest possible precision result is obtained with time precision parameter equal to an exit time of the fastest transition i.e. transition with the lowest exit time. Precision is lowered by multiplying this parameter with a constant, which is gradually increased. This constant determines x axis of a graph. Then, results obtained by computations with lower precision were compared to reference result.

Specifically, computation time and variance were determined. Variance was computed using L1 norm. L1 Norm is the sum of the magnitudes of the vectors in a space. For this case, sum of an absolute values in different states is computed. These two values were plotted into a graph, so results are easily interpreted and visible. To see how computation reacts on different number of states and different times moments, where transient distribution is computed, computations are made for various levels and times. Levels of the model define discretization of populations. There is always one level for empty population and bound for highest possible value of population. For simplification, I denote model only with highest number of levels for the model.

Ideally, computation time would be significantly lower and variance would not reach high values with growing time precision parameter.

In Figure 7.2 is shown that for some models, computation time is significantly lower with higher precision parameter and variance is still low. We can see unstable growing pattern, which might be caused by the way transitions are fired. This graph shows, that for stiff models, lower precision does not have a great variance and computation is much faster.

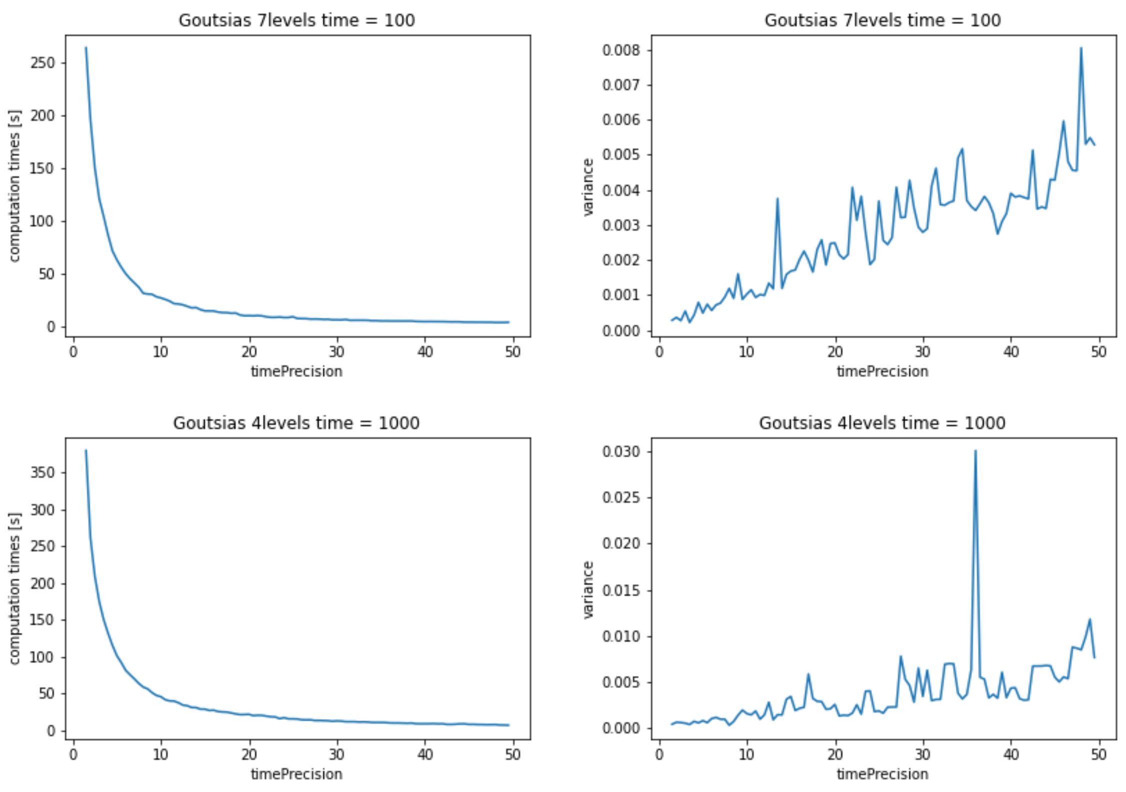


Figure 7.2: (left top) Graph showing computation times of transient analysis for time = 100, with growing time precision parameter, which is determined by fastest reaction exit time multiplied by constant shown at x axis for Goutsias model with 7 levels. (right top) Variance made by gradually higher parameter for the same model. (left bottom) Same graph as left top for 4 levels and time 1000. (right bottom) Same graph as right top for 4 levels and time 1000.

On the other hand, for large models with lot of states, result doesn't look so ideally as we can see in Figure 7.3. Computation time is still significantly lower, but variance for lower precision results makes the result basically useless. For those models, semi-quantitative approach is needed.

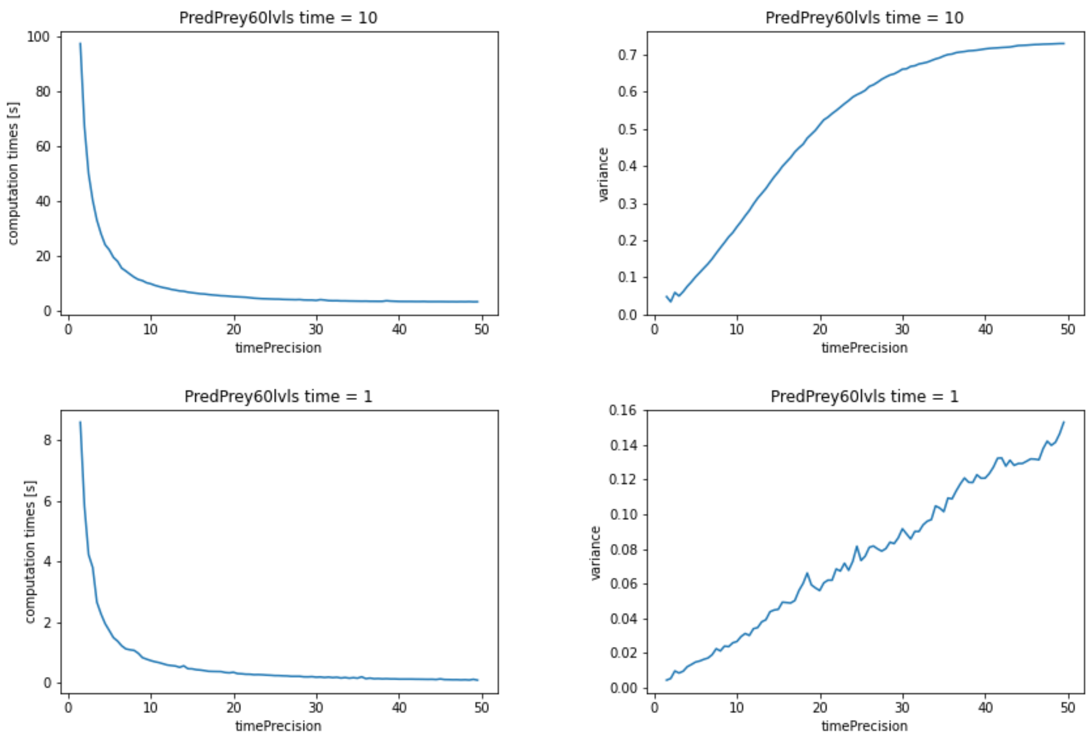


Figure 7.3: (left) Graph showing computation times of transient analysis for time = 10 and time = 1, with growing time precision parameter, which is determined by fastest reaction exit time multiplied by constant shown at x axis for PredatorPrey model with 60 levels. (right) Variances made by gradually higher parameter for the same model.

7.2.2 Multiple untils

With chain of untils formulas, it is possible to verify properties, that are impossible to formulate in classical CSL. The cost for that is much higher computation time. It is so, because of the used copies. For the chain of untils, in the second step, state space is tripled. That has obviously impact on computation times. To find out how big this effect is, comparison of computation times of verifying simple CSL formula and multiple until formula was made. It obviously depends on time, for which is second step performed as it takes most of the computation time. For following experiments, half of the time was in the second step. This comparison is shown in Figure 7.4. For growing number of levels, computation time of multiple untils takes up to 4 times more then simple CSL.

For Predator Prey model bounded exists formula revealing oscillation was tested similarly. Note that state space is doubled for the whole computation when verifying this formula. Computation for bounded exists formula took almost 30 times more then verifying simple CSL formula in some cases.

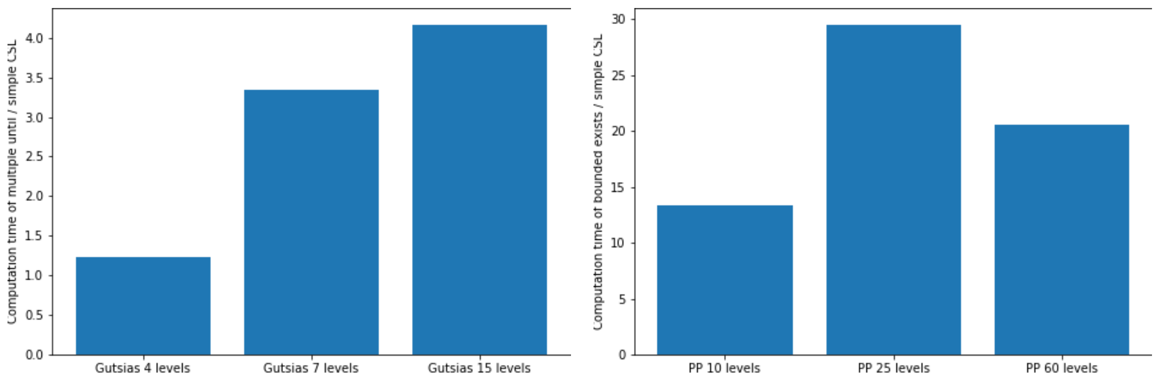


Figure 7.4: (left) Relation of computation times of chain of untils formula and simple CSl for Goutsias model with different levels abstraction. (right) The same comparison for PredatorPrey model.

7.3 Model-checking results

First model-checked property is bell shape property on the SimpleGene model using an multiple untils formula. See the Figure 3.3 right, there protein population is produced and then after DNA is blocked, it extincts. Multiple untils formula is made to be satisfied in this simulation. So it looks like this:

$$\varphi = ((P < 50)\ U^{[3,10]}\ (P \geq 50)\ U^{[12,50]}\ (P < 50))$$

Note, that it actually is satisfied in the shown simulation run as before time 3, population is bellow 50, in time interval [3,10] population exceeds 50 and it holds above 50 in time interval [10,12] and somewhere in time interval [12, 50], population again goes bellow 50. This property is satisfied with probability 35% when verified on abstraction with 5 population intervals. With growing number of intervals, this probability is lowered down to 26% for 30 levels.

To see, why is this smaller, than expected, simulations over used abstractions were made. In Figure 7.5, we can see, that this concrete run does not satisfy formula, because population drops bellow 50 before time 12. Also Proteins are produced faster than accepted by formula. When formula is edited like this:

$$\varphi = ((P < 50)\ U^{[0.1,7]}\ (P \geq 50)\ U^{[7,50]}\ (P < 50)),$$

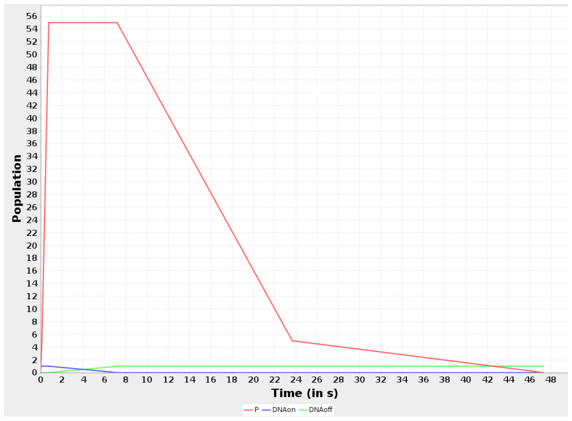
it is already satisfied with probability around 80%, which is close to expectation.

Another checked property is oscillation over the PredatorPrey model. Oscillation is expressed using multiple bounded exists operators:

$$\exists^{[0,10]}(Pred < 600) \wedge \exists^{[10,20]}(Pred > 1350) \wedge \exists^{20,30}(Pred < 600)$$

See the Figure 3.3 left and note that this formula is satisfied in that one simulation run as there is moment under time 10, whenever Predator population is bellow 600, also Predator population goes above 1350 in time interval [10,20] and in time [20,30], there is again moment, when the population is bellow 600. It reveals oscillation behaviour as growing and extincting of the population must be included in longer run. This property is satisfied with probability around 60% for different number of intervals in abstraction not changing much.

This number is lower than expectation given by behaviour in set of simulation runs. This is because in abstraction one of the population species extincts earlier than in CRN. It can be seen in Figure 7.5 right, where simulation over abstraction is shown.



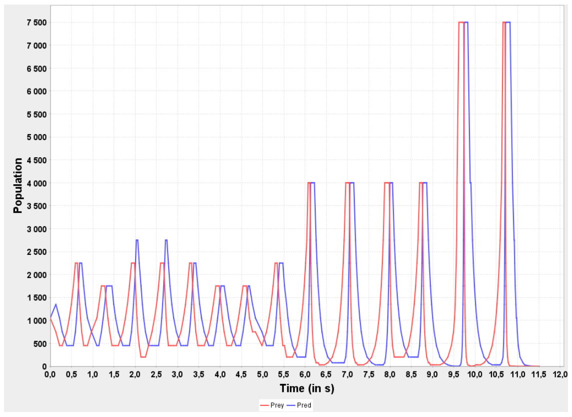


Figure 7.5: (left) Simulation run of SimpleGene abstraction with 20 intervals. (right) Simulation run of PredatorPrey abstraction with 25 intervals.

Chapter 8

Conclusion

In this work, novel abstraction for semi-quantitative approach for analysis of bio-chemical systems was discussed. Also it was shown, that this Dirac Semi-Markov Process abstraction is more precise than previously proposed Continuous Time Markov chain abstraction. Further, for this abstraction, algorithm and implementation for computation of transient analysis is provided. It is shown, that for some models it is possible to significantly lower the computation time with lowering the precision with almost no effect on the result. However this is not possible for models with lots of states, because with lower computation time variance grows fast and so provided result is basically useless. For these models, it is needed to add semi-quantitative reasoning over this algorithm, which might be focus of the future research.

The main focus of this work was introducing novel timed temporal logic formulas, that are able to formulate properties, which are important for biologists. For these formulas, model-checking algorithm was introduced and implemented. It was shown, that multiple until formulas and bounded exists formulas are able to formulate and quantify properties mentioned in the work. Even though this feature allows to quantify wanted properties, it also magnifies state space and so computation is much slower than transient analysis. This can be further optimized in the future. Also, adjustment of this algorithm for the semi-quantitative version of computation of transient analysis is challenge for the further research.

Bibliography

[1] Alur, R., Courcoubetis, C. and Dill, D. Model-Checking in Dense Real-Time. *Information and Computation*. 1993, vol. 104, no. 1, p. 2–34. DOI: https://doi.org/10.1006/inco.1993.1024. ISSN 0890-5401. Available at: https://www.sciencedirect.com/science/article/pii/S0890540183710242.

[2] Alur, R., Feder, T. and Henzinger, T. A. The Benefits of Relaxing Punctuality. *J. ACM*. New York, NY, USA: Association for Computing Machinery. january 1996, vol. 43, no. 1, p. 116–146. DOI: 10.1145/227595.227602. ISSN 0004-5411. Available at: https://doi.org/10.1145/227595.227602.

[3] Baier, C. and Katoen, J.-P. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008. ISBN 026202649X.

[4] Brijder, R. Computing with chemical reaction networks: a tutorial. *Natural Computing*. Mar 2019, vol. 18, no. 1, p. 119–137. DOI: 10.1007/s11047-018-9723-9. ISSN 1572-9796. Available at: https://doi.org/10.1007/s11047-018-9723-9.

[5] Brim, L., Češka, M. and Šafránek, D. *Model Checking of Biological Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. 63–112 p. ISBN 978-3-642-38874-3. Available at: https://doi.org/10.1007/978-3-642-38874-3_3.

[6] Buchholz, P. Exact performance equivalence: An equivalence relation for stochastic automata. *Theoretical Computer Science*. 1999, vol. 215, no. 1, p. 263–287. DOI: https://doi.org/10.1016/S0304-3975(98)00169-8. ISSN 0304-3975. Available at: https://www.sciencedirect.com/science/article/pii/S0304397598001698.

[7] Češka, M., Chau, C. and Křetínský, J. SeQuaiA: A Scalable Tool for Semi-Quantitative Analysis of Chemical Reaction Networks. In: Lahiri, S. K. and Wang, C., ed. *Computer Aided Verification*. Cham: Springer International Publishing, 2020, p. 653–666. ISBN 978-3-030-53288-8.

[8] Češka, M. and Křetínský, J. Semi-quantitative Abstraction and Analysis of Chemical Reaction Networks. In: Dillig, I. and Tasiran, S., ed. *Computer Aided Verification*. Cham: Springer International Publishing, 2019, p. 475–496. ISBN 978-3-030-25540-4.

[9] Chellaboina, V., Bhat, S. P., Haddad, W. M. and Bernstein, D. S. Modeling and analysis of mass-action kinetics. *IEEE Control Systems Magazine*. 2009, vol. 29, no. 4, p. 60–78. DOI: 10.1109/MCS.2009.932926.

[10] Clarke, E., Grumberg, O. and Long, D. Verification tools for finite-state concurrent systems. In: Bakker, J. W. de, Roever, W. P. de and Rozenberg, G.,

ed. *A Decade of Concurrency Reflections and Perspectives*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, p. 124–175. ISBN 978-3-540-48423-3.

[11] Ferm, L. and Lötstedt, P. Adaptive solution of the master equation in low dimensions. *Applied Numerical Mathematics*. 2009, vol. 59, no. 1, p. 187–204. DOI: https://doi.org/10.1016/j.apnum.2008.01.004. ISSN 0168-9274. Available at: https://www.sciencedirect.com/science/article/pii/S0168927408000263.

[12] Gillespie, D. T. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*. 1977, vol. 81, no. 25, p. 2340–2361. DOI: 10.1021/j100540a008. Available at: https://doi.org/10.1021/j100540a008.

[13] Goutsias, J. Quasiequilibrium approximation of fast reaction kinetics in stochastic biochemical systems. *The Journal of Chemical Physics*. 2005, vol. 122, no. 18, p. 184102. DOI: 10.1063/1.1889434. Available at: https://doi.org/10.1063/1.1889434.

[14] Katoen, J.-P., Klink, D., Leucker, M. and Wolf, V. Three-Valued Abstraction for Continuous-Time Markov Chains. In: Damm, W. and Hermanns, H., ed. *Computer Aided Verification*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, p. 311–324. ISBN 978-3-540-73368-3.

[15] Koymans, R. Specifying real-time properties with metric temporal logic. *Real-Time Systems*. Nov 1990, vol. 2, no. 4, p. 255–299. DOI: 10.1007/BF01995674. ISSN 1573-1383. Available at: https://doi.org/10.1007/BF01995674.

[16] Kwiatkowska, M., Norman, G. and Parker, D. Stochastic Model Checking. In: *Proceedings of the 7th International Conference on Formal Methods for Performance Evaluation*. Berlin, Heidelberg: Springer-Verlag, 2007, p. 220–270. SFM'07. ISBN 9783540724827.

[17] Madsen, C., Myers, C. J., Roehner, N., Winstead, C. and Zhang, Z. Utilizing stochastic model checking to analyze genetic circuits. In: IEEE. *2012 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. 2012, p. 379–386.

[18] Maler, O. and Nickovic, D. Monitoring Temporal Properties of Continuous Signals. In: Lakhnech, Y. and Yovine, S., ed. *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, p. 152–166. ISBN 978-3-540-30206-3.

[19] Pnueli, A. The temporal semantics of concurrent programs. *Theoretical Computer Science*. 1981, vol. 13, no. 1, p. 45–60. DOI: https://doi.org/10.1016/0304-3975(81)90110-9. ISSN 0304-3975. Special Issue Semantics of Concurrent Computation. Available at: https://www.sciencedirect.com/science/article/pii/0304397581901109.

[20] Rao, C. V. and Arkin, A. P. Stochastic chemical kinetics and the quasi-steady-state assumption: Application to the Gillespie algorithm. *The Journal of Chemical Physics*. 2003, vol. 118, no. 11, p. 4999–5010. DOI: 10.1063/1.1545446. Available at: https://doi.org/10.1063/1.1545446.

[21] Salis, H. and Kaznessis, Y. Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions. *The Journal of Chemical Physics*. 2005, vol. 122, no. 5, p. 054103. DOI: 10.1063/1.1835951. Available at: https://doi.org/10.1063/1.1835951.

[22] Van Kampen, N. *Stochastic Processes in Physics and Chemistry*. Elsevier Science, 1992. North-Holland Personal Library. ISBN 9780080571386. Available at: https://books.google.cz/books?id=3e7XbMoJzmoC.

[23] Češka, M. and Andriushchenko, R. *Modelling and Analysis of Probabilisitic Systems*. Faculty of Information Technology, Brno University of Technology, February 2021.

[24] Voit, E. *A First Course in Systems Biology*. 2ndth ed. Garland Science, 2017. ISBN 0815344678.

[25] Wolf, V., Goel, R., Mateescu, M. and Henzinger, T. A. Solving the chemical master equation using sliding windows. *BMC Systems Biology*. Apr 2010, vol. 4, no. 1, p. 42. DOI: 10.1186/1752-0509-4-42. ISSN 1752-0509. Available at: https://doi.org/10.1186/1752-0509-4-42.

[26] Zhang, J., Watson, L. T. and Cao, Y. Adaptive aggregation method for the chemical master equation. *International journal of computational biology and drug design*. Inderscience Publishers. 2009, vol. 2, no. 2, p. 134–148.